# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aerial cuBB</td>
<td>3</td>
</tr>
<tr>
<td>1.1</td>
<td>cuBB Installation Guide</td>
<td>3</td>
</tr>
<tr>
<td>1.1.1</td>
<td>Installing Tools on Grace Hopper MGX System</td>
<td>4</td>
</tr>
<tr>
<td>1.1.1.1</td>
<td>Supermicro Grace Hopper MGX Configuration</td>
<td>5</td>
</tr>
<tr>
<td>1.1.1.2</td>
<td>Cable Connection</td>
<td>6</td>
</tr>
<tr>
<td>1.1.1.3</td>
<td>System Firmware Upgrade</td>
<td>8</td>
</tr>
<tr>
<td>1.1.1.4</td>
<td>Install Ubuntu 22.04 Server</td>
<td>11</td>
</tr>
<tr>
<td>1.1.1.5</td>
<td>Configure the Network Interfaces</td>
<td>15</td>
</tr>
<tr>
<td>1.1.1.6</td>
<td>Disable Auto Upgrade</td>
<td>16</td>
</tr>
<tr>
<td>1.1.1.7</td>
<td>Install NVIDIA Optimized Ubuntu Kernel</td>
<td>17</td>
</tr>
<tr>
<td>1.1.1.8</td>
<td>Configure Linux Kernel Command-line</td>
<td>17</td>
</tr>
<tr>
<td>1.1.1.9</td>
<td>Apply the Changes and Reboot to Load the Kernel</td>
<td>18</td>
</tr>
<tr>
<td>1.1.1.10</td>
<td>Install Dependency Packages</td>
<td>18</td>
</tr>
<tr>
<td>1.1.1.11</td>
<td>Install DOCA OFED and Mellanox Firmware Tools on the Host</td>
<td>18</td>
</tr>
<tr>
<td>1.1.1.12</td>
<td>Install CUDA Driver</td>
<td>20</td>
</tr>
<tr>
<td>1.1.1.13</td>
<td>Install GDRCopy Driver</td>
<td>21</td>
</tr>
<tr>
<td>1.1.1.14</td>
<td>Install Docker CE</td>
<td>22</td>
</tr>
<tr>
<td>1.1.1.15</td>
<td>Install the Nvidia Container Toolkit</td>
<td>22</td>
</tr>
<tr>
<td>1.1.1.16</td>
<td>Update BF3 BFB Image and NIC Firmware</td>
<td>23</td>
</tr>
<tr>
<td>1.1.1.17</td>
<td>Install ptpl4 and phc2sys</td>
<td>25</td>
</tr>
<tr>
<td>1.1.1.18</td>
<td>Setup the Boot Configuration Service</td>
<td>30</td>
</tr>
<tr>
<td>1.1.1.19</td>
<td>Running Aerial on Grace Hopper</td>
<td>32</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Installing Tools on Dell R750</td>
<td>32</td>
</tr>
<tr>
<td>1.1.2.1</td>
<td>Dell PowerEdge R750 Server Configuration</td>
<td>32</td>
</tr>
<tr>
<td>1.1.2.2</td>
<td>Converged Accelerator Installation</td>
<td>32</td>
</tr>
<tr>
<td>1.1.2.3</td>
<td>Cable Connection</td>
<td>33</td>
</tr>
<tr>
<td>1.1.2.4</td>
<td>Configure BIOS Settings</td>
<td>34</td>
</tr>
<tr>
<td>1.1.2.5</td>
<td>Install Ubuntu 22.04 Server</td>
<td>36</td>
</tr>
<tr>
<td>1.1.2.6</td>
<td>Disable Auto Upgrade</td>
<td>36</td>
</tr>
<tr>
<td>1.1.2.7</td>
<td>Install the Low-Latency Kernel</td>
<td>36</td>
</tr>
<tr>
<td>1.1.2.8</td>
<td>Configure Linux Kernel Command-line</td>
<td>37</td>
</tr>
<tr>
<td>1.1.2.9</td>
<td>Apply the Changes and Reboot to Load the Kernel</td>
<td>37</td>
</tr>
<tr>
<td>1.1.2.10</td>
<td>Disabling Nouveau</td>
<td>38</td>
</tr>
<tr>
<td>1.1.2.11</td>
<td>Install Dependency Packages</td>
<td>38</td>
</tr>
<tr>
<td>1.1.2.12</td>
<td>Install RSHIM and Mellanox Firmware Tools on the Host</td>
<td>38</td>
</tr>
<tr>
<td>1.1.2.13</td>
<td>Install the CUDA Driver</td>
<td>40</td>
</tr>
<tr>
<td>1.1.2.14</td>
<td>Install the GDRCopy Driver</td>
<td>41</td>
</tr>
<tr>
<td>1.1.2.15</td>
<td>Install Docker CE</td>
<td>42</td>
</tr>
<tr>
<td>1.1.2.16</td>
<td>Install the Nvidia Container Toolkit</td>
<td>42</td>
</tr>
<tr>
<td>1.1.2.17</td>
<td>Update BF3 BFB Image and NIC Firmware</td>
<td>43</td>
</tr>
<tr>
<td>1.1.2.18</td>
<td>Update A100X BFB Image and NIC Firmware</td>
<td>45</td>
</tr>
<tr>
<td>1.1.2.19</td>
<td>Set Persistent NIC Interface Name</td>
<td>48</td>
</tr>
<tr>
<td>1.1.2.20</td>
<td>Install ptpl4 and phc2sys</td>
<td>49</td>
</tr>
</tbody>
</table>
# Aerial Data Lake

2.1 Target Audience .......................................................... 417
2.2 Key Features .............................................................. 417
2.3 Design ................................................................. 418
2.4 Installation .............................................................. 420
2.5 Usage ................................................................. 420
Aerial CUDA-Accelerated RAN brings together the Aerial software for 5G and AI frameworks and the NVIDIA accelerated computing platform, enabling TCO reduction and unlocking infrastructure monetization for telcos.

Aerial CUDA-Accelerated RAN has the following key features:

- Software-defined, scalable, modular, highly programmable and cloud-native, without any fixed function accelerators. Enables the ecosystem to flexibly adopt necessary modules for their commercial products.
- Full-stack acceleration of DU L1, DU L2+, CU, UPF and other network functions, enabling workload consolidation for maximum performance and spectral efficiency, leading to best-in-class system TCO.
- General purpose infrastructure, with multi-tenancy that can power both traditional workloads and cutting-edge AI applications for best-in-class RoA.

What’s New in 24-2

The following new features are available in release 24-2 for Aerial CUDA-Accelerated RAN:

- **Aerial cuPHY**: CUDA accelerated inline PHY
  - MGX Grace Hopper multicell capacity w/ telco-grade traffic model
    - 20 peak loaded 4T4R @ 100MHz
    - Capacity also validated with more challenging traffic model
      - PUSCH and PDCCH symbols in the S-slot
    - L1-L2 interface enhancements
    - Separate FAPI request timelines for PDSCH and PDCCH
- **Aerial cuMAC**: CUDA accelerated MAC scheduler
  - cuMAC-Sch
    - 4T4R CUDA implementation complete
  - cuMAC-CP
    - 4T4R implementation (Functional – early access)
- **Aerial cuBB/E2E**: System level / End-to-End validation
  - Over-The-Air (OTA) validation:
    - CBRS O-RU
    - 8 UE OTA w/ 6 UE/TTI for > 8 hours
  - RedHat-OCP:
    - Multicell capacity validated on MGX (GH200+BF3)
  - O-RAN Fronthaul:
    - 16-bit fixed point IQ sample validated E2E (Keysight eLSU)
    - Simultaneous dual-port FH capability (8 peak cells; 4 per port)
  - L2 integration:
    - Multi-L2 container instances per L1 validated E2E
- **pyAerial**: Python interface to Aerial cuPHY
- TensorRT inference engine
  - Jupyter notebook example using pyAerial to validate a neural PUSCH receiver
- LDPC API improvements
  - Added soft outputs to LDPC decoder
- LS channel estimation
- Limited support for Grace Hopper
  - Run pyAerial together with Aerial Data Lakes
The NVIDIA cuBB SDK provides GPU accelerated 5G signal processing pipeline including cuPHY for Layer 1 PHY, cuMAC for L2 scheduler, delivering unprecedented throughput and efficiency by keeping all the processing within the high-performance GPU memory.

Aerial cuBB is a software-defined, scalable, modular, highly programmable and cloud-native, without any fixed function accelerators. Enables the ecosystem to flexibly adopt necessary modules for their commercial products.

Aerial cuBB has the following key components:

- **cuPHY**: L1 library of the Aerial CUDA-Accelerated RAN. It is designed as an inline accelerator to run on NVIDIA GPUs and it does not require any additional hardware accelerator.

- **cuMAC**: L2 MAC Scheduler library of the Aerial CUDA-Accelerated RAN for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs.

### 1.1. cuBB Installation Guide

This section describes how to install the Aerial cuBB.

**Important Terms**
## Term or Abbreviation | Definition
--- | ---
Aerial | SDK that accelerates 5G RAN functions with NVIDIA GPUs
cuBB | CUDA GPU software libraries/tools that accelerate 5G RAN compute-intensive processing
cuPHY | CUDA 5G PHY layer software library for the cuBB
cuPHY-CP | cuPHY control-plane software
cuMAC | CUDA-based platform for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs
HDF5 | A data file format used for storing test vectors. The HDF5 software library provides the functions for reading and writing the test vectors.
CMake | A software tool for configuring the makefiles for building the CUDA examples (see [https://cmake.org/](https://cmake.org/))
DPDK | Data Plane Development Kit
CX6-DX | Mellanox ConnectX6-DX NIC

### 1.1.1. Installing Tools on Grace Hopper MGX System

This chapter describes how to install the required kernel, driver, and tools on the host. This is a one-time installation and can be skipped if the system has been configured already.

- In the following sequence of steps, the target host is *Supermicro Grace Hopper MGX System*.
- Depending on the release, tools that are installed in this section may need to be upgraded in the *Installing and Upgrading Aerial cuBB* section.
- After everything is installed and updated, refer to the *cuBB Quick Start Guide* for how to use Aerial cuBB.
1.1.1.1 Supermicro Grace Hopper MGX Configuration

Supermicro Server SKU: ARS-111GL-NHR (Config 2)

CPU Module | NVIDIA Grace Hopper Supership, CG1
---|---
BIOS | SMC
BMC | SMC
Chassis length (mm) | 900
Chassis width | 19"
Chassis form factor | 1U
Cooling | Air
M.2 1 | 960GB
M.2 2 | TBD
Cabling | Hot
Left short slot 0 | 1x 815 4TB
Left short slot 1
Center short slot 0
Center short slot 1
Right short slot 0 | 1x 815 4TB
Right short slot 1
Left long slot 0 | BF3 83220
900-09386-00CV-AA0
Left long slot 1
Center slot 0 | BF3 83220
900-09386-00CV-AA0
Center slot 1 | IO board
Right long slot 0 | 2x 1600W CRPS
Right long slot 1
Power | PSU
Storage bay

Top View:
1.1.1.2 Cable Connection

1.1.1.2.1 Host OS Internet Connection

The BF3 NICs are reserved for fronthaul and backhaul connections, a USB to Ethernet dongle to the back USB port is recommended for the host OS internet connection.

1.1.1.2.2 E2E Test Connection

To run end-to-end test with O-RU, the BF3 fronthaul port#0 or port#1 must be connected to the fronthaul switch. Make sure the PTP is configured to use the port connected to the fronthaul switch. The following diagram shows a typical E2E connection in O-RAN LLS-C3 topology.
1.1.1.2.3 cuBB Test Connection

To run cuBB end-to-end test with TestMAC and RU emulator, a R750 RU emulator is recommended to pair with the Grace Hopper MGX system. The BF3 NIC (P/N: 900-9D3B6-00CV-AA0) should be installed on the slot 7 of the R750 server as the picture shown below.

To provision the R750 RU emulator, follow the instructions at Installing Tools on Dell R750. Because the R750 RU emulator has no GPU, the Installing CUDA Driver can be skipped. Note that the PCI addresses of the BF3 ports are ca:00.0 and ca:00.1 on the R750 RU emulator.

```bash
$ lshw -c network -businfo
```

<table>
<thead>
<tr>
<th>Bus info</th>
<th>Device</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pci@0000:04:00.0 eno8303</td>
<td>network</td>
<td>NetXtreme BCM5720 Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>pci@0000:04:00.1 eno8403</td>
<td>network</td>
<td>NetXtreme BCM5720 Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>pci@0000:ca:00.0 aerial00</td>
<td>network</td>
<td>MT43244 BlueField-3 integrated Co</td>
<td></td>
</tr>
<tr>
<td>pci@0000:ca:00.1 aerial01</td>
<td>network</td>
<td>MT43244 BlueField-3 integrated Co</td>
<td></td>
</tr>
</tbody>
</table>

The Mellanox 200GbE direct attach copper cable is required to connect the Grace Hopper MGX and R750 RU emulator to run more than 10 cells. The 100GbE direct attach copper cable should be able to support 10C 59c BFP9 but it is not going to work for 20C 60c BFP9.

To run RU emulator on R750 + BF3, update the RU emulator yaml as below:
# For RU Emulator on R750 system

```
sed -i "s∕ul_core_list.*∕ul_core_list: [5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,→37,39,41,43]∕" $RU_YAML
```

```
sed -i "s∕dl_core_list.*∕dl_core_list: [4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,→36,38,40,42]∕" $RU_YAML
```

```
sed -i "s∕aerial_fh_split_rx_tx_mempool.*∕aerial_fh_split_rx_tx_mempool: 1∕" $RU_YAML
```

```
sed -i "s∕low_priority_core.*∕low_priority_core: 45∕" $RU_YAML
```

1.1.1.3 System Firmware Upgrade

During the first boot, login to BMC to check the firmware inventory. Go to **Dashboard** -> **Maintenance** -> **Firmware Management** -> **Inventory** to see the current firmware versions.

![Firmware Management Dashboard](image)

Here is the list of the minimum required versions. Upgrade the firmware to the following or newer versions, if your system has older firmware.
<table>
<thead>
<tr>
<th>Component</th>
<th>Firmware Version</th>
<th>Firmware filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC</td>
<td>1.02.01 (20231103)</td>
<td>BMC_SCMAST2600-ROT20-2501MS_20231103_01.02.01_STDsp.bin</td>
</tr>
<tr>
<td>BIOS</td>
<td>1.0 (20231026)</td>
<td>BIOS_G1SMH-G-1D31_20231026_1.0_STDsp.bin</td>
</tr>
<tr>
<td>FPGA</td>
<td>0.8A</td>
<td>FPGA_MBD-G1SMH-G-10XX1D31_20231018_00.8A.XX_STDsp.bin</td>
</tr>
<tr>
<td>VBIOS</td>
<td>96.00.84.00.02</td>
<td>g530_0206_888__9600840002-prod.fwpkg</td>
</tr>
<tr>
<td>EROT</td>
<td>1.03.0114.0000-n01</td>
<td>cec1736-ecfw-01.03.0114.0000-n01-rel-prod.fwpkg</td>
</tr>
<tr>
<td>CPLD Motherboard Misc</td>
<td>V0B</td>
<td>CPLD_XO3-GP03E0-10XX03E0_20231020_0B.XX_STDsp.jed</td>
</tr>
</tbody>
</table>

The recommended firmware update sequence is:

1. Power off host
2. Update BMC
3. Update CPLD motherboard misc
4. Update CPU EROt
5. Update FPGA
6. A/C power cycle
7. Update BIOS
8. Update VBIOS
9. Reboot or Power cycle

To update the firmware for a specific component, go to Dashboard -> Maintenance -> Firmware Management -> Update then select the component icon -> Next -> Select File -> Upload -> Update. For example, select BMC and its firmware file as follows:
For non-BMC firmware update, it is queued in the task list to update in next boot.
1.1.1.4 Install Ubuntu 22.04 Server

Download the Ubuntu server 22.04 ISO image for ARM-based system from https://ubuntu.com/download/server/arm. Before installing the system OS, prepare a bootable USB drive contains the OS image or configure the virtual media in the BMC for remote installation. Also verify that a USB to Ethernet dongle is connected to the back USB port for host internet access.

There are two ways to configure the virtual media. One is to share the OS ISO image by Windows network sharing or Samba sharing on Linux. Then go to BMC Dashboard -> Configuration -> Virtual Media to enter the virtual media connection info including the share host ip, image path, username and password. After the connection info is saved, click the Link icon to connect.

Another way to configure virtual media, is to select the Virtual Media icon from the remote console then mount the OS ISO image to the virtual CD/DVD drive.
Reboot the system after the virtual media is configured and connected. Press **F11** to enter the BIOS boot menu and select **UEFI: USB CD/DVD Drive** to boot with the virtual media.
Launch the SOL console from the BMC Remote Control menu. The SOL console is required to complete the Ubuntu OS installation.

**Note:** The Ubuntu 22.04.3 installation media does not include a required patch for the resolution of an issue with the `ast` driver. The `ast` driver is used to interface with the BMC. The absence of this patch causes distorted output from the on-board display port and remote console. Because of this, the OS installation must be done on the SOL console. The fix is included in the NVIDIA optimized Ubuntu kernel. After installing the NVIDIA optimized Ubuntu kernel, the output of the on-board display and the remote console from BMC will be normal again.

After seeing the GRUB menu from the SOL console, select **Ubuntu Server with the HWE Kernel** to install the Ubuntu server OS.
Follow the Ubuntu installation process with the notable selection below:

- **Continue in rich mode**
- **Continue without updating**
- **Ubuntu Server**
- **Install OpenSSH server**

When the installation is done, the console shows **Install complete** and **Reboot now**. Reboot the system and check the following:

- Check if the system time is correct to avoid apt update error.

Run the following commands to set the date and time via NTP once (this will not enable the NTP service):

```bash
sudo apt-get install ntpdate
sudo ntpdate -s pool.ntp.org
```

- Check if the GPU and NIC are detected by the OS.

Use the following commands to determine whether the GPU and NIC are detected by the OS:

```
$ lspci | grep -i nvidia
# GH200 GPU
0009:01:00.0 3D controller: NVIDIA Corporation Device 2342 (rev a1)

$ lspci | grep -i mellanox
# The first BF3 NIC (Fronthaul NIC)
0000:01:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
   - integrated ConnectX-7 network controller (rev 01)
0000:01:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
   - integrated ConnectX-7 network controller (rev 01)
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

0000:01:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3 SoC Management
    Interface (rev 01)
# The second BF3 NIC (Backhaul NIC)
0002:01:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
    integrated ConnectX-7 network controller (rev 01)
0002:01:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
    integrated ConnectX-7 network controller (rev 01)
0002:01:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3 SoC Management
    Interface (rev 01)

Use the following command to change the hostname:

$ sudo hostnamectl set-hostname NEW_HOSTNAME

To display the GRUB menu during boot, create /etc/default/grub.d/menu.cfg with the following content:

$ cat <<EOF | sudo tee /etc/default/grub.d/menu.cfg
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=5
GRUB_TERMINAL="console serial"
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_SERIAL_COMMAND="GRUB_SERIAL_COMMAND serial --unit=0 --speed=115200 --word=8 --
    --parity=no --stop=1"
EOF

1.1.1.5 Configure the Network Interfaces

The following installation steps need an Internet connection. Ensure that you have the proper netplan config for your local network.

The network interface names could change after reboot. To ensure persistent network interface names after reboot, create a persistent net link files under /etc/systemd/network, one for each interface.

To find the MAC address of the BlueField-3 NIC, run lshw to check for network devices and look for the ConnectX-7 entries.

$ sudo apt-get install jq -y
$ sudo lshw -json -C network | jq '.[].product, MAC: \.serial' | grep "ConnectX-7"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:ww:ww:ww"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:yy:yy:yy"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:zz:zz:zz"

Create files at /etc/systemd/network/ with the desired name for the interface and the MAC address found in the previous step.

Note: The rest of the document will assume the aerial00 and aerial01 interfaces are the ones connected to the RU emulator for the cuBB testing or the fronthaul switch for the E2E tests and that aerial00 is the interface used for PTP.
$ sudo nano /etc/systemd/network/20-aerial00.link

[Match]
MACAddress=94:6d:ae:ww:ww

[Link]
Name=aerial00

$ sudo nano /etc/systemd/network/20-aerial01.link

[Match]
MACAddress=94:6d:ae:xx:xx:xx

[Link]
Name=aerial01

$ sudo nano /etc/systemd/network/20-aerial02.link

[Match]
MACAddress=94:6d:ae:yy:yy:yy

[Link]
Name=aerial02

$ sudo nano /etc/systemd/network/20-aerial03.link

[Match]
MACAddress=94:6d:ae:zz:zz:zz

[Link]
Name=aerial03

To apply the change:

$ sudo netplan apply

1.1.1.6 Disable Auto Upgrade

Edit the `/etc/apt/apt.conf.d/20auto-upgrades` system file, and change the "1" to "0" for both lines. This prevents the installed version of the low latency kernel from being accidentally changed with a subsequent software upgrade.

$ sudo nano /etc/apt/apt.conf.d/20auto-upgrades

APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
1.1.1.7 Install NVIDIA Optimized Ubuntu Kernel

Run the following commands to install the NVIDIA optimized Ubuntu kernel.

```bash
$ sudo apt update
# NOTE: This will install the specific kernel version, not the latest NVIDIA optimized kernel.
$ sudo apt install -y linux-image-6.5.0-1019-nvidia-64k
```

Then, update the GRUB to change the default boot kernel. The version to use here depends on the latest version that was installed with the previous command:

```bash
# Update grub to change the default boot kernel
$ sudo sed -i 's/^GRUB_DEFAULT=.*//GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 6.5.0-1019-nvidia-64k"/ /etc/default/grub
```

1.1.1.8 Configure Linux Kernel Command-line

Ensure the `iommu.passthrough=y` kernel parameter is NOT passed to the kernel. This parameter prevents the GPU driver from loading so it must be removed if it is present.

Verify that the parameter is present by running:

```bash
$ grep iommu.passthrough=y /proc/cmdline
```

If the parameter is present, find the file that contains this parameter and remove it. For example:

```bash
$ grep -rns iommu.passthrough /etc/default/grub*
# Remove iommu.passthrough=y from the found file
$ sudo sed -i ' s/ iommu.passthrough=y//i' /etc/default/<found file>
```

To set kernel command-line parameters, edit the `GRUB_CMDLINE_LINUX` parameter in the grub file `/etc/default/grub.d/cmdline.cfg` and append or update the parameters described below. The following kernel parameters are optimized for GH200. To automatically append the grub file with these parameters, enter this command:

```bash
$ cat <<"EOF" | sudo tee /etc/default/grub.d/cmdline.cfg
GRUB_CMDLINE_LINUX="$GRUB_CMDLINE_LINUX pci=realloc=off pci=pcie_bus_safe default
--hugepagesz=512M hugepagesz=512M hugepages=48 tsc=reliable processor.max_cstate=0
--audit=0 idle=poll rcu_noob_poll nosoftlockup irqaffinity=0 isolcpus=managed_irq,
--domain,4-64 nohz_full=4-64 rcu_noobcs=4-64 earlycon module_blacklist=nouveau acpi_
--power_meter.force_cap_on=y numa_balancing=disable init_on_alloc=0 preempt=none"
EOF
```

**Note:** The hugepage size is 512MB which is optimized for the 64k page size kernel on ARM.
1.1.1.9 Apply the Changes and Reboot to Load the Kernel

$ sudo update-grub
$ sudo reboot

After rebooting, enter this command to verify that the kernel command-line parameters are configured properly:

$ uname -r
6.5.0-1019-nvidia-64k

$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-6.5.0-1019-nvidia-64k root=/devmapper/ubuntu--vg-ubuntu--lv ro
__pci=realloc=off pcie_bus_safe default_hugepagesz=512M hugepagesz=512M
__hugepages=32 tsc=reliable processor.max_cstate=0 audit=0 idle=poll rcu_nocb_poll
__nosoftlockup irqaffinity=0 isolcpus=managed_irq,domain,4-47 nohz_full=4-47 rcu_
__nocbs=4-47 earlycon module_blacklist=nouveau acpi_power_meter.force_cap_on=y numa_
__balancing=disable init_on_alloc=0 preempt=none

Enter this command to check if hugepages are enabled:

$ grep -i huge /proc/meminfo
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
FileHugePages: 0 kB
HugePages_Total: 32
HugePages_Free: 32
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 524288 kB
Hugetlb: 16777216 kB

1.1.1.10 Install Dependency Packages

Enter these commands to install the prerequisite packages:

$ sudo apt-get update
$ sudo apt-get install -y build-essential linux-headers-$(uname -r) dkms unzip
__linuxptp pv apt-utils net-tools

1.1.1.11 Install DOCA OFED and Mellanox Firmware Tools on the Host

Check if there is an existing MOFED installed on the host system.

$ ofed_info -s
OFED-internal-23.10-1.1.9:

Uninstall MOFED if it is present.

$ sudo /usr/sbin/ofed_uninstall.sh

Download the doca-host_2.7.0-204000-24.04-ubuntu2204_arm64.deb package here and copy it to the local file system on the server.

Enter the following commands to install DOCA OFED.
# Install DOCA OFED
$ sudo dpkg -i doca-host_2.7.0-204000-24.04-ubuntu2204_arm64.deb
$ sudo apt update
$ sudo apt install -y doca-ofed

# To check what version of OFED you have installed
$ ofed_info -s
OFED-internal-24.04-0.6.6:

Enter the following commands to install Mellanox firmware tools.

# Install Mellanox Firmware Tools
$ export MFT_VERSION=4.28.0-92
$ tar xvf mft-$MFT_VERSION-arm64-deb.tgz
$ sudo mft-$MFT_VERSION-arm64-deb/install.sh

$ sudo mst version
mst, mft 4.28.0-92, built on Apr 25 2024, 15:22:48. Git SHA Hash: N/A

$ sudo mst start

# check NIC PCIe bus addresses and network interface names
$ sudo mst status -v
MST modules:
--------------
  MST PCI module is not loaded
  MST PCI configuration module loaded
PCI devices:
------------

Enter these commands to check the link status of port 0:

# Here is an example if the port 0 of fronthaul NIC is connected to another server or
  switch via a 200GbE DAC cable.
$ sudo mlxlink -d 0000:01:00.0

Operational Info
-----------------
State : Active
Physical state : LinkUp
Speed : 200G
Width : 4x
FEC : Standard_RS-FEC - (544,514)
Loopback Mode : No Loopback
Auto Negotiation : ON

(continues on next page)
Supported Info
--------------

Enabled Link Speed (Ext.) : 0x00003ff2 (200G_2X, 200G_4X, 100G_1X, 100G_2X, 100G_4X, 50G_1X, 50G_2X, 40G, 25G, 10G, 1G)

Supported Cable Speed (Ext.) : 0x000017f2 (200G_4X, 100G_2X, 100G_4X, 50G_1X, 50G_2X, 40G, 25G, 10G, 1G)

Troubleshooting Info
---------------------

Status Opcode : 0
Group Opcode : N/A
Recommendation : No issue was observed

Tool Information
----------------

Firmware Version : 32.39.2048
amBER Version : 2.22

1.1.1.12 Install CUDA Driver

If the system has an older driver installed, unload the current driver modules and uninstall the old driver, using the following:

```
# Unload the current driver modules
$ for m in $(lsmod | awk "/^\[^[:space:]]*\(nvidia|nv_|gdrdrv\)/ {print \$1}"); do echo Unload $m...; sudo rmmod $m; done

# Remove the driver if it was installed by runfile installer before.
$ sudo /usr/bin/nvidia-uninstall
```

Create the driver module config with the following recommended settings:

```
$ cat <<EOF | sudo tee /etc/modprobe.d/nvidia.conf
options nvidia NVreg_RegistryDwords="RMNvLinkDisableLinks=0x3FFFF;"
EOF
```

Run the following commands to install the NVIDIA open-source GPU kernel driver (OpenRM).

```
# Install NVIDIA GPU driver 555.42.02 to run Aerial L1 in non-MIG mode.
$ wget https://us.download.nvidia.com/XFree86/aarch64/555.42.02/NVIDIA-Linux-aarch64-555.42.02.run
$ sudo sh NVIDIA-Linux-aarch64-555.42.02.run --silent -m kernel-open

# Install NVIDIA GPU driver 550.54.15 to run Aerial L1 in MIG mode.
$ wget https://us.download.nvidia.com/tesla/550.54.15/NVIDIA-Linux-aarch64-550.54.15.run
$ sudo sh NVIDIA-Linux-aarch64-550.54.15.run --silent -m kernel-open

# Verify that the driver is loaded successfully
$ nvidia-smi
```

(continues on next page)
1.1.1.13 Install GDRCopy Driver

Run the following commands to install the GDRCopy driver. If the system has an older version installed, remove the old driver first.

**Warning:** GDRCopy driver must be installed after the CUDA driver.

```
# Check the installed GDRCopy driver version
$ apt list --installed | grep gdrdrv-dkms

# Remove the driver, if you have the older version installed.
$ sudo apt purge gdrdrv-dkms
$ sudo apt autoremove

# Install GDRCopy driver
$ wget https://developer.download.nvidia.com/compute/redist/gdrcopy/CUDA%2012.2/...ubuntu22_04/aarch64/gdrdrv-dkms_2.4-1_arm64.Ubuntu22_04.deb
$ sudo dpkg -i gdrdrv-dkms_2.4-1_arm64.Ubuntu22_04.deb
```
1.1.1.14 Install Docker CE

The full official instructions for installing Docker CE can be found here: https://docs.docker.com/engine/install/ubuntu/#install-docker-engine. The following instructions are one supported way of installing Docker CE:

**Warning:** To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

```bash
$ sudo apt-get update
$ sudo apt-get install -y ca-certificates curl gnupg
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu "$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
$ sudo docker run hello-world
```

1.1.1.15 Install the Nvidia Container Toolkit

Locate and follow the nvidia-container-toolkit install instructions.

Or use the following instructions as an alternate way to install the nvidia-container-toolkit. Version 1.14.1-1 is supported.

**Warning:** To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

```bash
&&
$ sudo apt-get update

$ sudo apt-get install -y nvidia-container-toolkit
$ sudo nvidia-ctk runtime configure --runtime=docker
$ sudo systemctl restart docker
$ sudo docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi
```
1.1.1.16 Update BF3 BFB Image and NIC Firmware

Note:

▶ The following instructions are for BF3 NIC (OPN: 900-9D3B6-00CV-A; PSID: MT_0000000884) specifically.
▶ There is no need to switch to DPU mode if using the BFB image below.
▶ This BFB image will update the NIC firmware automatically.

```
# Enable MST
$ sudo mst start
$ sudo mst status

MST modules:
------------------
  MST PCI module is not loaded
  MST PCI configuration module loaded

MST devices:
-------------
/dev/mst/mt41692_pciconf0 - PCI configuration cycles access.
  domain:bus:dev.fn = 0000:01:00.0 addr.reg=88 data.
    → reg=92 cr_bar.gw_offset=-1
/dev/mst/mt41692_pciconf1 - PCI configuration cycles access.
  domain:bus:dev.fn = 0002:01:00.0 addr.reg=88 data.
    → reg=92 cr_bar.gw_offset=-1
    Chip revision is: 01

# Download the BF3 BFB image

# Update the BFB image of the 1st BF3
$ sudo bfb-install -r rshim0 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb

# Update the BFB image of the 2nd BF3
$ sudo bfb-install -r rshim1 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb

Pushing bfb
1.41GiB 0:01:24 [17.1MiB/s] [=>]
Collecting BlueField booting status. Press Ctrl+C to stop...
INFO[PSC]: PSC BL1 START
INFO[BL2]: start
INFO[BL2]: boot mode (rshim)
INFO[BL2]: VDDQ adjustment complete
INFO[BL2]: VDDQ: 1120 mV
INFO[BL2]: DDR POST passed
INFO[BL2]: UEFI loaded
INFO[BL31]: start
INFO[BL31]: lifecycle GA Secured
INFO[BL31]: VDD: 851 mV
ERR[BL31]: MB timeout
INFO[BL31]: runtime
INFO[UEFI]: eMMC init

(continues on next page)
INFO: eMMC probed
INFO: UPVS valid
INFO: PMI: updates started
INFO: PMI: total updates: 1
INFO: PMI: updates completed, status 0
INFO: PCIe enum start
INFO: PCIe enum end
INFO: UEFI Secure Boot (enabled)
INFO: Redfish enabled
INFO: Partial NIC
INFO: power capping disabled
INFO: exit Boot Service
INFO: Ubuntu installation started
INFO: Installing OS image
INFO: Ubuntu installation completed
WARN: Skipping BMC components upgrade.
INFO: NIC firmware update done
INFO: Installation finished

# Wait 10 minutes to ensure the card initializes properly after the BFB installation
$ sleep 600

# NOTE: Requires a full power cycle from host with cold boot

# Verify NIC FW version after reboot
$ sudo mst start
$ sudo flint -d /dev/mst/mt41692_pciconf0 q
Image type: FS4
FW Version: 32.41.1000
FW Release Date: 28.4.2024
Product Version: 32.41.1000
Rom Info: type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
type=UEFI Virtio blk version=22.4.13 cpu=AMD64,AARCH64
type=UEFI version=14.34.12 cpu=AMD64,AARCH64
type=PXE version=3.7.400 cpu=AMD64
Description: UID GuidsNumber
Base GUID: 946dae0300f5aa8e
Base GUID: 946daef5aa8e
Image VSD: N/A
Device VSD: N/A
PSID: MT_0000000884
Security Attributes: secure-fw

Run the following commands to configure the BF3 NIC:

# Setting BF3 port to Ethernet mode (not Infiniband)
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P1=2
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P2=2

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_MODEL=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_PAGE__SUPPLIER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_ESWITCH__MANAGER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_IB_VPORT0=EXT_...HOST_PF

(continues on next page)
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_OFFLOAD ENGINE=DISABLED

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set CQE_COMPRESSION=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set PROG_PARSE_GRAPH=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set ACCURATE_TX_SCHEDULER=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set FLEX_PARSER_PROFILE_ENABLE=4
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set REAL_TIME_CLOCK_ENABLE=1

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_PXE ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_ARM ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_x86 ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_ARM ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_x86 ENABLE=0

# NOTE: Requires a full power cycle from host with cold boot
# Verify that the NIC FW changes have been applied

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\|INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\|INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"

    INTERNAL_CPU_MODEL       EMBEDDED_CPU(1)
    INTERNAL_CPU_PAGE_SUPPLIER EXT_HOST_PF(1)
    INTERNAL_CPU_ESWITCH_MANAGER EXT_HOST_PF(1)
    INTERNAL_CPU_IB_VPORT0    EXT_HOST_PF(1)
    INTERNAL_CPU_OFFLOAD_ENGINE DISABLED(1)
    FLEX_PARSER_PROFILE_ENABLE 4
    PROG_PARSE_GRAPH          True(1)
    ACCURATE_TX_SCHEDULER     True(1)
    CQE_COMPRESSION           AGGRESSIVE(1)
    REAL_TIME_CLOCK_ENABLE    True(1)
    LINK_TYPE_P1              ETH(2)
    LINK_TYPE_P2              ETH(2)

1.1.1.17 Install ptp4l and phc2sys

Enter these commands to configure PTP4L, assuming that aerial00 NIC interface and CPU core 41 are used for PTP:

$ cat <<EOF | sudo tee /etc/ptp.conf
[global]
dataset_comparison  G.8275.x
G.8275.defaultDS.localPriority 128
maxStepsRemoved 255
logAnnounceInterval -3
logSyncInterval -4
logMinDelayReqInterval -4
G.8275.portDS.localPriority 128
network_transport L2

(continued from next page)
domainNumber 24
tx_timestamp_timeout 30
slaveOnly 1
clock_servo pi
step_threshold 1.0
egressLatency 28
pi_proportional_const 4.65
pi_integral_const 0.1

[aerial00]
announceReceiptTimeout 3
delay_mechanism E2E
network_transport L2
EOF

$ cat <<EOF | sudo tee /lib/systemd/system/ptp4l.service
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l
After=network.target

[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStartPre=ifconfig aerial00 up
ExecStartPre=ethtool --set-priv-flags aerial00 tx_port_ts on
ExecStartPre=ethtool --set-priv-flags aerial00 rx off tx off
ExecStartPre=ifconfig aerial01 up
ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
ExecStartPre=ethtool --set-priv-flags aerial01 rx off tx off
ExecStart=taskset -c 41 /usr/sbin/ptp4l -f /etc/ptp.conf

[Install]
WantedBy=multi-user.target
EOF

$ sudo systemctl daemon-reload
$ sudo systemctl restart ptp4l.service
$ sudo systemctl enable ptp4l.service

One server becomes the master clock, as shown below:

$ sudo systemctl status ptp4l.service

• ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2023-08-08 19:37:56 UTC; 2 weeks 3 days ago
    Docs: man:ptp4l
    Main PID: 1120 (ptp4l)
    Tasks: 1 (limit: 94533)
    Memory: 460.0K
    CPU: 9min 8.089s
    CGroup: /system.slice/ptp4l.service
           1120 /usr/sbin/ptp4l -f /etc/ptp.conf

(continues on next page)
Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp4l[81287.043]: selected local clock b8cef6.ffe.d333be as best master
Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp4l[81287.043]: port 1: assuming the grand master role
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: timed out while polling for tx timestamp
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: increasing tx_
timestamp_timeout may correct this issue, but it is likely caused by a driver bug
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: port 1: send sync failed
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: port 1: MASTER to FAULTY on FAULT_DETECTED (FT_UNSPECIFIED)
Aug 11 20:45:07 aerial-devkit taskset[1120]: ptp4l[263239.522]: port 1: FAULTY to LISTENING on INIT_COMPLETE
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: port 1: LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: selected local clock b8cef6.ffe.d333be as best master
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: port 1: assuming the grand master role

The other becomes the secondary, follower clock, as shown below:

```sh
$ sudo systemctl status ptp4l.service

ptp4l.service - Precision Time Protocol (PTP) service
        Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
            Active: active (running) since Tue 2023-11-21 17:49:00 UTC; 5 days ago
        Docs: man:ptp4l
              (continues from previous page)
        Main PID: 203 (ptp4l)
            Tasks: 1 (limit: 146916)
            Memory: 2.7M
            CPU: 6min 16.710s
        CGroup: /system.slice/ptp4l.service
               2023 /usr/sbin/ptp4l -f /etc/ptp.conf

Nov 27 05:11:18 gh-smc-cg1-qs-01 ptp4l[2023]: [475374.166] rms 2 max 4 freq -
        13578 +/- 11 delay -33 +/-  0
Nov 27 05:11:19 gh-smc-cg1-qs-01 ptp4l[2023]: [475375.166] rms 2 max 5 freq -
        13578 +/- 10 delay -34 +/-  0
Nov 27 05:11:20 gh-smc-cg1-qs-01 ptp4l[2023]: [475376.166] rms 4 max 8 freq -
        13584 +/- 17 delay -34 +/-  1
Nov 27 05:11:21 gh-smc-cg1-qs-01 ptp4l[2023]: [475377.166] rms 4 max 8 freq -
        13586 +/- 20 delay -35 +/-  1
Nov 27 05:11:22 gh-smc-cg1-qs-01 ptp4l[2023]: [475378.166] rms 4 max 7 freq -
        13588 +/- 18 delay -33 +/-  1
Nov 27 05:11:23 gh-smc-cg1-qs-01 ptp4l[2023]: [475379.166] rms 4 max 9 freq -
        13579 +/- 28 delay -34 +/-  1
Nov 27 05:11:24 gh-smc-cg1-qs-01 ptp4l[2023]: [475380.167] rms 3 max 7 freq -
        13584 +/- 14 delay -34 +/-  1
```

(continues on next page)
Enter the commands to turn off NTP:

```bash
$ sudo timedatectl set-ntp false
$ timedatectl
```

Run PHC2SYS as service:

PHC2SYS is used to synchronize the system clock to the PTP hardware clock (PHC) on the NIC.

Specify the network interface used for PTP and system clock as the slave clock.

```bash
# If more than one instance is already running, kill the existing PHC2SYS sessions.

# Command used can be found in /lib/systemd/system/phc2sys.service
# Update the ExecStart line to the following
$ cat <<EOF | sudo tee /lib/systemd/system/phc2sys.service
[Unit]
Description=Synchronize system clock or PTP hardware clock (PHC)
Documentation=man:phc2sys
Requires=ptp4l.service
After=ptp4l.service

[Service]
Restart=always
RestartSec=5s
Type=simple
# Gives ptp4l a chance to stabilize
ExecStartPre=sleep 2
ExecStart=/bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp4l$(ethtool -T aerial00 | grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -0 0 -R 256 -u 256"

[Install]
WantedBy=multi-user.target
EOF
```

After the PHC2SYS config file is changed, run the following:

```bash
$ sudo systemctl daemon-reload
$ sudo systemctl restart phc2sys.service
# Set to start automatically on reboot
```
$ sudo systemctl enable phc2sys.service

# check that the service is active and has converged to a low rms value (<30) and that the correct NIC has been selected (aerial00):
$ sudo systemctl status phc2sys.service

phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset: enabled)
Active: active (running) since Tue 2023-11-21 17:49:02 UTC; 5 days ago
   Docs: man:phc2sys
Process: 2037 ExecStartPre=sleep 2 (code=exited, status=0/SUCCESS)
Main PID: 2102 (sh)
Tasks: 2 (limit: 146916)
Memory: 1.9M CPU: 25min 30.299s
CGroup: /system.slice/phc2sys.service

   2102 /bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -T aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u 256"

   2108 /usr/sbin/phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u 256

Verify that the system clock is synchronized:

$ timedatectl

Local time: Mon 2023-11-27 06:02:44 UTC
Universal time: Mon 2023-11-27 06:02:44 UTC
RTC time: Mon 2023-11-27 06:02:44
Time zone: Etc/UTC (UTC, +0000)

System clock synchronized: yes
NTP service: inactive
RTC in local TZ: no
1.1.1.18 Setup the Boot Configuration Service

Create the directory /usr/local/bin and create the /usr/local/bin/nvidia.sh file to run the commands with every reboot.

**Note:** The command for “nvidia-smi lgc” expects just one GPU device (-i 0). This needs to be modified if the system uses more than one GPU. The mode must be set to 1 for the GH200 so that it can utilize the max clock rate, otherwise it is limited to 1830MHz with the default mode=0.

```
$ cat <<"EOF" | sudo tee /usr/local/bin/nvidia.sh
#!/bin/bash
mst start
nvidia-smi -i 0 -lgc $(nvidia-smi -i 0 --query-supported-clocks=graphics --format=csv, --noheader,nounits | sort -h | tail -n 1) --mode=1
nvidia-smi -mig 0
echo -1 > /proc/sys/kernel/sched_rt_runtime_us
EOF
```

Create a system service file to be loaded after network interfaces are up.

```
$ cat <<EOF | sudo tee /lib/systemd/system/nvidia.service
[Unit]
After=network.target
[Service]
ExecStart=/usr/local/bin/nvidia.sh
[Install]
WantedBy=default.target
EOF
```

Create a system service file for nvidia-persistenced to be run at startup.

**Note:** This file was created following the sample from /usr/share/doc/NVIDIA_GLX-1.0/samples/nvidia-persistenced-init.tar.bz2

```
cat <<EOF | sudo tee /lib/systemd/system/nvidia-persistenced.service
[Unit]
Description=NVIDIA Persistence Daemon
Wants=syslog.target
[Service]
Type=forking
ExecStart=/usr/bin/nvidia-persistenced
ExecStopPost=/bin/rm -rf /var/run/nvidia-persistenced
[Install]
WantedBy=multi-user.target
EOF
```

Then set the file permissions, reload the systemd daemon, enable the service, restart the service when installing the first time, and check status.
The output of the last command should look like this:

```bash
aerial@server:~$ sudo systemctl status nvidia.service
nvidia.service
   Loaded: loaded (/lib/systemd/system/nvidia.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2024-06-07 20:11:55 UTC; 2s ago
      Process: 3300619 ExecStart=/usr/local/bin/nvidia.sh (code=exited, status=0/SUCCESS)
   Main PID: 3300619 (code=exited, status=0/SUCCESS)
   CPU: 1.091s
Jun 07 20:11:54 server nvidia.sh[3300620]: Loading MST PCI module - Success
Jun 07 20:11:54 server nvidia.sh[3300620]: [warn] mst_pciconf is already loaded, skipping
Jun 07 20:11:55 server nvidia.sh[3300620]: Unloading MST PCI module (unused) - Success
Jun 07 20:11:55 server nvidia.sh[3302599]: GPU clocks set to "(gpuClkMin 1980, gpuClkMax 1980)" for GPU 00000009:01:00.0
Jun 07 20:11:55 server nvidia.sh[3302599]: All done.
Jun 07 20:11:55 server nvidia.sh[3302599]: All done.
Jun 07 20:11:55 server systemd[1]: nvidia.service: Consumed 1.091s CPU time.

aerial@server:~$ sudo systemctl status nvidia-persistenced.service
nvidia-persistenced.service - NVIDIA Persistence Daemon
   Loaded: loaded (/lib/systemd/system/nvidia-persistenced.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-06-05 21:42:17 UTC; 1 day 22h ago
      Main PID: 1858 (nvidia-persistenced)
       Tasks: 1 (limit: 146899)
      Memory: 36.5M
      CPU: 2.353s
    CGroup: /system.slice/nvidia-persistenced.service
               1858 /usr/bin/nvidia-persistenced
Jun 05 21:42:15 server systemd[1]: Starting NVIDIA Persistence Daemon...
Jun 05 21:42:15 server nvidia-persistenced[1858]: Started (1858)
```
1.1.1.19 Running Aerial on Grace Hopper

The default MGX CG1 configs within the Aerial source are:

- cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml
- cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08_CG1.yaml

Pass F08_CG1 to the cuphycontroller_scf executable to select them.

1.1.2. Installing Tools on Dell R750

This chapter describes how to install the required kernel, driver, and tools on the host. This is a one-time installation and can be skipped if the system has been configured already.

- In the following sequence of steps, the target host is Dell PowerEdge R750.
- Depending on the release, tools that are installed in this section may need to be upgraded in the Installing and Upgrading Aerial cuBB section.
- After everything is installed and updated, refer to the cuBB Quick Start Guide on how to use Aerial cuBB.

1.1.2.1 Dell PowerEdge R750 Server Configuration

1. Dual Intel Xeon Gold 6336Y CPU @ 2.4G, 24C/48T (185W)
2. 512GB RDIMM, 3200MT/s
3. 1.92TB, Enterprise NVMe
4. Riser Config 2, Full Length, 4x16, 2x8 slots (PCIe gen 4)
5. Dual, Hot-Plug Power Supply Redundant (1+1), 1400W or 2400W
6. GPU Enablement
7. NVIDIA Converged Accelerator: A100X

1.1.2.2 Converged Accelerator Installation

R750 supports PCIe 4.0 x16 at slot 2,3,6,7 and x8 at slot 4,5. Follow the table below to install single or dual converged accelerator in the assigned slot and ensure the GPU power cable is connected properly. These are the GPU installation instructions from Dell R750 Installation Manual.

**NOTE:** Only use SIG_PWR_3 and SIG_PWR_4 connectors on the motherboard for GPU power.

<table>
<thead>
<tr>
<th>GPU</th>
<th>Slot</th>
<th>GPU Power</th>
<th>NUMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU#1</td>
<td>7 (Riser 4)</td>
<td>SIG_PWR_3</td>
<td>1</td>
</tr>
<tr>
<td>GPU#2</td>
<td>2 (Riser 1)</td>
<td>SIG_PWR_4</td>
<td>0</td>
</tr>
</tbody>
</table>

Rear View:
1.1.2.3 Cable Connection

1. To run end-to-end test with O-RU, the converged accelerator port#0 or port#1 must be connected to the fronthaul switch. Make sure the PTP is configured to use the port connected to the fronthaul switch.

2. To run cuBB end-to-end test with TestMAC and RU emulator, an Aerial Devkit is required to run RU emulator. The converged accelerator port#1 on R750 must be connected to CX6-DX NIC port#0 on Aerial Devkit (RU emulator server) via Mellanox 100GbE direct attach copper cable.
1.1.2.4 Configure BIOS Settings

During the first boot, change the BIOS settings in the following order. The same settings can be changed via BMC: Configuration ▶ BIOS Settings.

**Integrated Devices**: Enable Memory Mapped I/O above 4GB and change Memory Mapped I/O Base to 12TB.

**System Profile Settings**: Change System Profile to *Performance* and Workload Profile to *Low Latency Optimized Profile*. 
Processor Settings: Aerial CUDA-Accelerated RAN supports both HyperThreaded mode (experimental) or non-HyperThreaded mode (default) but make sure the kernel command line and the CPU core affinity in the cuPHYController YAML match the BIOS settings.

To enable HyperThreading, enable the Logical Processor. To disable HyperThreading, disable the Logical Processor.

Save the BIOS settings, then reboot the system.
1.1.2.5 Install Ubuntu 22.04 Server

After installing Ubuntu 22.04 Server, verify the following:

- System time is correct to avoid apt update error. If not, see How to fix system time.
- LVM volume uses the whole disk space. If not, see How to resize LVM volume.
- GPU and NIC are detected by the OS:

  Use the following commands to determine whether the GPU and NIC are detected by the OS:

  ```
  $ lspci | grep -i nvidia
  # If the system has A100X GPU installed
  cf:00.0 3D controller: NVIDIA Corporation Device 20b8 (rev a1)
  $ lspci | grep -i mellanox
  # If the system has A100X GPU installed
  cc:00.0 Ethernet controller: Mellanox Technologies MT42822 BlueField-2
  → integrated ConnectX-6 Dx network controller (rev 01)
  cc:00.1 Ethernet controller: Mellanox Technologies MT42822 BlueField-2
  → integrated ConnectX-6 Dx network controller (rev 01)
  ```

1.1.2.6 Disable Auto Upgrade

Edit the `etc/apt/apt.conf.d/20auto-upgrades` system file, and change the "1" to "0" for both lines. This prevents the installed version of the low latency kernel from being accidentally changed with a subsequent software upgrade.

```
$ sudo nano /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

1.1.2.7 Install the Low-Latency Kernel

If the low latency kernel is not installed, you must remove the old kernels and keep only the latest generic kernel. Enter the following command to list the installed kernels:

```
$ dpkg --list | grep -i 'linux-image' | awk '/ii/ { print $2}'
# To remove old kernel
$ sudo apt-get purge linux-image-<old kernel version>
$ sudo apt-get autoremove
```

Install the low-latency kernel with the specific version listed in the release manifest.

```
$ sudo apt-get update
$ sudo apt-get install -y linux-image-5.15.0-1042-nvidia-lowlatency
```

Update the GRUB to change the default boot kernel:

```
# Update grub to change the default boot kernel
$ sudo sed -i 's/GRUB_DEFAULT=.*/GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 5.15.0-1042-nvidia-lowlatency"/ /etc/default/grub
```
1.1.2.8 Configure Linux Kernel Command-line

To set kernel command-line parameters, edit the `GRUB_CMDLINE_LINUX_DEFAULT` parameter in the GRUB file `/etc/default/grub` and append/update the parameters described below. The following kernel parameters are optimized for Xeon Gold 6336Y CPU and 512GB memory.

To automatically append the GRUB file with these changes, enter this command:

```
# When HyperThread is disabled (default)
$ sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT=\"[^\"]*\"/\& pci=realloc=off default_hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable clocksourced=tsc intel_idle.
max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_pstate=disable audit=0
 idle=poll rcu_noct_poll nosoftlockup iommu=off irqaffinity=0-3 isolcpus=managed_irq,
domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 noht numa_balancing=disable/\' /etc/default/grub
```

```
# When HyperThread is enabled (experimental)
$ sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT=\"[^\"]*\"/\& pci=realloc=off default_hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable clocksourced=tsc intel_idle.
max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_pstate=disable audit=0
 idle=poll rcu_noct_poll nosoftlockup iommu=off irqaffinity=0-3 isolcpus=managed_irq,
domain,4-95 nohz_full=4-95 rcu_nocbs=4-95 noht numa_balancing=disable/\' /etc/default/grub
```

The CPU-cores-related parameters must be adjusted depending on the number of CPU cores on the system. In the example above, the "4-47" value represents CPU core numbers 4 to 47; you may need to adjust this parameter depending on the HW configuration. By default, only one DPDK thread is used. The isolated CPUs are used by the entire cuBB software stack. Use the `nproc --all` command to see how many cores are available. Do not use core numbers that are beyond the number of available cores.

**Warning:** These instructions are specific to Ubuntu 22.04 with a 5.15 low-latency kernel provided by Canonical. Make sure the kernel commands provided here are suitable for your OS and kernel versions and revise these settings to match your system if necessary.

1.1.2.9 Apply the Changes and Reboot to Load the Kernel

```
$ sudo update-grub
$ sudo reboot
```

After rebooting, enter the following command to verify that the system has booted into the low-latency kernel:

```
$ uname -r
5.15.0-1042-nvidia-lowlatency
```

Enter this command to verify that the kernel command-line parameters are configured properly:

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.15.0-1042-nvidia-lowlatency root=/dev/mapper/ubuntu--vg-ubuntu--lv ro pci=realloc=off default_hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable
 clocksourced=tsc intel_idle.max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_
pstate=disable audit=0 idle=poll rcu_noct_poll nosoftlockup iommu=off irqaffinity=0-3
isolcpus=managed_irq,domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 noht numa_
balancing=disable
```
Enter this command to verify if hugepages are enabled:

```
$ grep -i huge /proc/meminfo
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
FileHugePages: 0 kB
HugePages_Total: 16
HugePages_Free: 16
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 1048576 kB
Hugetlb: 16777216 kB
```

1.1.2.10 Disabling Nouveau

Enter this command to disable nouveau:

```
$ cat <<EOF | sudo tee /etc/modprobe.d/blacklist-nouveau.conf
blacklist nouveau
options nouveau modeset=0
EOF
```

Regenerate the kernel initramfs and reboot the system:

```
$ sudo update-initramfs -u
$ sudo reboot
```

1.1.2.11 Install Dependency Packages

Enter these commands to install prerequisite packages:

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential linux-headers-$(uname -r) dkms unzip "linuxptp pv"
```

1.1.2.12 Install RSHIM and Mellanox Firmware Tools on the Host

**Note:**

1. Aerial has been using Mellanox inbox driver instead of MOFED since the 23-4 release. MOFED must be removed if it is installed on the system.

2. RSHIM package is shared via PID account. If you cannot access it, contact NVIDIA CPM.

Check if there is an existing MOFED installed on the host system.

```
$ ofed_info -s
MLNX_OFED_LINUX-23.07-0.5.0.0:
```

Uninstall MOFED if it is present.
Download the rshim package and copy it to the local file system on the server.

Enter the following commands to install rshim driver.

```
# Install rshim
$ sudo apt-get install libfuse2
$ sudo dpkg -i rshim_2.0.17.g0caa378_amd64.deb
```

Enter the following commands to install Mellanox firmware tools.

```
# Install Mellanox Firmware Tools
$ export MFT_VERSION=4.28.0-92
$ wget https://www.mellanox.com/downloads/MFT/mft-$MFT_VERSION-x86_64-deb.tgz
$ tar xvf mft-$MFT_VERSION-x86_64-deb.tgz
$ sudo mft-$MFT_VERSION-x86_64-deb/install.sh
```

```
# Verify the install Mellanox firmware tool version
$ sudo mst version
mst, mft 4.28.0-92, built on Apr 25 2024, 15:22:58. Git SHA Hash: N/A
```

```
# check NIC PCIe bus addresses and network interface names
$ sudo mst status -v
```

```
# Here is the result of GPU#1 on slot 7
MST modules:
---------------------
MST PCI module is not loaded
MST PCI configuration module loaded
PCI devices:
---------------------
<table>
<thead>
<tr>
<th>DEVICE_TYPE</th>
<th>MST</th>
<th>NUMA</th>
<th>PCI</th>
<th>RDMA</th>
<th>NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlueField3</td>
<td>/dev/mst/mt41692_pciconf0.1 cc:00.1 mlx5_1 net-aerial00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BlueField3</td>
<td>/dev/mst/mt41692_pciconf0  cc:00.0 mlx5_0 net-aerial01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Enter these commands to check the link status of port 0:

```
# Here is an example if port 0 is connected to another server via a 100GbE DAC cable.
$ sudo mlxlink -d cc:00.0
```

Operation Info
---------------------
State : Active
Physical state : LinkUp
Speed : 100G
Width : 4x
FEC : Standard RS-FEC - RS(528,514)
Loopback Mode : No Loopback
Auto Negotiation : ON

Supported Info
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

--------------

Enabled Link Speed (Ext.) : 0x00003ff2 (200G_2X, 200G_4X, 100G_1X, 100G_2X, 100G_4X, 50G_1X, 50G_2X, 40G, 25G, 10G, 1G)

Supported Cable Speed (Ext.) : 0x000002f2 (100G_4X, 50G_2X, 40G, 25G, 10G, 1G)

Troubleshooting Info
-------------------

Status Opcode : 0
Group Opcode : N/A
Recommendation : No issue was observed

Tool Information
----------------

Firmware Version : 32.39.2048
amBER Version : 3.2
MFT Version : mft 4.28.0-92

### 1.1.2.13 Install the CUDA Driver

**Note:** Aerial has been using the open-source GPU kernel driver (OpenRM) since the 23-4 release.

If the system has older driver installed, you must unload the current driver modules and uninstall the old driver.

```bash
# Unload the current driver modules
$ for m in $(lsmod | awk "/^\([^[:space:]]*nvidia|nv_|gdrdrv/\) {print \$1}"); do echo
  Unload $m...; sudo rmmod $m; done

# Remove the driver if it was installed by runfile installer before.
$ sudo /usr/bin/nvidia-uninstall
```

Run the following commands to install the **NVIDIA open-source GPU kernel driver** (OpenRM).

```bash
# Install CUDA driver
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/555.42.02/NVIDIA-Linux-x86_64-555.42.02.run
$ sudo sh NVIDIA-Linux-x86_64-555.42.02.run --silent -m kernel-open

# Verify that the driver is loaded successfully
$ nvidia-smi
```

<table>
<thead>
<tr>
<th>NVIDIA-SMI 555.42.02</th>
<th>Driver Version: 555.42.02</th>
<th>CUDA Version: 12.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Name</td>
<td>Persistence-M</td>
<td>Bus-Id</td>
</tr>
<tr>
<td>ECC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fan Temp Perf</td>
<td>Pwr:Usage/Cap</td>
<td>Memory-Usage</td>
</tr>
<tr>
<td>Compute M.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIG M.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(continues on next page)
1.1.2.14 Install the GDRCopy Driver

Run the following commands to install the GDRCopy driver. If the system has an older version installed, you must remove the old driver.

**Warning:** GDRCopy driver must be installed after CUDA.

```bash
# Check the installed GDRCopy driver version
$ apt list --installed | grep gdrdrv-dkms

# Remove the driver if you have the older version installed.
$ sudo apt purge gdrdrv-dkms
$ sudo apt autoremove

# Install GDRCopy driver
$ wget https://developer.download.nvidia.com/compute/redist/gdrcopy/CUDA%2012.2/ubuntu22_04/x64/gdrdrv-dkms_2.4-1_amd64.Ubuntu22_04.deb
$ sudo dpkg -i gdrdrv-dkms_2.4-1_amd64.Ubuntu22_04.deb
```
1.1.2.15 Install Docker CE

The full official instructions for installing Docker CE can be found on the Docker website: https://docs.docker.com/engine/install/ubuntu/#install-docker-engine. The following instructions are one supported way of installing Docker CE:

```
Warning: To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

$ sudo apt-get update
$ sudo apt-get install -y ca-certificates curl gnupg
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu/$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
$ sudo docker run hello-world
```

1.1.2.16 Install the Nvidia Container Toolkit

Locate and follow the nvidia-container-toolkit install instructions. Or use the following instructions as an alternate way to install the nvidia-container-toolkit. Version 1.14.1-1 is supported.

```
Warning: To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

$ sudo apt-get update
$ sudo apt-get install -y nvidia-container-toolkit
$ sudo nvidia-ctk runtime configure --runtime=docker
$ sudo systemctl restart docker
$ sudo docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi
```
1.1.2.17 Update BF3 BFB Image and NIC Firmware

Note:

► The following instructions are for BF3 NIC (OPN: 900-9D3B6-00CV-A; PSID: MT_0000000884) specifically.

► There is no need to switch to DPU mode if using the BFB image below.

► This BFB image will update the NIC firmware automatically.

# Enable MST
$ sudo mst start
$ sudo mst status

MST modules:
-------------
MST PCI module is not loaded
MST PCI configuration module loaded

MST devices:
-------------
/dev/mst/mt41692_pciconf0 - PCI configuration cycles access.
   domain:bus:dev.fn = 0000:cc:00.0 addr.reg=88 data.
   → reg=92 cr_bar.gw_offset=-1
   Chip revision is: 01

# Download the BF3 BFB image

# Here is the command to flash BFB image. NOTE: If there are multiple BF3 NICs, repeat the same command with rshim<0..N-1>. N is the number of BF3 NICs.
$ sudo bfb-install -r rshim0 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb

Pushing bfb
1.41GiB 0:01:24 [17.1MiB/s] [          ]
   Collecting BlueField booting status. Press Ctrl+C to stop...
INFO[PSC]: PSC BL1 START
INFO[BL2]: start
INFO[BL2]: boot mode (rshim)
INFO[BL2]: VDDQ adjustment complete
INFO[BL2]: VDDQ: 1120 mV
INFO[BL2]: DDR POST passed
INFO[BL2]: UEFI loaded
INFO[BL31]: start
INFO[BL31]: lifecycle GA Secured
INFO[BL31]: VDD: 851 mV
ERR[BL31]: MB timeout
INFO[BL31]: runtime
INFO[UEFI]: eMMC init
INFO[UEFI]: eMMC probed
INFO[UEFI]: UPVS valid
INFO[UEFI]: PMI: updates started
INFO[UEFI]: PMI: total updates: 1
INFO[UEFI]: PMI: updates completed, status 0

(continues on next page)
INFO [UEFI]: PCIe enum start
INFO [UEFI]: PCIe enum end
INFO [UEFI]: UEFI Secure Boot (enabled)
INFO [BL31]: Partial NIC
INFO [BL31]: power capping disabled
INFO [UEFI]: exit Boot Service
INFO [MISC]: Ubuntu installation started
INFO [MISC]: Installing OS image
INFO [MISC]: Ubuntu installation completed
WARN [MISC]: Skipping BMC components upgrade.
INFO [MISC]: Updating NIC firmware...
INFO [MISC]: NIC firmware update done
INFO [MISC]: Installation finished

# Wait 10 minutes to ensure the card initializes properly after the BFB installation
$ sleep 600

# NOTE: Requires a full power cycle from host with cold boot

# Verify NIC FW version after reboot
$ sudo mst start
$ sudo flint -d /dev/mst/mt41692_pciconf0 q
Image type: FS4
FW Version: 32.41.1000
FW Release Date: 28.4.2024
Product Version: 32.41.1000
Rom Info: type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
type=UEFI Virtio blk version=22.4.13 cpu=AMD64,AARCH64
type=UEFI version=14.34.12 cpu=AMD64,AARCH64
type=PXE version=3.7.400 cpu=AMD64
Description: UID GuidsNumber
Base GUID: 946dae0300f5aa8e
Base MAC: 946daef5aa8e
Image VSD: N/A
Device VSD: N/A
PSID: MT_0000000884
Security Attributes: secure-fw

Run the following commands to configure the BF3 NIC:

# Setting BF3 port to Ethernet mode (not Infiniband)
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P1=2
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P2=2

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_MODEL=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_PAGE_-_SUPPLIER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_ESWITCH_-_MANAGER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_IB_VPORT0=EXT_-_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_OFFLOAD_-_ENGINE=DISABLED

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set CQE_COMPRESSION=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set PROG_PARSE_GRAPH=1

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set ACCURATE_TX_SCHEDULER=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set FLEX_PARSER_PROFILE_ENABLE=4
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set REAL_TIME_CLOCK_ENABLE=1

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_PXE_ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_ARM_ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_x86_ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_ARM_ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_x86_ENABLE=0

# NOTE: Requires a full power cycle from host with cold boot

# Verify that the NIC FW changes have been applied
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\|INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\|INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"

INTERNAL_CPU_MODEL = EMBEDDED_CPU(1)
INTERNAL_CPU_PAGE_SUPPLIER = EXT_HOST_PF(1)
INTERNAL_CPU_ESWITCH_MANAGER = EXT_HOST_PF(1)
INTERNAL_CPU_IB_VPORT0 = EXT_HOST_PF(1)
INTERNAL_CPU_OFFLOAD_ENGINE = DISABLED(1)
FLEX_PARSER_PROFILE_ENABLE = True(1)
PROG_PARSE_GRAPH = True(1)
ACCURATE_TX_SCHEDULER = True(1)
CQE_COMPRESSION = AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE = True(1)
LINK_TYPE_P1 = ETH(2)
LINK_TYPE_P2 = ETH(2)

1.1.2.18 Update A100X BFB Image and NIC Firmware

NOTE: The following instructions are specifically for A100X boards. Ensure RSHIM and MFT are installed on the system.

# Enable MST
$ sudo mst start
$ sudo mst status

MST modules:
-------------
MST PCI module is not loaded
MST PCI configuration module loaded

MST devices:
-------------
/dev/mst/mt41686_pciconf0 - PCI configuration cycles access.
reg=92 cr_bar.gw_offset=-1
domain:bus:dev.fn=0000:cc:00.0 addr.reg=88 data.
Chip revision is: 01

(continues on next page)
# Change to DPU mode
$ sudo mlxconfig -d /dev/mst/m41686_pciconf0 -y INTERNAL_CPU_MODEL=1 INTERNAL_CPU_OFFLOAD_ENGINE=0

# NOTE: Requires a power cycle to take effect
$ sudo reboot

# Update BFB image first
$ wget https://content.mellanox.com/BlueField/BFBs/Ubuntu22.04/DOCA_2.5.0_BSP_4.5.0_Ubuntu_22.04-1.23-10.prod.bfb
$ sudo bfb-install -r rshim0 -b DOCA_2.5.0_BSP_4.5.0_Ubuntu_22.04-1.23-10.prod.bfb

Pushing bfb
920MiB 0:01:51 [8.22MiB/s]

Collecting BlueField booting status. Press Ctrl+C to stop...
INFO [BL2]: start
INFO [BL2]: DDR POST passed
INFO [BL2]: UEFI loaded
INFO [BL31]: start
INFO [BL31]: lifecycle Secured (development)
INFO [BL31]: runtime
INFO [UEFI]: eMMC init
INFO [UEFI]: UPVS valid
INFO [UEFI]: eMMC probed
INFO [UEFI]: PMI: updates started
INFO [UEFI]: PMI: boot image update
INFO [UEFI]: PMI: updates completed, status 0
INFO [UEFI]: PCIe enum start
INFO [UEFI]: PCIe enum end
INFO [UEFI]: exit Boot Service
INFO [MISC]: Ubuntu installation started
INFO [MISC]: Installing OS image
INFO [MISC]: Installation finished

# Wait 10 minutes to ensure the card initializes properly after the BFB installation
$ sleep 600

# Update NIC firmware
$ sudo flint -d /dev/mst/m41686_pciconf0 -i fw-BlueField-2-rel-24_39_2048-699210040230_Ax-NVME-20.4.1-UEFI-21.4.13-UEFI-22.4.12-UEFI-14.32.17-FlexBoot-3.7.300.signed.bin -y b

Current FW version on flash: 24.35.1012
New FW version: 24.39.2048

FSMST_INITIALIZE - OK
Writing Boot image component - OK
Restoring signature - OK
# Change to NIC mode

$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 -y s INTERNAL_CPU_MODEL=1 INTERNAL_CPU_OFFLOAD_ENGINE=1

# NOTE: Requires a full power cycle from host with cold boot

# Verify NIC FW version after reboot

$ sudo mst start
$ sudo flint -d /dev/mst/mt41686_pciconf0 q

Image type: FS4
FW Version: 24.39.2048
FW Release Date: 29.11.2023
Product Version: 24.39.2048
Rom Info:
  type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
  type=UEFI Virtio blk version=22.4.12 cpu=AMD64,AARCH64
  type=UEFI version=14.32.17 cpu=AMD64,AARCH64
  type=PXE version=3.7.300 cpu=AMD64

Description: UID GuidsNumber
  Base GUID: 48b02d03005f770c 16
  Base MAC: 48b02d5f770c 16

Image VSD: N/A
Device VSD: N/A
PSID: NVD0000000015
Security Attributes: secure-fw

Run the following code to switch the A100x to the BF2-as-CX mode:

# Setting BF2 port to Ethernet mode (not Infiniband)

$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set LINK_TYPE_P1=2
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set LINK_TYPE_P2=2

# Setting BF2 Embedded CPU mode

$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set INTERNAL_CPU_MODEL=1
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set INTERNAL_CPU_PAGE_SUPPLIER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set INTERNAL_CPU_ESWITCH_MANAGER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set INTERNAL_CPU_IB_VPORT0=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set INTERNAL_CPU_OFFLOAD_ENGINE=DISABLED

# Accurate scheduling related settings

$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set CQE_COMPRESSION=1
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set PROG_PARSE_GRAPH=1
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set ACCURATE_TX_SCHEDULER=1
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set FLEX_PARSER_PROFILE_ENABLE=4
$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 --yes set REAL_TIME_CLOCK_ENABLE=1

# NOTE: Requires a power cycle of the host for those settings to take effect

# Verify that the NIC FW changes have been applied

$ sudo mlxconfig -d /dev/mst/mt41686_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\|INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\|INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"

(continues on next page)
### 1.1.2.19 Set Persistent NIC Interface Name

Configure the network link files so that the NIC interfaces always come up with the same name. Run `lshw -c network -businfo` to find the current interface name on the target bus address then run `ip link` to find the corresponding MAC address by the interface name. After identifying the MAC address, create files at `/etc/systemd/network/NN-persistent-net.link` with the following information:

```
[Match]
MACAddress= {{item.mac}}

[Link]
Name= {{item.name}}
```

The following network link files set the converged accelerator port#0 to aerial00 and port#1 to aerial01:

```
$ sudo nano /etc/systemd/network/11-persistent-net.link

# Update the MAC address to match the converged accelerator port 0 MAC address
[Match]
MACAddress= 48:b0:2d:xx:xx:xx

[Link]
Name= aerial00

$ sudo nano /etc/systemd/network/12-persistent-net.link

# Update the MAC address to match the converged accelerator port 1 MAC address
[Match]
MACAddress= 48:b0:2d:yy:yy:yy

[Link]
Name= aerial01
```

Reboot the system after creating these files.
1.1.2.20 Install ptp4l and phc2sys

Enter these commands to configure PTP4L assuming the aerial00 NIC interface and CPU core 41 are used for PTP:

$ cat <<EOF | sudo tee /etc/ptp.conf
[global]
dataset_comparison G.8275.x
G.8275.defaultDS.localPriority 128
maxStepsRemoved 255
logAnnounceInterval -3
logSyncInterval -4
logMinDelayReqInterval -4
G.8275.portDS.localPriority 128
network_transport L2
domainNumber 24
tx_timestamp_timeout 30
slaveOnly 1
clock_servo pi
step_threshold 1.0
egressLatency 28
pi_proportional_const 4.65
pi_integral_const 0.1
[aerial00]
announceReceiptTimeout 3
delay_mechanism E2E
network_transport L2
EOF

cat <<EOF | sudo tee /lib/systemd/system/ptp4l.service
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l
After=network.target

[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStartPre=ifconfig aerial00 up
ExecStartPre=ethtool --set-priv-flags aerial00 tx_port_ts on
ExecStartPre=ethtool -A aerial00 rx off tx off
ExecStartPre=ifconfig aerial01 up
ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
ExecStartPre=ethtool -A aerial01 rx off tx off
ExecStart=taskset -c 41 /usr/sbin/ptp4l -f /etc/ptp.conf

[Install]
WantedBy=multi-user.target
EOF

$ sudo systemctl daemon-reload
$ sudo systemctl restart ptp4l.service
$ sudo systemctl enable ptp4l.service

One server becomes the master clock, as shown below:
$ sudo systemctl status ptp4l.service

• ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2023-08-08 19:37:56 UTC; 2 weeks 3 days ago
  Docs: man:ptp4l
  Main PID: 1120 (ptp4l)
  Tasks: 1 (limit: 94533)
  Memory: 460.0K
  CPU: 9min 8.089s
  CGroup: ∕system.slice∕ptp4l.service

Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp4l[81287.043]: selected local clock
  b8cef6.fffe.d333be as best master
Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp4l[81287.043]: port 1: assuming the
  grand master role
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: timed out while
  polling for tx timestamp
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: increasing tx_
  timestamp_timeout may correct this issue, but it is likely caused by a driver bug
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: port 1: send sync
  failed
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp4l[263223.379]: port 1: MASTER to
  FAULTY on FAULT_DETECTED (FT_UNSPECIFIED)
Aug 11 20:45:07 aerial-devkit taskset[1120]: ptp4l[263239.522]: port 1: FAULTY to
  LISTENING on INIT_COMPLETE
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: port 1: LISTENING to
  MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: selected local clock
  b8cef6.fffe.d333be as best master
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp4l[263239.963]: port 1: assuming the
  grand master role

The other becomes the secondary, follower clock, as shown below:

$ sudo systemctl status ptp4l.service

• ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2023-08-22 16:25:41 UTC; 3 days ago
  Docs: man:ptp4l
  Main PID: 3251 (ptp4l)
  Tasks: 1 (limit: 598810)
  Memory: 472.0K
  CPU: 2min 48.984s
  CGroup: /system.slice/ptp4l.service

Aug 25 19:58:34 aerial-r750 taskset[3251]: ptp4l[272004.187]: rms 8 max 15 freq -
  14495 +/- 9 delay 11 +/- 0
Aug 25 19:58:35 aerial-r750 taskset[3251]: ptp4l[272005.187]: rms 6 max 12 freq -
  14480 +/- 7 delay 11 +/- 1
Aug 25 19:58:36 aerial-r750 taskset[3251]: ptp4l[272006.187]: rms 8 max 12 freq -
  14465 +/- 5 delay 10 +/- 0

(continues on next page)
Enter the commands to turn off NTP:

```bash
$ sudo timedatectl set-ntp false
$ timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
Universal time: Thu 2022-02-03 22:30:58 UTC
RTC time: Thu 2022-02-03 22:30:58
Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: no
NTP service: inactive
RTC in local TZ: no
```

Run PHC2SYS as service:

PHC2SYS is used to synchronize the system clock to the PTP hardware clock (PHC) on the NIC.

Specify the network interface used for PTP and system clock as the slave clock.

```bash
# If more than one instance is already running, kill the existing
# PHC2SYS sessions.

# Command used can be found in /lib/systemd/system/phc2sys.service
# Update the ExecStart line to the following
$ cat <<EOF | sudo tee /lib/systemd/system/phc2sys.service
[Unit]
Description=Syncronize system clock or PTP hardware clock (PHC)
Documentation=man:phc2sys
After=ntpdate.service
Requires=ptp4l.service
After=ptp4l.service

[Service]
Restart=always
RestartSec=5s
Type=simple
# Gives ptp4l a chance to stabilize
ExecStartPre=sleep 2
ExecStart=/bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp$(ethtool -T aerial00| grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -0 0 -R -v 256 -u 256"

[Install]
```

(continues on next page)
After the PHC2SYS config file is changed, run the following:

```bash
$ sudo systemctl daemon-reload
$ sudo systemctl restart phc2sys.service

# Set to start automatically on reboot
$ sudo systemctl enable phc2sys.service
```

Verify that the system clock is synchronized:

```bash
$ timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
    Universal time: Thu 2022-02-03 22:30:58 UTC
    RTC time: Thu 2022-02-03 22:30:58
    Time zone: Etc/UTC (UTC, +0000)

System clock synchronized: yes
NTP service: inactive
RTC in local TZ: no
```
1.1.2.21 Setup the Boot Configuration Service

Create the directory `/usr/local/bin` and create the `/usr/local/bin/nvidia.sh` file to run the commands with every reboot. The command for “nvidia-smi lgc” expects just one GPU device (-i 0). This needs to be modified, if the system uses more than one GPU.

```bash
$ cat <<"EOF" | sudo tee /usr/local/bin/nvidia.sh
#!/bin/bash
mst start
nvidia-smi -i 0 -lgc $(nvidia-smi -i 0 --query-supported-clocks=graphics --format=csv, _noheader,nounits | sort -h | tail -n 1)
nvidia-smi -mig 0
echo -1 > /proc/sys/kernel/sched_rt_runtime_us
EOF
```

Create a system service file to be loaded after network interfaces are up.

```bash
$ cat <<EOF | sudo tee /lib/systemd/system/nvidia.service
[Unit]
After=network.target

[Service]
ExecStart=/usr/local/bin/nvidia.sh

[Install]
WantedBy=default.target
EOF
```

Create a system service file for `nvidia-persistenced` to be run at startup.

Note: This file was created following the sample from `/usr/share/doc/NVIDIA_GLX-1.0/samples/nvidia-persistenced-init.tar.bz2`

```bash
cat <<EOF | sudo tee /lib/systemd/system/nvidia-persistenced.service
[Unit]
Description=NVIDIA Persistence Daemon
Wants=syslog.target

[Service]
Type=forking
ExecStart=/usr/bin/nvidia-persistenced
ExecStopPost=/bin/rm -rf /var/run/nvidia-persistenced

[Install]
WantedBy=multi-user.target
EOF
```

Then set the file permissions, reload the systemd daemon, enable the service, restart the service when installing the first time, and check status

```bash
sudo chmod 744 /usr/local/bin/nvidia.sh
sudo chmod 664 /lib/systemd/system/nvidia.service
sudo chmod 664 /lib/systemd/system/nvidia-persistenced.service
```

(continues on next page)
sudo systemctl daemon-reload
sudo systemctl enable nvidia-persistenced.service
sudo systemctl enable nvidia.service
sudo systemctl restart nvidia.service
sudo systemctl restart nvidia-persistenced.service
sudo systemctl status nvidia.service
sudo systemctl status nvidia-persistenced.service

The output of the last command should look like this:

```
airial@server:~$ sudo systemctl status nvidia.service
nvidia.service
   Loaded: loaded (/lib/systemd/system/nvidia.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2024-06-07 20:26:06 UTC; 2s ago
     Process: 251860 ExecStart=/usr/local/bin/nvidia.sh (code=exited, status=0/SUCCESS)
   Main PID: 251860 (code=exited, status=0/SUCCESS)
         CPU: 788ms

Jun 07 20:26:05 server nvidia.sh[251862] : Starting MST (Mellanox Software Tools)
   driver set
Jun 07 20:26:05 server nvidia.sh[251862] : Loading MST PCI module - Success
Jun 07 20:26:05 server nvidia.sh[251862] : [warn] mst_pciconf is already loaded, skipping
Jun 07 20:26:06 server nvidia.sh[252732] : Unloading MST PCI module (unused) - Success
Jun 07 20:26:06 server nvidia.sh[252732] : GPU clocks set to "(gpuClkMin 1410, gpuClkMax 1410)" for GPU 00000000:CF:00.0

airial@server:~$ sudo systemctl status nvidia-persistenced.service
nvidia-persistenced.service - NVIDIA Persistence Daemon
   Loaded: loaded (/lib/systemd/system/nvidia-persistenced.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-06-07 20:25:57 UTC; 3s ago
     Process: 251836 ExecStart=/usr/bin/nvidia-persistenced (code=exited, status=0/SUCCESS)
   Main PID: 251837 (nvidia-persistenced)
         Tasks: 1 (limit: 598792)
       Memory: 672.0K
         CPU: 9ms
     CGroup: /system.slice/nvidia-persistenced.service
           251837 /usr/bin/nvidia-persistenced

Jun 07 20:25:57 server systemd[1] : Starting NVIDIA Persistence Daemon...
Jun 07 20:25:57 server nvidia-persistenced[251837] : Started (251837)
```
1.1.3. Installing and Upgrading Aerial cuBB

You must update the dependent software components to the specific version listed in the Release Manifest.

If you are upgrading a Grace Hopper MGX system, follow Installing Tools on Grace Hopper to upgrade the dependent SW first.

If you are upgrading a Dell R750 system with A100X converged accelerator, follow Installing Tools on Dell R750 to upgrade the dependent SW first.

1.1.3.1 Removing the Old Aerial cuBB Container

This step is optional. To remove the old cuBB container, enter the following commands:

```bash
$ sudo docker stop <cuBB container name>
$ sudo docker rm <cuBB container name>
```

1.1.3.2 Installing the New Aerial cuBB Container

The cuBB container is available on the NVIDIA GPU Cloud (NGC). Follow the instructions on that page to pull the container and to run the container.

**Note:** If you receive the cuBB container image via nvonline, run “docker load < cuBB container image file” to load the image. Then use the same docker run command detailed on the NGC page to launch it.

1.1.4. Aerial System Scripts

1.1.4.1 System Configuration Validation Script

Included in the release package is a script that checks and displays key system configuration settings that are important for running the Aerial cuBB SDK.

```bash
$ pip3 install psutil
$ cd $cuBB_SDK/cuPHY/util/cuBB_system_checks
$ sudo -E python3 ./cuBB_system_checks.py
```

The output of cuBB_system_checks.py may differ slightly between bare-metal and container versions of the environment. The script helps to retrieve the software-component versions and hardware configuration. Refer to the Release Manifest in the cuBB Release Notes to ensure the correct software-component versions are installed. Below is an example output on a bare-metal platform:

```
# To get the system or ptp info, the command has to run on the host.
$ sudo -E python3 ./cuBB_system_checks.py --sys

-----General---------------------------
Hostname : smc-gh-01
IP address : 192.168.1.100
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

Linux distro : "Ubuntu 22.04.3 LTS"
Linux kernel version : 6.5.0-1019-nvidia

Manufacturer : Supermicro
Product Name : ARS-111GL-NHR
Base Board Manufacturer : Supermicro
Base Board Product Name : G1SMH-G
Chassis Manufacturer : Supermicro
Chassis Type : Other
Chassis Height : 1U
Processor : Grace A02
Max Speed : Unknown
Current Speed : 3402 MHz

$ sudo -E python3 ./cuBB_system_checks.py

Hostname : smc-gh-01
IP address : 192.168.1.100
Linux distro : "Ubuntu 22.04.3 LTS"
Linux kernel version : 6.5.0-1019-nvidia

Audit subsystem : audit=0
Clock source : N/A
HugePage count : hugepages=32
HugePage size : hugepagesize=512M
CPU idle time management : idle=poll
Max Intel C-state : N/A
Intel IOMMU : N/A
IO MMU : N/A
Isolated CPUs : N/A
Corrected errors : N/A
Adaptive-tick CPUs : nohz_full=4-47
Soft-lockup detector disable : nosoftlockup
Max processor C-state : processor.max_cstate=0
RCU callback polling : rcu_nocb_poll
No-RCU-callback CPUs : rcu_nocbs=4-47
TSC stability checks : tsc=reliable

CPU cores : 72
Thread(s) per CPU core : 1
CPU MHz : N/A
CPU sockets : 1

CUDA_DEVICE_MAX_CONNECTIONS : N/A
cuBB_SDK : N/A

HugePage count : 32
Free HugePages : 31
HugePage size : 524288 kB
Shared memory size : 240G

GPU driver version : 555.42.02
CUDA version : 12.5
GPU0

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

GPU product name : NVIDIA GH200 480GB
GPU persistence mode : Enabled
Current GPU temperature : 36 C
GPU clock frequency : 1980 MHz
Max GPU clock frequency : 1980 MHz
GPU PCIe bus id : 00000009:01:00.0

-----GPUDirect topology---------------------------

<table>
<thead>
<tr>
<th>NUMA ID</th>
<th>NIC0</th>
<th>NIC1</th>
<th>NIC2</th>
<th>NIC3</th>
<th>CPU Affinity</th>
<th>NUMA Affinity</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>0-71</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NIC0</td>
<td>SYS</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIC1</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>SYS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIC2</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>PIX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIC3</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
X = Self
SYS = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UIP)
NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
PHB = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
PBX = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
PIX = Connection traversing at most a single PCIe bridge
NV# = Connection traversing a bonded set of # NVLinks

NIC Legend:
NIC0: mlx5_0
NIC1: mlx5_1
NIC2: mlx5_2
NIC3: mlx5_3

-----Mellanox NICs---------------------------

NIC0
NIC product name : BlueField3
NIC part number : 900-9D3B6-00CV-A_Ax
NIC PCIe bus id : /dev/mst/m41692_pciconf1
NIC FW version : 32.41.1000
FLEX_PARSER_PROFILE_ENABLE : 4
PROG_PARSE_GRAPH : True(1)
ACCURATE_TX_SCHEDULER : True(1)
CQE_COMPRESSION : AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE : True(1)

NIC1
NIC product name : BlueField3
NIC part number : 900-9D3B6-00CV-A_Ax
NIC PCIe bus id : /dev/mst/m41692_pciconf0
NIC FW version : 32.41.1000
FLEX_PARSER_PROFILE_ENABLE : 4
PROG_PARSE_GRAPH : True(1)
ACCURATE_TX_SCHEDULER : True(1)
CQE_COMPRESSION : AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE : True(1)

---- Mellanox NIC Interfaces----------------------

**Interface0**
- Name : aerial00
- Network adapter : mlx5_0
- PCIe bus id : 0000:01:00.0
- Ethernet address : 94:6d:ae:c7:62:00
- Operstate : up
- MTU : 1514
- RX flow control : off
- TX flow control : off
- PTP hardware clock : 0
- QoS Priority trust state : pcp
- PCIe MRRS : 4096 bytes

**Interface1**
- Name : aerial01
- Network adapter : mlx5_1
- PCIe bus id : 0000:01:00.1
- Ethernet address : 94:6d:ae:c7:62:01
- Operstate : up
- MTU : 1500
- RX flow control : off
- TX flow control : off
- PTP hardware clock : 1
- QoS Priority trust state : pcp
- PCIe MRRS : 512 bytes

**Interface2**
- Name : aerial02
- Network adapter : mlx5_2
- PCIe bus id : 0002:01:00.0
- Ethernet address : 94:6d:ae:c7:6b:80
- Operstate : down
- MTU : 1500
- RX flow control : on
- TX flow control : on
- PTP hardware clock : 2
- QoS Priority trust state : pcp
- PCIe MRRS : 512 bytes

**Interface3**
- Name : aerial03
- Network adapter : mlx5_3
- PCIe bus id : 0002:01:00.1
- Ethernet address : 94:6d:ae:c7:6b:81
- Operstate : down
- MTU : 1500
- RX flow control : on
- TX flow control : on
- PTP hardware clock : 3
- QoS Priority trust state : pcp
- PCIe MRRS : 512 bytes

---- Linux PTP------------------------------------

ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2024-06-05 21:42:18 UTC; 6h ago
  Docs: man:ptp4l

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

```
   phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
   Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-06-05 21:42:20 UTC; 6h ago
   Docs: man:phc2sys
   Process: 4529 ExecStartPre=sleep 2 (code=exited, status=0/SUCCESS)
   Main PID: 4573 (sh)
   Tasks: 2 (limit: 146899)
   Memory: 2.1M
   CPU: 1min 14.399s
   CGroup: /system.slice/phc2sys.service
        /bin/sh -c "taskset -c 47 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -T aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -0 0 -R 256 -u 256"
        4587 /usr/sbin/phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -n 24 -0 0 -R 256 -u 256
```

1.1. cuBB Installation Guide

59
1.1.4.1.1 Checking the NIC Status

To query back the Mellanox NIC firmware settings initialized with the script above, use these commands:

```bash
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\|INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\|INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"
```

```
INTERNAL_CPU_MODEL              EMBEDDED_CPU(1)
INTERNAL_CPU_PAGE_SUPPLIER      EXT_HOST_PF(1)
INTERNAL_CPU_ESWITCH_MANAGER   EXT_HOST_PF(1)
INTERNAL_CPU_IB_VPORT0          EXT_HOST_PF(1)
INTERNAL_CPU_OFFLOAD_ENGINE    DISABLED(1)
FLEX_PARSER_PROFILE_ENABLE     4
PROG_PARSE_GRAPH               True(1)
ACCURATE_TX_SCHEDULER          True(1)
CQE_COMPRESSION                AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE         True(1)
```

(continues on next page)
To check the current status of a NIC port, use this command:

```
$ sudo mlxlink -d /dev/mst/mt41692_pciconf0
```

**Operational Info**

---
- State: Active
- Physical state: LinkUp
- Speed: 200G
- Width: 4x
- FEC: Standard_RS-FEC - (544,514)
- Loopback Mode: No Loopback
- Auto Negotiation: ON

**Supported Info**

---
- Enabled Link Speed (Ext.): 0x00003ff2 (200G_2X, 200G_4X, 100G_1X, 100G_2X, 100G_4X, 40G_1X, 40G_2X, 40G_4X, 10G_1X, 10G_2X, 10G_4X, 500G_1X, 500G_2X, 1G_1X, 1G_2X, 1G_4X)
- Supported Cable Speed (Ext.): 0x000017f2 (200G_4X, 100G_2X, 100G_4X, 500G_1X, 500G_2X, 40G_1X, 40G_2X, 40G_4X, 10G_1X, 10G_2X, 10G_4X, 500G_1X, 500G_2X)

**Troubleshooting Info**

---
- Status Opcode: 0
- Group Opcode: N/A
- Recommendation: No issue was observed

**Tool Information**

---
- Firmware Version: 32.41.1000
- amBER Version: 3.2
- MFT Version: mft 4.28.0-92

Alternatively, you can use the System Configuration Validation Script to obtain a full list of configuration settings.

### 1.1.5. Troubleshooting

This page documents solutions to common issues that you might encounter.
1.1.5.1 Hugepages Issues

Normally the hugepages settings are updated through the /etc/default/grub configuration file. However, depending on the version of operating system, the settings changes may become overwritten by another configuration file: /etc/grub.

1.1.5.2 Remove Old CUDA Toolkit and Driver

If the system has an old version installed, run the following to remove the CUDA Toolkit and driver:

```
sudo apt-get --purge remove "*cublas*" "*cufft*" "*curand*" "*cusolver*" "*cusparse*"
"*npp*" "*nvjpeg*" "cuda*" "*nsight*" "*nvidia*"
sudo apt-get autoremove
```

1.1.5.3 How to Fix Apt Update Error Due to Incorrect System Time

You may see the apt update error if the system time is incorrect.

```
E: Release file for https://download.docker.com/linux/ubuntu/dists/focal/InRelease is
   not valid yet (invalid for another 2d 10h 51min 11s).
Updates for this repository will not be applied.
```

Run the following commands to set the date and time via NTP once (this will not enable the NTP service):

```
sudo apt-get install ntpdate
sudo ntpdate -s pool.ntp.org
```

1.1.5.4 How to Resize the Default LVM Volume

When installing Ubuntu 22.04 server, it partitions the whole disk but only creates a 200GB logical volume. This is what you will see on a newly installed devkit:

```
# Devkit has 1TB SSD but default lv uses only 200GB
lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0 7:0 0 55.5M 1 loop /snap/core18/2246
loop1 7:1 0 55.5M 1 loop /snap/core18/2253
loop2 7:2 0 67.3M 1 loop /snap/1xd/21545
loop3 7:3 0 67.2M 1 loop /snap/1xd/21835
loop4 7:4 0 61.9M 1 loop /snap/core20/1242
loop5 7:5 0 61.9M 1 loop /snap/core20/1169
loop6 7:6 0 32.5M 1 loop /snap/snapd/13640
loop7 7:7 0 42.2M 1 loop /snap/snapd/14066
sda 8:0 0 894.3G 0 disk
   sda1 8:1 0 512M 0 part /boot/efi
   sda2 8:2 0 1G 0 part /boot
   sda3 8:3 0 892.8G 0 part
ubuntu--vg-ubuntu--lv 253:0 0 200G 0 lvm /
```

The following commands resize the logic volume to use the entire disk, then resize the file system to use the entire logic volume.
# Test mode first
```
sudo lvresize -t -v -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
```

# Remove -t if test mode succeeds
```
sudo lvresize -v -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
```
```
lssblk
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAJ:MIN</th>
<th>RM</th>
<th>SIZE</th>
<th>RO</th>
<th>TYPE</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop0</td>
<td>7:0</td>
<td>0</td>
<td>55.5M</td>
<td>1</td>
<td>loop</td>
<td>/snap/core18/2246</td>
</tr>
<tr>
<td>loop1</td>
<td>7:1</td>
<td>0</td>
<td>55.5M</td>
<td>1</td>
<td>loop</td>
<td>/snap/core18/2253</td>
</tr>
<tr>
<td>loop2</td>
<td>7:2</td>
<td>0</td>
<td>67.3M</td>
<td>1</td>
<td>loop</td>
<td>/snap/lxd/21545</td>
</tr>
<tr>
<td>loop3</td>
<td>7:3</td>
<td>0</td>
<td>67.2M</td>
<td>1</td>
<td>loop</td>
<td>/snap/lxd/21835</td>
</tr>
<tr>
<td>loop4</td>
<td>7:4</td>
<td>0</td>
<td>61.9M</td>
<td>1</td>
<td>loop</td>
<td>/snap/core20/1242</td>
</tr>
<tr>
<td>loop5</td>
<td>7:5</td>
<td>0</td>
<td>61.9M</td>
<td>1</td>
<td>loop</td>
<td>/snap/core20/1169</td>
</tr>
<tr>
<td>loop6</td>
<td>7:6</td>
<td>0</td>
<td>32.5M</td>
<td>1</td>
<td>loop</td>
<td>/snap/snapd/13640</td>
</tr>
<tr>
<td>loop7</td>
<td>7:7</td>
<td>0</td>
<td>42.2M</td>
<td>1</td>
<td>loop</td>
<td>/snap/snapd/14066</td>
</tr>
<tr>
<td>sda</td>
<td>8:0</td>
<td>0</td>
<td>894.3G</td>
<td>0</td>
<td>disk</td>
<td></td>
</tr>
<tr>
<td>sda1</td>
<td>8:1</td>
<td>0</td>
<td>512M</td>
<td>0</td>
<td>part</td>
<td>/boot/efi</td>
</tr>
<tr>
<td>sda2</td>
<td>8:2</td>
<td>0</td>
<td>1G</td>
<td>0</td>
<td>part</td>
<td>/boot</td>
</tr>
<tr>
<td>sda3</td>
<td>8:3</td>
<td>0</td>
<td>892.8G</td>
<td>0</td>
<td>part</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ubuntu--vg-ubuntu--lv</td>
</tr>
</tbody>
</table>

# Resize file system
```
sudo resize2fs -p /dev/mapper/ubuntu--vg-ubuntu--lv
```
```
df -h -T
```

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Type</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>udev</td>
<td>devtmpfs</td>
<td>39G</td>
<td>0</td>
<td>39G</td>
<td>0%</td>
<td>/dev</td>
</tr>
<tr>
<td>tmpfs</td>
<td>tmpfs</td>
<td>9.4G</td>
<td>2.0M</td>
<td>9.4G</td>
<td>1%</td>
<td>/run</td>
</tr>
<tr>
<td>/dev/mapper/ubuntu--vg-ubuntu--lv</td>
<td>ext4</td>
<td>878G</td>
<td>77G</td>
<td>764G</td>
<td>10%</td>
<td>/</td>
</tr>
<tr>
<td>tmpfs</td>
<td>tmpfs</td>
<td>47G</td>
<td>0</td>
<td>47G</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>tmpfs</td>
<td>tmpfs</td>
<td>5.0M</td>
<td>0</td>
<td>5.0M</td>
<td>0%</td>
<td>/run/lock</td>
</tr>
<tr>
<td>tmpfs</td>
<td>tmpfs</td>
<td>47G</td>
<td>0</td>
<td>47G</td>
<td>0%</td>
<td>/sys/fs/cgroup</td>
</tr>
<tr>
<td>/dev/sda2</td>
<td>ext4</td>
<td>976M</td>
<td>460M</td>
<td>450M</td>
<td>51%</td>
<td>/boot</td>
</tr>
<tr>
<td>/dev/loop0</td>
<td>squashfs</td>
<td>56M</td>
<td>56M</td>
<td>0</td>
<td>100%</td>
<td>/snap/core18/2246</td>
</tr>
<tr>
<td>/dev/sda1</td>
<td>vfat</td>
<td>511M</td>
<td>5.3M</td>
<td>506M</td>
<td>2%</td>
<td>/boot/efi</td>
</tr>
<tr>
<td>/dev/loop1</td>
<td>squashfs</td>
<td>56M</td>
<td>56M</td>
<td>0</td>
<td>100%</td>
<td>/snap/core18/2253</td>
</tr>
<tr>
<td>/dev/loop5</td>
<td>squashfs</td>
<td>62M</td>
<td>62M</td>
<td>0</td>
<td>100%</td>
<td>/snap/core20/1169</td>
</tr>
<tr>
<td>/dev/loop2</td>
<td>squashfs</td>
<td>68M</td>
<td>68M</td>
<td>0</td>
<td>100%</td>
<td>/snap/lxd/21545</td>
</tr>
<tr>
<td>/dev/loop4</td>
<td>squashfs</td>
<td>62M</td>
<td>62M</td>
<td>0</td>
<td>100%</td>
<td>/snap/core20/1242</td>
</tr>
<tr>
<td>/dev/loop6</td>
<td>squashfs</td>
<td>33M</td>
<td>33M</td>
<td>0</td>
<td>100%</td>
<td>/snap/snapd/13640</td>
</tr>
<tr>
<td>/dev/loop3</td>
<td>squashfs</td>
<td>68M</td>
<td>68M</td>
<td>0</td>
<td>100%</td>
<td>/snap/lxd/21835</td>
</tr>
<tr>
<td>/dev/loop7</td>
<td>squashfs</td>
<td>43M</td>
<td>43M</td>
<td>0</td>
<td>100%</td>
<td>/snap/snapd/14066</td>
</tr>
<tr>
<td>overlay</td>
<td>overlay</td>
<td>878G</td>
<td>77G</td>
<td>764G</td>
<td>10%</td>
<td>/var/lib/docker/</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>--overlay2/851cbfd83b022a24f61fb0f87a007c56da8065a7528f6b661bf45d3d65cc787/merged</td>
</tr>
<tr>
<td>tmpfs</td>
<td>tmpfs</td>
<td>9.4G</td>
<td>4.0K</td>
<td>9.4G</td>
<td>1%</td>
<td>/run/user/1000</td>
</tr>
</tbody>
</table>
1.1.5.5 How to Identify the NIC Interface Name and MAC Address

Use the `sudo lshw -c network |grep -i 'product\|bus info\|name\|serial'` command to find the bus address and MAC address of each NIC on the system. Here is an example:

```
$ sudo lshw -c network |grep -i 'product\|bus info\|name\|serial'
product: I210 Gigabit Network Connection
  bus info: pci@0000:05:00.0
  logical name: eno1
  serial: 18:c0:4d:79:49:b6
product: I210 Gigabit Network Connection
  bus info: pci@0000:06:00.0
  logical name: enp6s0
  serial: 18:c0:4d:79:49:b7
product: MT2892 Family [ConnectX-6 Dx]
  bus info: pci@0000:b5:00.0
  logical name: ens6f0
  serial: b8:ce:f6:33:fd:ee
product: MT2892 Family [ConnectX-6 Dx]
  bus info: pci@0000:b5:00.1
  logical name: ens6f1
  serial: b8:ce:f6:33:fd:ef
```

1.2. cuBB Quickstart Guide

This section explains how to run the Aerial cuBB software examples.

**Important Terms**

<table>
<thead>
<tr>
<th>Term or Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>Software suite that accelerates 5G RAN functions with NVIDIA GPUs</td>
</tr>
<tr>
<td>cuBB</td>
<td>CUDA GPU software libraries/tools that accelerate 5G RAN compute-intensive processing</td>
</tr>
<tr>
<td>cuPHY</td>
<td>CUDA 5G PHY layer software library of the cuBB</td>
</tr>
<tr>
<td>cuPHY-CP</td>
<td>cuPHY control-plane software</td>
</tr>
<tr>
<td>HDF5</td>
<td>A data file format used for storing test vectors. The HDF5 software library provides the functions for reading and writing test vectors.</td>
</tr>
<tr>
<td>CMake</td>
<td>A software tool for configuring the makefiles for building the CUDA examples (<a href="https://cmake.org/">https://cmake.org/</a>)</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>DOCA</td>
<td>DOCA is a software framework that helps developers create applications and services on top of the NVIDIA BlueField networking platform.</td>
</tr>
<tr>
<td>GDR</td>
<td>GPUDirect RDMA</td>
</tr>
<tr>
<td>FH</td>
<td>Fronthaul</td>
</tr>
<tr>
<td>TV</td>
<td>Test Vector</td>
</tr>
</tbody>
</table>
1.2.1. cuBB Quickstart Overview

The diagrams below show the Aerial cuBB software and hardware components.

- **cuPHY** is the GPU-Accelerated 5G PHY layer software library and examples. It provides GPU-offloaded 5G signal processing.

- **DPDK** is the software library that provides network data transfer acceleration. The public version of DPDK now contains features like eCPRI flow steering and accurate TX scheduling, which Aerial uses.

- **cuPHY-CP** is the cuPHY Control-Plane software that provides the control plane interface between the layer 1 cuPHY and the upper layer stack.

Shown below is the block diagram of the cuPHY-CP. It supports multi-cell. Included with cuPHY-CP are the built-in test MAC and RU emulator modules.

The Aerial cuBB makes use of the DPDK for the network interface. It provides efficient high-speed network data connectivity to GPU processing of network data.

The diagram below shows the overall Aerial cuBB software and hardware stack layers:
1.2.2. Generating TV and Launch Pattern Files

Since the cuBB 22-2.2 release, the test vectors are not included in the release package. You must generate the TV files before running cuPHY examples or cuBB end-to-end test.

**Note:** TV generation is **NOT** supported on ARM because Matlab Compiler SDK doesn’t support it yet.

1.2.2.1 Using Aerial Python mcore Module

No Matlab license is required to generate TV files using the Aerial Python mcore module. The cuBB container already has aerial_mcore installed.

To generate the test vectors required for end-to-end testing, follow these steps:

1. Run the following inside the Aerial container. It completes in less than a minute.

```bash
cd ${cuBB_SDK}/5GModel/aerial_mcore/examples
source ./scripts/setup.sh
dir .scripts/gen_e2e_ota_tvs.sh
ls -lh GPU_test_input/
cp GPU_test_input/* ${cuBB_SDK}/testVectors/
```

The following is example output from the above commands:

```
aerial@c_aerial_aerial:/opt/nvidia/cuBB/5GModel/aerial_mcore$ source ./scripts/setup.sh
[Aerial Python] aerial@c_aerial_aerial:/opt/nvidia/cuBB/5GModel/aerial_mcore$ ./
dir .scripts/gen_e2e_ota_tvs.sh
```

(continues on next page)
2. Copy the output to the testVectors folder.

To generate all of the TV files, including files that are not necessary for E2E testing, follow these steps:

1. Run the following commands inside the Aerial container.

   ```bash
   cd ${cuBB_SDK}/5GModel/aerial_mcore/examples
   source ..∕scripts/setup.sh
   export REGRESSION_MODE=1
   time python3 .∕example_5GModel_regression.py allChannels
   echo $?
   ls -aF GPU_test_input/
   du -h GPU_test_input/  
   ```

   **Note:** The TV generation may take a few hours on the devkit with the current isocpus parameter setting in the kernel command line. The host must have at least 64GB of memory and 430GB of available disk space. Hyperthreading must be enabled.

2. Review the output from the above commands; an example is shown below. The “real” time takes less than one hour on a 24-core x86 host. The `echo $?` command shows the exit code of the process, which should be 0, while a non-zero exit code indicates a failure.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Compliance_Test</th>
<th>Error</th>
<th>Test_Vector</th>
<th>Error</th>
<th>Performance_Test</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSB</td>
<td>37</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PDCCH</td>
<td>71</td>
<td>0</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PDSCH</td>
<td>274</td>
<td>0</td>
<td>286</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CSIRS</td>
<td>96</td>
<td>0</td>
<td>87</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DLMIX</td>
<td>0</td>
<td>0</td>
<td>1049</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PRACH</td>
<td>60</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>PUCCH</td>
<td>469</td>
<td>0</td>
<td>469</td>
<td>0</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>PUSCH</td>
<td>388</td>
<td>0</td>
<td>398</td>
<td>0</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>SRS</td>
<td>125</td>
<td>0</td>
<td>125</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ULMIX</td>
<td>58</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BFW</td>
<td>58</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

   Total 1568 0 3230 0 185 0

   **Total time for runRegression is 2147 seconds**

   Parallel pool using the `local` profile is shutting down.

   ```
   real 36m51.931s
   user 585m1.704s
   sys 10m28.322s
   ```
To generate the launch pattern for each test case using cubb_scripts, follow these steps:

1. Run the following commands:
   ```bash
cd $cuBB_SDK
cd cubb_scripts
python3 auto_lp.py -i ./5GModel/aerial_mcore/examples/GPU_test_input -t launch_pattern_nrSim.yaml
```

2. Copy the launch pattern and TV files to the testVectors repo:
   ```bash
cd $cuBB_SDK
cp ./5GModel/aerial_mcore/examples/GPU_test_input/*h5 ./testVectors/.
cp ./5GModel/aerial_mcore/examples/GPU_test_input/launch_pattern* ./testVectors/multi-cell/.
```

### 1.2.2.2 Using Matlab

To generate TV files using Matlab:

1. Run the following command in Matlab:
   ```matlab
cd('nr_matlab'); startup; [nTC, errCnt] = runRegression({'TestVector'}, {'allChannels'}, 'compact', [0, 1]);
```

   All the cuPHY TVs are generated and stored under nr_matlab/GPU_test_input.

2. Generate the launch pattern for each test case using cubb_scripts:
   ```bash
cd $cuBB_SDK
cd cubb_scripts
python3 auto_lp.py -i ./5GModel/nr_matlab/GPU_test_input -t launch_pattern_nrSim.yaml
```

3. Copy the launch pattern and TV files to testVectors repo.
   ```bash
cd $cuBB_SDK
cp ./5GModel/nr_matlab/GPU_test_input/TVnr_* ./testVectors/.
cp ./5GModel/nr_matlab/GPU_test_input/launch_pattern* ./testVectors/multi-cell/.
```

### 1.2.3. Running Aerial cuPHY

Aerial cuPHY provides the cuPHY library and several examples that link with the library. Here we include instructions on using MATLAB to generate TVs. Please refer to [Generating TV and Launch Pattern Files](#) for using Aerial Python mcore Module to generate TVs.
1.2.3.1 Building Aerial cuPHY

1.2.3.1.1 Prerequisites

The following instructions assume the system configuration and Aerial cuBB installation are done. If not, see the cuBB Install Guide to complete the installation or upgrade process.

After powering on the system, use the following commands to verify that the GPU and NIC are in the correct state:

```bash
# Verify GPU is detected and CUDA driver version matches the release manifest.
$ nvidia-smi

# Verify NIC is detected: Example CX6-DX
$ sudo lshw -c network -businfo
```

<table>
<thead>
<tr>
<th>Bus info</th>
<th>Device</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pci@0000:05:00.0</td>
<td>eno1</td>
<td>network</td>
<td>I210 Gigabit Network Connection</td>
</tr>
<tr>
<td>pci@0000:06:00.0</td>
<td>enp6s0</td>
<td>network</td>
<td>I210 Gigabit Network Connection</td>
</tr>
<tr>
<td>pci@0000:b5:00.0</td>
<td>ens6f0</td>
<td>network</td>
<td>MT2892 Family [ConnectX-6 Dx]</td>
</tr>
<tr>
<td>pci@0000:b5:00.1</td>
<td>ens6f1</td>
<td>network</td>
<td>MT2892 Family [ConnectX-6 Dx]</td>
</tr>
</tbody>
</table>

# Verify the link state is right. Assuming NIC port 0 is connected.

```bash
$ sudo mlxlink -d b5:00.0
```

Operational Info

---------------

State : Active
Physical state : LinkUp
Speed : 100G
Width : 4x
FEC : Standard RS-FEC - RS(528,514)
Loopback Mode : No Loopback
Auto Negotiation : ON

Supported Info

--------------

Enabled Link Speed (Ext.) : 0x000007f2 (100G_2X,100G_4X,50G_1X,50G_2X,40G,25G, ...10G,1G)
Supported Cable Speed (Ext.) : 0x000002f2 (100G_4X,50G_2X,40G,25G,10G,1G)

Troubleshooting Info

------------

Status Opcode : 0
Group Opcode : N/A
Recommendation : No issue was observed.
1.2.3.1.2 Set Up the Host Environment

Set up the environment by following the cuBB Installation Guide for the server type you are using.

1.2.3.1.3 Launch the cuBB Container

Use the following command to launch the cuBB container:

```bash
$ sudo docker exec -it cuBB /bin/bash
```

1.2.3.1.4 Build Aerial cuPHY in the Container

Build cuPHY in the cuBB container using the following commands:

```bash
$ cd /opt/nvidia/cuBB/cuPHY
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native -DCMAKE_INSTALL_PREFIX=./install
$ cmake --build build
```

cuPHY is, by default, built in Release mode. The option BUILD_DOCS=ON is also enabled by default to allow the make to generate the Doxygen documentation for the cuPHY library API. To disable this option, pass -DBUILD_DOCS=OFF to the CMake command line. The output directory is cuPHY/install/docs.

To put the built cuPHY headers and libraries into an installation directory so that other applications using the cuPHY library can compile and link with cuPHY, use the commands from the current build directory:

```bash
$ cmake --install build
```

This creates the include and lib directories under the cuPHY/install directory.

1.2.3.2 Building and running on separate servers

When building the source code on one server, and running the binaries on another server, it might be important to use the correct toolchain for the target.

The source code directory cuPHY/cmake/toolchains contains toolchains for the following targets:

- x86-64: devkit, r750, x86-64
- arm: grace-cross, bf3

A new toolchain file might need to be created if using a different target.

The toolchain file defines what compiler to use, and the value of AERIAL_ARCH_TUNE_FLAGS

One way to make sure that the flag is correct, is to do the following:

Run the aerial_sdk container on the target, inside the container run the following command:

```bash
$ gcc -march=native -Q --help=target
```

Run the aerial_sdk container on the build server, inside the container run the following command:
$ gcc -march=<march for target> -Q --help=target

Make sure the outputs from both commands are the same. Create a toolchain file and use it when building aerial_sdk:

$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/my-target

1.2.3.3 Running the cuPHY Examples

This section describes how to run the Aerial cuPHY standalone example programs. They read test vector data files as input. Refer to the Supported Test Vector Configurations section of the cuPHY Release Notes to determine which test vectors to use for different configurations. Do not use old test vectors from previous cuBB releases with the example programs of this release.

1.2.3.3.1 Generating Test Vectors using Matlab 5GModel

Run this Matlab command:

```
cd('nr_matlab'); startup; [nTC, errCnt] = runRegression({'TestVector'}, {'allChannels →'}, 'compact', [0, 1]);
```

All the cuPHY test vectors are generated and stored under nr_matlab/GPU_test_input.

1.2.3.3.2 Instructions for Testing cuPHY Channels Manually

1.2.3.3.2.1 PUSCH

**Test Vectors**
Match test vector name with PUSCH\_gNB\_CUPHY\_*\_h5

**How to Run**

- Streams mode: cuPHY/build/examples/pusch\_rx\_multi\_pipe/cuphy\_ex\_pusch\_rx\_multi\_pipe -i ~/<tv\_name>.h5
- Graphs mode: cuPHY/build/examples/pusch\_rx\_multi\_pipe/cuphy\_ex\_pusch\_rx\_multi\_pipe -i ~/<tv\_name>.h5 -m 1

**Expected Outcome**
Test 1 (CRC test KPI): All test cases must have zero CRC errors (only CRC errors, not correct ones, are reported when the channel is run).
1.2.3.3.2 PUCCH

Test Vectors
Match test vector name with $PUCCH_F^*\_gNB\_CUPHY\_*\_h5$

How to Run
$PUCCH$ format $0/1/2/3$: cuPHY/build/examples/pucch_rx_pipeline/
cuphy_ex_pucch_rx_pipeline -i <tv_name>

Expected Outcome

► cuphy_ex_pucch_Fx_receiver checks if the test vector includes PFx UCI first.
► If the test-vector UCI format is not expected, it displays "No PFx UCI received".

```bash
./build/examples/pucch_F0_receiver/cuphy_ex_pucch_F0_receiver -i
./GPU_test_input/TVnr_6109_PUCCH_gNB_CUPHY_s0p126.h5
---------- COMPUTE-SANITIZER
No PF0 UCI received.
---------- LEAK SUMMARY: 0 bytes leaked in 0 allocations
---------- ERROR SUMMARY: 0 errors
```

► If the test-vector UCI format is expected, it compares UCI output.xzsd.

```bash
./build/examples/pucch_F0_receiver/cuphy_ex_pucch_F0_receiver -i
./GPU_test_input/TVnr_6026_PUCCH_gNB_CUPHY_s0p1.h5
---------- COMPUTE-SANITIZER
comparing cuPHY F0 UCI output to reference output: 0 mismatches out of 1 UCIs
---------- LEAK SUMMARY: 0 bytes leaked in 0 allocations
---------- ERROR SUMMARY: 0 errors
```

1.2.3.3.2.3 PRACH

Test Vectors
Match test vector name with $PRACH\_gNB\_CUPHY\_*\_h5$

How to Run
cuPHY/build/examples/prach_receiver_multi_cell/prach_receiver_multi_cell -i
<tv_name> -r <num_iteration> -k

Expected Outcome

► prach_receiver_multi_cell compares against the reference measurements in the test vector.
► Measured values are displayed and if they are within tolerance the message is displayed:

```
========> Test PASS
```
1.2.3.3.2.4 PDSCH

Test Vectors
Match test vector name with $PDSCH\_gNB\_CUPHY\_*.h5$

How to Run
▶ PDSCH in non-AAS mode, streams: cuPHY/build/examples/pdsch_tx/cuphy_ex_pdsch_tx $\sim/<tv\_name>.h5\ 2\ 0\ 0$
▶ PDSCH in non-AAS mode, graphs: cuPHY/build/examples/pdsch_tx/cuphy_ex_pdsch_tx $\sim/<tv\_name>.h5\ 2\ 0\ 1$

Expected Outcome
Test 1 (correctness against reference model): Channel reports correct match with reference model

1.2.3.3.2.5 PDCCH

Test Vectors
Match test vector name with $PDCCH\_gNB\_CUPHY\_*.h5$

How to Run
▶ Streams mode: cuPHY/build/examples/pdcch/embed_pdcch_tf_signal -i $\sim/<tv\_name>.h5\ -m\ 0$
▶ Graphs mode: cuPHY/build/examples/pdcch/embed_pdcch_tf_signal -i $\sim/<tv\_name>.h5\ -m\ 1$

Expected Outcome
Test 1 (correctness against reference model): Test PASS

1.2.3.3.2.6 SSB

Test Vectors
Match test vector name with $SSB\_gNB\_CUPHY\_*.h5$

How to Run
▶ Streams mode: cuPHY/build/examples/ss/testSS -i $\sim/<tv\_name>.h5\ -m\ 0$
▶ Graphs mode: cuPHY/build/examples/ss/testSS -i $\sim/<tv\_name>.h5\ -m\ 1$

Expected Outcome
Test 1 (correctness against reference model): Test PASS
1.2.3.3.7 CSI-RS

Test Vectors
Match test vector name with \textit{CSIRS}\_\textit{gNB}\_\textit{CUPHY}\_*.h5

How to Run

\begin{itemize}
  \item Streams mode: \texttt{cuPHY/build/examples/csi\_rs/nzp_csi\_rs\_test -i <tv\_name> -m 0}
  \item Graphs mode: \texttt{cuPHY/build/examples/csi\_rs/nzp_csi\_rs\_test -i <tv\_name> -m 1}
\end{itemize}

Expected Outcome
Test 1 (correctness against reference model): Test PASS

1.2.3.3.8 SRS

Test Vectors
Match test vector name with \textit{SRS}\_\textit{gNB}\_\textit{CUPHY}\_*.h5

How to Run

\begin{itemize}
  \item Streams mode: \texttt{cuPHY/build/examples/srs\_rx\_pipeline/cuphy\_ex\_srs\_rx\_pipeline -i <tv\_name> -r <num\_iteration> -m 0}
  \item Graphs mode: \texttt{cuPHY/build/examples/srs\_rx\_pipeline/cuphy\_ex\_srs\_rx\_pipeline -i <tv\_name> -r <num\_iteration> -m 1}
\end{itemize}

Expected Outcome
Test 1 (correctness against reference model): SRS reference check: PASSED!; Timing results are provided

1.2.3.3.9 BFC

Test Vectors
Match test vector name with \textit{BFW}\_\textit{gNB}\_\textit{CUPHY}\_*.h5

How to Run

\begin{itemize}
  \item Streams mode: \texttt{cuPHY/build/examples/bfc/cuphy\_ex\_bfc -i <tv\_name> -r <num\_iteration> -m 0}
  \item Graphs mode: \texttt{cuPHY/build/examples/bfc/cuphy\_ex\_bfc -i <tv\_name> -r <num\_iteration> -m 1}
  \item Add \texttt{-c} to enable reference check (default disabled)
\end{itemize}

Expected Outcome
Test 1 (measure latency without reference check): Timing results are provided
Test 2 (correctness against reference model using \texttt{-c}): Test PASS; Timing results are provided
1.2.3.3.3 Instructions for LDPC Performance Test

The `ldpc_perf_collect.py` Python script from the cuPHY repository can be used to perform error rate tests for the cuPHY LDPC decoder. There are test input files defined for \( Z = \{64, 128, 256, 384\} \), \( BG = \{1,2\} \). The current tests check whether the block error rate (BLER, also sometimes referred to as Frame Error Rate or FER) is less than 0.1.

From the build directory, the following commands run the tests:

```bash
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG1_→Z64_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG1_→Z128_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG1_→Z256_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG1_→Z384_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG2_→Z64_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG2_→Z128_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG2_→Z256_BLER0.1.txt -f -w 800 -P
..∕util∕ldpc∕ldpc_perf_collect.py --mode test -i ..∕util∕ldpc∕test∕ldpc_decode_BG2_→Z384_BLER0.1.txt -f -w 800 -P
```

Each test input file contains multiple tests for different code rates, as specified by the number of parity nodes.

1.2.3.4 Running cuPHY Performance Testing Scripts

`aerial_sdk/testBenches` provides a multi-cell multi-channel test bench to test cuPHY standalone performance. It relies on NVIDIA Multi-Process Service (MPS) to share the GPU among multiple channels. Specifically, there are two folders and their relationship can be summarized as follows:
Aerial CUDA-Accelerated RAN, Release 24-2

**cubb_gpu_test_bench**: a C test bench that runs the multi-cell multi-channel cuPHY standalone GPU workload (that is, without I/O to and from NIC or layer 2). The input of `cubb_gpu_test_bench` are test vectors, a Yaml file, and some command options to run the GPU workload. The output is a buffer-XX.txt file that has the logs, channel start/end times, debug info, etc. Here XX is the number of cells used in testing.

**perf**: a set of Python scripts to automate performance testing using `cubb_gpu_test_bench`. The Python scripts can help generate the Yaml file and command options, config GPU and MPS before running `cubb_gpu_test_bench`; collect the test results by reading the output buffer-XX.txt from `cubb_gpu_test_bench`.

### 1.2.3.4.1 Generating Test Vectors using Matlab 5GModel

Run this Matlab command:

```matlab
cd <5GModel root>/nr_matlab
startup
genCfgTV_perf_ss('performance-avg.xlsm');
genCfgTV_perf_ss_bwc('performance-avg.xlsm');
genCfgTV_perf_pucch();
genCfgTV_perf_pdcch();
genCfgTV_perf_prach();
genCfgTV_perf_csirs();
genCfgTV_perf_ssb();
genCfgTV_perf_srs();
```

All the cuPHY Performance test vectors are generated and stored under nr_matlab/GPU_test_input.
1.2.3.4.2 Measuring cuPHY Performance using cubb_gpu_test_bench

Requirements:

- The performance measurements can be run using a Linux environment making one or more GPU available. Such environment is here assumed to have:
  - bash or zsh as default shell
  - Python 3.8+ and the following packages: numpy, pyCUDA, pyYAML
  - CUDA toolkit 11.4 or above properly configured so that \textit{nvidia-cuda-mps-control} and \textit{nvidia-smi} are in PATH
  - The executable \texttt{cubb\_gpu\_test\_bench} is located in the \texttt{<testBenches>/build} folder.

There are three steps when measuring cell capacity using \texttt{cubb\_gpu\_test\_bench}. The \textit{perf} folder provides some predefined test cases. Below is an example of 4T4R (F08) using TDD pattern DDDSU-DDDD.

1. Generate the JSON file that defines the use case (e.g., 8~16 peak or average cells)

   \begin{verbatim}
   python3 generate_avg_TDD.py --peak 8 9 10 11 12 13 14 15 16 --avg 0 --exact --case F08
   \end{verbatim}

2. Measure the latency of all channels based on predefined patterns

   \begin{verbatim}
   python3 measure.py --cuphy <testBenches>/build --vectors <test_vectors> --config
   ... --freq <GPU_freq> --start <cell_start> --cap <cell_cap> --iterations 1 --slots
   ... --nSlots --power <budget> --target <sms_prach> <sms_pdcch> <sms_pucch> <sms_pdsch>
   ... --sms_pusch --sms_ssb --2cb_per_sm --save_buffer --priority --prach --prach_isolate
   ... --pdcch --pdcch_isolate --pucch --pucch_isolate --tdd_pattern dddsuudddd
   ... --pusch_cascaded --ssb --csirs --groups_dl --pack_pdsch --groups_pusch --ldpc_parallel
   ... --graph
   \end{verbatim}

   where:
   - \texttt{<GPU\_ID>} is the ID of the GPU on which the measurements are to be run; e.g., 0 for single GPU systems
   - \texttt{<GPU\_freq>} is the GPU clock frequency in MHz
   - \texttt{<cell\_start>} is the minimum number of cells to be tested
   - \texttt{<cell\_cap>} is the maximum number of cells to be tested. The Python scripts will run \texttt{cubb\_gpu\_test\_bench} for a range of \([<cell\_start>, <cell\_cap>]\) cells and collect the latency results
   - \texttt{<budget>} is the power budget in Watts
   - \texttt{<sms\_channelName>} is the number of streaming multiprocessors used per MPS sub-context for each channel during the run, where \textit{channelName} can be PRACH, PDCCH, PUCCH, PDSCH, PUSCH, SSB
   - \texttt{--graph} add this option to run in graph mode, otherwise in stream mode
   - Notes: use \texttt{--test} to see what yaml file and commend options the Python scripts generated without running the tests on GPU

3. Visualize the latency of each channel (this step requires Python library matplotlib). We generate \texttt{compare-<date>.png} showing the CDF of the latency for all tested channels:

   - if run in stream mode:
Aerial CUDA-Accelerated RAN, Release 24-2

```python
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_<sms_pushc>_<sms_ssb>_sweep_streams_avg_F08.json --cells <nCell>+0
```

▶ if run in graph mode:

```python
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_<sms_pushc>_<sms_ssb>_sweep_graphs_avg_F08.json --cells <nCell>+0
```

where:

▶ `<nCell>` is the number of cells we would like to visualize the latency results

It is possible to compare latency results of different number of cells in one figure. For instance, we can compare the latency of 8 cells and 9 cells:

```python
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_<sms_pushc>_<sms_ssb>_sweep_graphs_avg_F08.json <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_<sms_pushc>_<sms_ssb>_sweep_graphs_avg_F08.json --cells 8+0 9+0
```

In all cases, Aerial CUDA-Accelerated RAN offers the possibility of measuring the latency of all workloads including:

▶ Dynamic and heterogeneous traffic (meaning that each cell is stimulated with different test vectors and every slot sees a different allocation of the test vectors to the considered cells)

▶ Specific traffic models

1.2.4. Running cuBB End-to-End

Beyond the cuPHY layer 1 PHY software and its standalone examples, this section describes how to build and run the cuBB software components shown in the block diagram below.

▶ The cuPHYController block operates between L2 and the RU fronthaul interface. It interfaces through cuPHY and DOCA GPUNetIO + DPDK to operate the GPU and the NIC.

▶ L2 Adapter: This module communicates with L2 or TestMAC through FAPI messages over nvIPC. It receives downlink and uplink scheduling commands from L2 and converts it to internal cuPHYDriver API calls.

▶ cuPHYDriver: This module distributes the UL and DL tasks among the available worker threads. It interacts with GPU for the following tasks:

▶ To prepare and trigger a new UL/DL cuPHY processing through the cuPHY API.

▶ To launch UL packets ordering with CUDA kernel.

It interacts with DPU/NIC through the Fronthaul Driver to send and receive ORAN fronthaul packets (C/U-plane).

▶ The RU Emulator emulates the network traffic of single or multiple RU. It validates the following:

▶ All packet timing for DL direction packets (i.e. DL-C, UL-C, DL-U) based on configurable ORAN packet windows.

▶ It checks for all packets that the eCPRI packet structure is aligned to ORAN specs.

▶ It validates the IQ samples in the DL U-plane payload and expected section sizes for different compression methods.

▶ It validates the BFW IQ samples in DL/UL C-plane, and RE mask in DL-C for CSI-RS/PDSCH.
The TestMAC simulates the L2 and provides the FAPI interface over nvIPC. It validates the following:

▶ It calculates the expected throughput data from the launch pattern and TVs and print to console. Then a python script can be used to validate the throughput of both TestMAC and RU. The throughput data include: Prmb/HARQ/SR/CSI/SRS number, channel numbers, DL/UL data rate. Unit is number per seconds.

▶ It validates the UL FAPI message data structure and TB buffers by comparing with the preloaded data from TVs.

▶ It validates the UL FAPI timing (The number of slots that the UL FAPI messages expect to receive).

The cuPHYController is exercised with an environment between the RU Emulator and the TestMAC.

The L1/L2 interface is based on the 5G FAPI 222.10.02 with partial 222.10.04 defined by the Small Cell Forum (SCF). For the supported message and PDU types and exceptions, refer to cuBB Release Notes.

1.2.4.1 Building the cuBB End-to-End

The following procedure describes the steps for building the end-to-end components in Aerial cuBB.

1. Inside the cuBB container, use the following command:

```bash
$ cd /opt/nvidia/cuBB
```

2. Create the build directory:

```bash
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native
```

**Note:** The compile time flag `DYNAMIC_SFN_SLOT` has been replaced by the l2_adapter yaml startup time option `enableTickDynamicSfnSlot`. The default is 1 (Dynamic SFN Slot) instead of 0. For more information, refer to the cuPHY Controller documentation.
3. Choose your build options:

- If building to enable supported FAPI 10.04 fields (for example, SRS), then add the following flag:

  `$ cmake <existing flags> -DSCF_FAPI_10_04=ON`

- If building to run Test Mode (TM) tests, then add the following flag:

  `$ cmake <existing flags> -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON`

- To run more than 16 cells, please add the below cmake flag

  `$ cmake <existing flags> -DENABLE_20C=ON`

- To build for Hopper GPU on Grace Hopper MGX system, then add the following flag:

  `$ cmake <existing flags>`

Here is the table of supported build variants:

<table>
<thead>
<tr>
<th>RU Type \ Build Options</th>
<th>FAPI 10.02: Default (no build flag)</th>
<th>Enable FAPI 10.04 fields: -DSCF_FAPI_10_04=ON</th>
<th>Enable Test Mode: -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON</th>
<th>Enable 20C: -DENABLE_20C=ON</th>
<th>Grace Hopper MGX: -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON</th>
</tr>
</thead>
</table>

**Note:** When building for E2E test, “-DENABLE_L2_SLT_RSP=ON” is enabled by default in the cmake build options. It requires the L2 to support the vendor-specific message “SLOT.response”. If the L2 doesn’t support it, “-DENABLE_L2_SLT_RSP=OFF” must be included in the cmake build option to turn off this feature in L1.
ENABLE_L2_SLT_RSP=ON is recommended.

Option ENABLE_L2_SLT_RSP must be configured with the same value in L1, L2, and libnvipc.so standalone build for L2: (1) L1: cuBB_SDK. (2) libnvipc.so standalone build for L2. Refer to ${cuBB_SDK}/cuPHY-CP/gt_common_libs/README.md. (3) L2: gNB DU code which includes nv_ipc.h. To confirm whether it was enabled, run "grep ENABLE_L2_SLT_RSP build/CMakeCache.txt" for (1) and (2), print sizeof(nv_ipc_t) in L2 code for (3).

1.2.4.1.1 Additional Build Options

For all of our F08 performance benchmarking, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -DSCF_-
   →FAPI_10_04=ON -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON
```

For 20C on GH testing, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -DSCF_-
   →FAPI_10_04=ON -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON -DENABLE_20C=ON
```

For UL heavy TDD pattern testing, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -DENABLE_-
   →UL_HEAVY_TDD=ON
```

To build all Aerial cuBB components, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native
$ cmake --build build
```

To build only the cuPHY, use these commands:

```
$ cd ${cuBB_SDK}/cuPHY
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native
$ cmake --build build
```

To build only the Test MAC, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t test_mac
```

To build only the cuPHY controller, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t cuphycontroller_scf
```

To build only the cuPHY driver, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t cuphydriver
```

To build only the RU emulator, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t ru_emulator
```
To compile the Aerial code in the container on a devkit or Dell R750 machine that has isolcpus restricting cores, you can override isolcpus using the following command:

```bash
$ sudo chrt -r 1 sudo -u aerial taskset -c 10-20 cmake --build build
```

The example command uses cores 10-20.

### 1.2.4.2 nvlog configuration

Aerial-SDK use nvlog as logger. It is based on the opensource FMT logger. Configuration file is located at `.∕cuPHY∕nvlog∕config∕nvlog_config.yaml`.

Log files are stored at `/_tmp` directory by default and the path can be overridden by environment variable `AERIAL_LOG_PATH`.

Maximum log file size can be configured by `max_file_size_bytes` to avoid exhausting the system disk storage.

To configure global log level, set “shm_log_level: <level>”. To configure log level for a specific tag, add a “shm_level: <level>” line under the tag name line. As an example, below configuration sets global log level to 3 - CONSOLE level and sets "FH.LATE_PACKETS" tag to 5 - INFO level.

```yaml
# log files stored at /tmp directory (default)
# log file path can be customized using environment variable $AERIAL_LOG_PATH
# Log levels: 0 - NONE, 1 - FATAL, 2 - ERROR, 3 - CONSOLE, 4 - WARNING, 5 - INFO, 6 - DEBUG, 7 - VERBOSE

nvlog:
  shm_log_level: 3  # Global log level
  max_file_size_bytes: 5000000000  # Size in bytes The rotating log files in /tmp
      (default)
  nvlog_tags:
    - 0:  # Reserve number 0 for no tag print
      shm_level: 5  # Example: overlay shm_log_level for a tag
    - 621: "FH.LATE_PACKETS"
      shm_level: 5
```

### 1.2.4.3 Updating Configuration Files for End-to-End

This section describes the config parameters that you can modify to run end-to-end.

#### 1.2.4.3.1 Server #1 (to Run TestMAC and cuPHYController)

There are several common configurations. Check and edit the following parameters in the `.yaml` file:

Configure the NIC address in the following configuration files depending on the setup you are using, these are the default files provided:

- `cuphycontroller_F08_CG1.yaml`
- `cuphycontroller_F08_R750.yaml`
- `cuphycontroller_nrSim_SCF.yaml`
Edit the NIC PCIe address to match the NIC hardware PCIe address. For example, the FH NIC on R750 gNB uses PCIe address 0000:cc:00.0:

$ sed -i "s/ nic:.*/nic: 0000:cc:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml

Check the GPU ID for the GPU that is sharing the PCIe switch with the NIC. The gpus parameter shown below has a default value of 0 for a GPU ID of 0. If GPU 0 is not the GPU you want to use, replace 0 in the sed command line and run it:

$ sed -i "/gpus:/{n;s/.*/   - 0/}" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml

If the system has only one GPU card, you can keep the default setting of 0.

To identify which GPU is sharing the PCIe switch with the NIC, use the following command:

$ nvidia-smi topo -m

In the output, look for the GPU connected to the NIC with connection type of PIX (where they intersect in the table). In the example below, GPU 0 in the column is the one with the PIX intersecting with Mellanox mlx5_0 and mlx5_1. Use GPU ID value of 0 for the .yaml gpus parameter.

<table>
<thead>
<tr>
<th>GPU0 mlx5_0 mlx5_1</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>X PIX PIX</td>
<td>0-23</td>
</tr>
<tr>
<td>mlx5_0 PIX X</td>
<td></td>
</tr>
<tr>
<td>mlx5_1 PIX X</td>
<td></td>
</tr>
</tbody>
</table>

The meaning of PIX is:

- **X** = Self
- **SYS** = Connection traversing PCIe and the SMP interconnect between NUMA nodes (e.g., QPI/UIP)
- **NODE** = Connection traversing PCIe and the interconnect between PCIe Host Bridges within a NUMA node
- **PHB** = Connection traversing PCIe and a PCIe Host Bridge (typically the CPU)
- **PXB** = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
- **PIX** = Connection traversing at most a single PCIe bridge
- **NV#** = Connection traversing a bonded set of # NVLinks

**Note:** Aerial-SDK expects the set of eAxCid ports to be the same between DL and UL channels (excluding PRACH). Make sure that the same set of port indices in the YAML configuration file are configured for DL and UL channels. For example, if the set of port indices [0,8,1,2] are configured for PDSCH, the same setting should be used for PDCCH, SSB/PBCH, and CSI-RS. Similarly, if the set of port indices [0,8] are configured for PUSCH, the same set of indices should be used for PUCCH. The number of eAxCid ports between DL and UL channels does not need to be the same.

To enable early HARQ set pushc_subSlotProcEn to 1 in cuphycontroller config:

```
$ sed -i "s/ pushc_subSlotProcEn:.*/ pushc_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml
```

To activate early HARQ set uciIndPerSlot to 2 in test_mac_config.yaml:

---

### 1.2. cuBB Quickstart Guide

---
Aerial CUDA-Accelerated RAN, Release 24-2

```
sed -i "s/ uciIndPerSlot :.*/ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
->test_mac_config.yaml
```

**Note:** This split UCI.indication with early-HARQ feature is enabled only in FAPI 10.04. To enable this feature, build with compilation flag -DSCF_FAPI_10_04=ON. This feature is enabled at cuPHY, if pusch_subSlotProcEn is set to 1 in cuphycontroller config. But cuPHY does not report early HARQ for UCI on PUSCH until L2 sends config.request with TLV 0x102B indicationInstancesPerSlot UCI.indication = 2. To instruct testMac to send this TLV in config.request set uciIndPerSlot to 2 in test_mac_config.yaml.

```
sed -i "s/ pusch_subSlotProcEn:.*/ pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/
cupyhycontroller/config/cupyhycontroller_F08_*.yaml
sed -i "s/ uciIndPerSlot :.*/ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
->test_mac_config.yaml
```

```
sed -i "s/ mCh_segment_proc_enable:.*/ mCh_segment_proc_enable: 1/" ${cuBB_SDK}/cuPHY-
..CP/cupyhycontroller/config/cupyhycontroller_F08_*.yaml
sed -i "s/ channel_segment_timelines:.*/ channel_segment_timelines: 1/" ${cuBB_SDK}/
cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```

**Note:** To enable enhanced L1-L2 interace, early-HARQ feature must be enabled as above and compiled with FAPI 10.04. To enable this feature, build with compilation flag -DSCF_FAPI_10_04=ON. To instruct testMac to send TLV CONFIG_TLV_VENDOR_CHAN_SEGMENT (0xA018), set channel_segment_timelines to 1 in test_mac_config.yaml. The expectation is that there is an Error.Indication sent when the timelines don’t meet the processing from cuPHYDriver.

1.2.4.3.2 Server #2 (to Run RU Emulator)

The RU emulator reads a configuration file located at: $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml.

Before running the ru-emulator, modify the config.yaml to match your server system hardware settings.

There are two parameters to modify in the config.yaml file:

```yaml
# PCI Address of NIC interface used
nic_interface: b5:00.0
# MAC address of cuPHYController port in use on server#1
peerethaddr: 1c:34:da:ff:ff:fe
```

Update the nic_interface and peerethaddr according to the systems used. Look up the addresses of these NIC interfaces.

- **nic_interface** is the NIC port PCIe bus address on the system running RU emulator. Replace 0000:b5:00.0 with the PCIe address of NIC for use.
- **peerethaddr** is the NIC port MAC address on the system running cuPHYController. Replace the MAC address with the MAC address of the NIC used in Server#1.

Replace 0000:b5:00.0 with the PCIe address of NIC port for use:
Replace the MAC address with the MAC address of the NIC port used in Server#1:

```bash
$ sed -i "s/nic_interface.*\nic_interface: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml

$ sed -i "s/peerethaddr.*\peerethaddr: 1c:34:da:ff:ff:fe/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
```

Run the following command on the host to identify the correct PCIe address and the MAC address.

```bash
$ sudo lshw -c network -businfo
```

table:

<table>
<thead>
<tr>
<th>Bus info</th>
<th>Device</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pci@0000:05:00.0</td>
<td>eno1</td>
<td>network</td>
<td>I210 Gigabit Network Connection</td>
</tr>
<tr>
<td>pci@0000:06:00.0</td>
<td>enp6s0</td>
<td>network</td>
<td>I210 Gigabit Network Connection</td>
</tr>
<tr>
<td>pci@0000:b5:00.0</td>
<td>ens6f0</td>
<td>network</td>
<td>MT2892 Family [ConnectX-6 Dx]</td>
</tr>
<tr>
<td>pci@0000:b5:00.1</td>
<td>ens6f1</td>
<td>network</td>
<td>MT2892 Family [ConnectX-6 Dx]</td>
</tr>
<tr>
<td>vethdf87878</td>
<td>network</td>
<td>Ethernet interface</td>
<td></td>
</tr>
</tbody>
</table>

To find the MAC address of the NIC port, run the following command:

```bash
$ ifconfig -a
```

The MAC addresses of the NIC port are under the link/ether label.

1.2.4.4 Running Environment Initialization for End-to-End

This section describes how to run the various cuBB software components together. Here, the cuBB uses the GPU and the NIC for cuPHY L1 compute and for network data traffic acceleration.

A network connection is used between the two servers to physically connect the RU emulator and the cuBB gNB software stack.
To verify that PTP4L and PHC2SYS services are running, run the following commands on the host:

```bash
$ sudo systemctl status ptp4l.service
...
# check that the service is active and has low rms value (<30):
$ sudo systemctl status phc2sys.service
```

Verify the System Clock is synchronized and that NTP is off:

```bash
$ timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
Universal time: Thu 2022-02-03 22:30:58 UTC
RTC time: Thu 2022-02-03 22:30:58
Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
NTP service: inactive
RTC in local TZ: no
```

### 1.2.4.5 Running Examples for End-to-End (SCF FAPI)

This section describes how to run the cuBB end-to-end using the SCF FAPI.

There are three use case examples:

- Use case 1: testMAC + SCF L2 Adapter Standalone
- Use case 2: testMAC + cuPHYController_SCF + RU Emulator
- Use case 3: testMAC + cuPHYController_SCF + RU Emulator P5G PRACH

#### 1.2.4.5.1 Running testMAC + SCF L2 Adapter Standalone

1. Build all the modules as described in *Building cuBB for End-to-End.*
2. Run l2adapter in standalone mode:
   ```bash
   sudo $cuBB_SDK/build/cuPHY-CP/scf2adapter/scf_app/cuppycontroller\/l2_adapter_cuppycontroller_scf
   ```
3. Run testMAC after l2adapter starts.
   You can run different cases:
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac <Fxx> <xC> [--channels <CHANNELS>] --no-validation

Examples:
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 1C --no-validation
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 2C --no-validation
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 3C --no-validation
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 4C --no-validation

4. Test result and test log: In the testMAC terminal output below, you can see the TTI tick counter and throughput:

```
08:32:15.793986 Cell 0 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400
   Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150
   INV 0
08:32:15.793996 Cell 1 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400
   Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150
   INV 0
08:32:15.794000 Cell 2 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400
   Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150
   INV 0
08:32:15.794003 Cell 3 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400
   Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150
   INV 0
```

1.2.4.5.2 Running testMAC + cuPHYController_SCF + RU Emulator

**Note:** Before running the cuBB test case, restart MPS in each run. Run the following commands to export environment variables and restart MPS in the cuphycontroller terminal (do not run this for test_mac and ru-emulator).

```
# Export variables
export CUDA_DEVICE_MAX_CONNECTIONS=8
export CUDA_MPS_PIPE_DIRECTORY=/var
export CUDA_MPS_LOG_DIRECTORY=/var

# Stop existing MPS
sudo -E echo quit | sudo -E nvidia-cuda-mps-control

# Start MPS
sudo -E nvidia-cuda-mps-control -d
sudo -E echo start_server -uid 0 | sudo -E nvidia-cuda-mps-control
```

The nvlog level can be changed in $cuBB_SDK/cuPHY/nvlog/config/nvlog_config.yaml if needed. For example, to change to console only log level:

```
name: phy
- shm_log_level: 5 # SHM log level
+ shm_log_level: 3 # SHM log level
```

Execute the following command to disable GPU (if there is one) for ru_emulator.
Export `CUDA_VISIBLE_DEVICES=""`

Export might not work in some system environments. In this case, add the value before command as shown in the following example:

```
sudo -E CUDA_VISIBLE_DEVICES="" ./ru_emulator xxx
```

Without `CUDA_VISIBLE_DEVICES=""`, the following log is seen when `ru_emulator` is started with a GPU on the host. It does not affect the functionality.

```
```

When running on a dual CPU R750 machine, you must be NUMA aware to get the best performance:

Verify the NUMA that the GPU is on, and configure the CPUs used/numactl accordingly.

Configure the workers in `$cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml` to use CPUs on the same NUMA node:

```
workers_ul:
  - 5
  - 7
workers_dl:
  - 11
  - 13
  - 15
```

Use numactl to ensure the memory allocation is also on the right NUMA node for the process:

```
sudo -E numactl -N 1 -m 1 ./cuphycontroller_scf F08_R750
sudo numactl -N 1 -m 1 ./test_mac x
```

### 1.2.4.5.2.1 Running the F08 Test Cases

Configure the `cell_group` in `$cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml`: Set `cell_group` to 1 and set `cell_group_num` to the number of cells to run.

For running on a R750 A100X machine: Configure the `cell_group` in `$cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml`:

For example, to run 1C:

```
cell_group: 1
cell_group_num: 1
```

To run 2C:

```
cell_group: 1
cell_group_num: 2
```

To run 3C:

```
cell_group: 1
cell_group_num: 3
```
To run 4C:

```yaml
cell_group: 1
cell_group_num: 4
```

F08 traffic patterns:

For Patterns 59 and 60, you must enable the OTA conformance features in `cuphycontroller_F08_R750.yaml`:

```yaml
pusch_tdi: 1
pusch_cfo: 1
pusch_to: 1
pusch_dftsofdm: 0
pusch_select_eqcoeffalgo: 1
pusch_polarDcdrListSz: 8
```

For Patterns 60 you must set the `pusch_nMaxPrb` for each cell in `cuphycontroller_F08_R750.yaml`:

```yaml
pusch_nMaxPrb: 136
```

For Pattern 61, you must set the `pusch_nMaxPrb` for each cell in `cuphycontroller_F08_CG1.yaml`, this allows us to test 20C on Grace Hopper system:

```yaml
pusch_nMaxPrb: 36
```

For patterns 59 and onwards, use 4 UL antenna streams, you must change these fields for the cuPHY-Controller config files:

```bash
# 4 UL Antenna config
sed -i 's/eAxC_UL: \[8, 0\]/eAxC_UL: \[8, 0, 1, 2\]/' ${cuBB_SDK}/cuPHY-CP/ru-emulator/...
cupycontroller/config/config.yaml
sed -i 's/eAxC_UL: \[1, 2\]/eAxC_UL: \[1, 2, 4, 9\]/' ${cuBB_SDK}/cuPHY-CP/...
cupycontroller/config/config.yaml
sed -i 's/eAxC_UL: \[0, 1\]/eAxC_UL: \[0, 1, 2, 3\]/' ${cuBB_SDK}/cuPHY-CP/...
cupycontroller/config/config.yaml
```

23-4 supports early HARQ processing. For the C and D variations of pattern 59 and 60, enable early HARQ processing with the following configurations:

```bash
# For early HARQ
sed -i 's/uciIndPerSlot :.*/uciIndPerSlot : 2/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/...
test_mac_config.yaml
```

(continues on next page)
# For early non HARQ

```bash
sed -i 's∕uciIndPerSlot :.*∕uciIndPerSlot : 0∕' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
   → test_mac_config.yaml
sed -i "s/perSubSlotProcEn :.*∕perSubSlotProcEn: 0∕" ${cuBB_SDK}/cuPHY-CP/
   → cuPHY-CP/config/cupycontroller/config/cupycontroller_F08_R750.yaml
```

# For Enhanced L1 - L2 Interface

```bash
sed -i 's∕uciIndPerSlot :.*∕uciIndPerSlot : 2∕' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
   → test_mac_config.yaml
sed -i "s/perSubSlotProcEn :.*∕perSubSlotProcEn: 1∕" ${cuBB_SDK}/cuPHY-CP/
   → cuPHY-CP/config/cupycontroller/config/cupycontroller_F08_R750.yaml
```

# Run F08 1C only as Enhanced L1 - L2 Interface is intended for 1 Cell.

For 23-3 and onwards, pattern 46 and 47 have become legacy patterns. Patterns 59 peak and 60
average are the latest patterns used for performance testing. On R750 A100X DU system F08 4C
with pattern 59 (peak pattern).

For performance testing, use the following settings for testMAC to adjust the schedule time of
the FAPI command, this requires a builder thread:

```bash
# testMAC configs for scheduling FAPI messages with appropriate L2 delay, also
   → configure testMAC to stop after 600k slots:
   sed -i 's/schedule_total_time :.*∕schedule_total_time: 470000∕' ${cuBB_SDK}/cuPHY-CP/
   → testMAC/testMAC/test_mac_config.yaml
sed -i 's/builder_thread_enable :.*∕builder_thread_enable: 1∕' ${cuBB_SDK}/cuPHY-CP/
   → testMAC/testMAC/test_mac_config.yaml
sed -i 's/fapi_delay_bit_mask :.*∕fapi_delay_bit_mask: 0xF∕' ${cuBB_SDK}/cuPHY-CP/
   → testMAC/testMAC/test_mac_config.yaml

# optionally configure the test duration with the number of test_slots. Keep test_slots:
   → 0 to run indefinitely.
   sed -i 's/test_slots : 0∕test_slots: 600000∕' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_
       → mac_config.yaml
```

# testMAC core configs, use free cores on the same NUMA, for example, the following
   → settings can be applied to an R750 using NUMA 1:
   sed -i -z 's/cpu_affinity :\s*\[0-9\]+∕cpu_affinity: 35/2' ${cuBB_SDK}/cuPHY-CP/
   → testMAC/testMAC/test_mac_config.yaml
sed -i -z 's/cpu_affinity :\s*\[0-9\]+∕cpu_affinity: 33/1' ${cuBB_SDK}/cuPHY-CP/
   → testMAC/testMAC/test_mac_config.yaml
```

You must enable the PUSCH conformance flags and RU Emulator validation to account for beamform-

```bash
# cuphycontroller configs for PUSCH conformance flags:
   sed -i "s/perSubSlotProcEn :.*∕perSubSlotProcEn: 1∕" ${cuBB_SDK}/cuPHY-CP/cupycontroller/config/
   → cuphycontroller_F08_R750.yaml
sed -i "s/perSubSlotProcEn :.*∕perSubSlotProcEn: 1∕" ${cuBB_SDK}/cuPHY-CP/cupycontroller/config/
   → cuphycontroller_F08_R750.yaml
```

(continues on next page)
To test 4T 4R TDD 7 beams series 59 and 60 with 80 slot patterns have been generated:

**Series 59c: 20C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed**

- PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
- PUSCH: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
- UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2
- PDCCH: 12 DCI / slot (6 DL + 6 UL)
- PUCCH: 18 UE frequency multiplexed (PF1)

**Frame 0**

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

**Frame 1**

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

**Frame 2**

- Slot 0, 1, 2: ssb *2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

**Frame 3**

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
Series 59d: 20C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 24 PUCCH UCIs freq-multiplexed:

- PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
- PUSCH: 6 UEG / slot, MCS 27, 41 PRBs / UEG, (33 PRBs / UEG when having 4 PRACH, 35 PRBs / UEG when having 3 PRACH)
- UCI@PUSCH: 0 HARQ, 37 CSI-1, 5 CSI-2
- PDCCH: 12 DCI / slot (6 DL + 6 UL)
- PUCCH: 24 UE frequency multiplexed (PF1)

Frame 0

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

Frame 1

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

Frame 2

- Slot 0, 1, 2: ssb *2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

Frame 3

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
- TRS/CSI-RS in symbol 6+10 / 12 for even case number
- TRS/CSI-RS in symbol 5+9 / 13 for odd case number

Series 59e: 30C peak cells, 7 beams, Full BW CSI-RS, 1 dmrs, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed

- Same settings as 59c expect that only 1 dmrs.
Aerial CUDA-Accelerated RAN, Release 24-2

- **PDSCH**: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
- **PUSCH**: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
- **UCI@PUSCH**: 4 HARQ, 37 CSI-1, 5 CSI-2
- **PDCCH**: 12 DCI / slot (6 DL + 6 UL)
- **PUCCH**: 18 UE frequency multiplexed (PF1)
- **TRS/CSI-RS** in symbol 6+10 / 12 for even case number
- **TRS/CSI-RS** in symbol 5+9 / 13 for odd case number
- **Series 60c**: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 18 PUCCH UCIs freq-multiplexed

- **PDSCH**: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having SSB)
- **PUSCH**: 6 UEG / slot, MCS 27, 19 PRBs / UEG, (11 PRBs / UEG when having 4 PRACH, 13 PRBs / UEG when having 3 PRACH)
- **UCI@PUSCH**: 4 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
- **PDCCH**: 12 DCI / slot (6 DL + 6 UL)
- **PUCCH**: 18 UE frequency multiplexed (PF1)

**Frame 0**

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

**Frame 1**

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

**Frame 2**

- Slot 0, 1, 2: ssb *2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

**Frame 3**

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number

▶ Series 60d: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 24 PUCCH UCIs freq-multiplexed:

▶ PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)
▶ PUSCH: 6 UEG / slot, MCS 27, 18 PRBs / UEG, (10 PRBs / UEG when having 4 PRACH, 12 PRBs / UEG when having 3 PRACH)
▶ UCI@PUSCH: 0 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)
▶ PUCCH: 24 UE frequency multiplexed (PF1)

▶ Frame 0

▶ Slot 0, 1, 2: ssb (2 blocks),
▶ Slot 3, ssb (1 block)
▶ Slot 6,8,10,16, TRS + CSIRS
▶ Slot 7,9,11,17, TRS
▶ Slot 5,15, PRACH

▶ Frame 1

▶ Slot 6,8,10, TRS + CSIRS
▶ Slot 7,9,11, TRS
▶ Slot 5,15, PRACH

▶ Frame 2

▶ Slot 0, 1, 2: ssb *2,
▶ Slot 3, ssb
▶ Slot 6,7,8,9,10,11, 16,17 TRS
▶ Slot 5,15, PRACH

▶ Frame 3

▶ Slot 6,7,8,9,10,11 TRS
▶ Slot 5,15, PRACH
▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number

▶ Series 62c: 30C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed, PUSCH in S slot

▶ 59c + 4 symbols of pusch in S slot
PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
PUSCH: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2
PDCCH: 12 DCI / slot (6 DL + 6 UL)
PUCCH: 18 UE frequency multiplexed (PF1)

Frame 0
- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

Frame 1
- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

Frame 2
- Slot 0, 1, 2: ssb *2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

Frame 3
- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
- TRS/CSI-RS in symbol 6+10 / 12 for even case number
- TRS/CSI-RS in symbol 5+9 / 13 for odd case number

Series 63c: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 18 PUCCH UCIs freq-multiplexed, PUSCH in S slot

- 59c + 4 symbols of pusch in S slot
- PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)
- PUSCH: 6 UEG / slot, MCS 27, 19 PRBs / UEG, (11 PRBs / UEG when having 4 PRACH, 13 PRBs / UEG when having 3 PRACH)
- UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
- PDCCH: 12 DCI / slot (6 DL + 6 UL)
PUCCH: 18 UE frequency multiplexed (PF1)

Frame 0

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

Frame 1

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

Frame 2

- Slot 0, 1, 2: ssb *2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

Frame 3

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
- TRS/CSI-RS in symbol 6+10 / 12 for even case number
- TRS/CSI-RS in symbol 5+9 / 13 for odd case number

For best performance the following example commands include numactl, which can be used for systems with the GPU located on NUMA 1 on a multi-NUMA system. For single NUMA systems, the numactl -N 1 -m 1 part of the command can be omitted.

```bash
sudo -E numactl -N 1 -m 1 ./cuphycontroller_scf F08_R750
sudo numactl -N 1 -m 1 ./test_mac F08 4C 59
sudo ./ru_emulator F08 4C 59
```

(continues on next page)
On R750 A100X DU system F08 4C with pattern 60 (average pattern):

```
sudo -E numactl -N 1 -m 1 ./cuphycontroller_scf F08_R750
sudo numactl -N 1 -m 1 ./test_mac F08 4C 60
sudo ./ru_emulator F08 4C 60
```

### 1.2.4.5.2.2 Simultaneous FH Port Test Configs with RU Emulator

The following TC can be tested with both FH ports:

- BFP9 2C 59c

To set up the two port test, you must set up the configurations appropriately.

You can choose between the following verified 2 port test topologies:

- **1 GH and 1 RU server**
  
  - GH P0 <-> RU P0
  
  - GH P1 <-> RU P1

- **1 GH and 2 RU server**
  
  - GH P0 <-> RU 1 P0
  
  - GH P1 <-> RU 2 P0

Note: For the scenario with 1 GH and 2 RU server, we need the three setups to be synchronized, i.e. with a FH switch as the PTP master in between the three systems.

cuPHYController configuration:
In the cuPHYController cell configurations, you could set port that the cell would run traffic on:

```yaml
nics:
  - nic: 0000:01:00.0
    mtu: 1514
    cpu_mbufs: 196608
    uplane_tx_handles: 64
    txq_count: 48
    rxq_count: 16
    txq_size: 8192
    rxq_size: 16384
    gpu: 0
  - nic: 0000:01:00.1
    mtu: 1514
    cpu_mbufs: 196608
    uplane_tx_handles: 64
    txq_count: 48
    rxq_count: 16
    txq_size: 8192
    rxq_size: 16384
    gpu: 0

In the cuPHYController cell configurations, you could set port that the cell would run traffic on:

```yaml
cells:
  - name: O-RU 0
    [...]
    nic: 0000:01:00.0
  - name: O-RU 1
    [...]
    nic: 0000:01:00.1

For the first topology with a single RU emulator system, you could specify the NIC interfaces and the peer ethernet addresses with the address of the DU ports, for example:

```yaml
nics:
  - nic_interface: 0000:cc:00.0
  - nic_interface: 0000:cc:00.1

peers:
  - peerethaddr: 48:b0:2d:a6:28:02 # MAC address of DU port 0
  - peerethaddr: 48:b0:2d:a6:28:03 # MAC address of DU port 1

Similarly for RU emulator config, appropriately assign the NIC and peer addresses, based on the index in the lists defined above:

```yaml

```
1.2.4.5.3 Running the nrSim Test Cases

1.2.4.5.3.1 PBCH

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 1901 --channels PBCH
sudo ./ru_emulator nrSim 1901 --channels PBCH
# Expect RU Emulator to report 100 PBCH per second

1.2.4.5.3.2 PDCCH_DL

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 2901 --channels PDCCH_DL
sudo ./ru_emulator nrSim 2901 --channels PDCCH_DL
# Expect RU Emulator to report 100 PDCCH_DL per second

1.2.4.5.3.3 PDSCH

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 3901 --channels PDSCH
sudo ./ru_emulator nrSim 3901 --channels PDSCH
# Expect RU Emulator to report 100 PDSCH per second

1.2.4.5.3.4 PUSCH

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7901 --channels PUSCH
sudo ./ru_emulator nrSim 7901 --channels PUSCH
# Expect testMAC to report 100 PUSCH per second

# PUSCH Mapping Type B
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7258 --channels PUSCH
sudo ./ru_emulator nrSim 7258 --channels PUSCH
# Expect testMAC to report 100 PUSCH per second

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7259 --channels PUSCH
sudo ./ru_emulator nrSim 7259 --channels PUSCH
# Expect testMAC to report 100 PUSCH per second

#CSI P2
sed -i "s/enable_csp2_v3.*/enable_csp2_v3: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
--config/cuphycontroller_nrSim_SCF.yaml

(continues on next page)
sed -i "s∕enable_csip2_v3.*∕enable_csip2_v3: 1∕" $cuBB_SDK/cuPHY-CP/testMAC/testMAC/
  → test_mac_config.yaml

# Restart MPS
sed -i "s∕ uciIndPerSlot :.*∕ uciIndPerSlot : 2∕" $cuBB_SDK/cuPHY-CP/testMAC/testMAC/
  → test_mac_config.yaml
sudo -E ./cuphycontroller_scf nrSim_SCF
dudo ./test_mac nrSim 7599 --channels PUSCH
dudo ./ru_emulator nrSim 7599 --channels PUSCH
# Expect testMAC to report 100 PUSCH and 100 CSIP2 per second

# Restart MPS
sed -i "s∕ uciIndPerSlot :.*∕ uciIndPerSlot : 2∕" $cuBB_SDK/cuPHY-CP/testMAC/testMAC/
  → test_mac_config.yaml
sudo -E ./cuphycontroller_scf nrSim_SCF
dudo ./test_mac nrSim 7600 --channels PUSCH
dudo ./ru_emulator nrSim 7600 --channels PUSCH
# Expect testMAC to report 100 PUSCH and 100 CSIP2 per second

1.2.4.5.3.5 PRACH

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
dudo ./test_mac nrSim 5901 --channels PRACH
dudo ./ru_emulator nrSim 5901 --channels PRACH
# Expect testMAC to report 100 Preambles per second

# PRACH 16 PID/Slot and PRACH B4 4FDM
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
dudo ./test_mac nrSim 5013 --channels PRACH
dudo ./ru_emulator nrSim 5013 --channels PRACH
Expect testMAC to receive 1600 Preambles per second

- Change tv_prach field as below in cuphycontroller_nrSim_SCF.yaml
  tv_prach: TVnr_5013_PRACH_gNB_CUPHY_s1p0.h5

Expect 4 RO occasions in each slot in phy.log in sequence mentioned in below.
RO 0 - PrmbIndex (2,5,8,11)
RO 1 - PrmbIndex (14,17,20,23)
RO 2 - PrmbIndex (32,35,26,29)
RO 3 - PrmbIndex (38,41,44,47)

# grep -i "RO\|prmbIndex" phy.log
15:57:41.161874 I [DRV.PRACH] RO 0 SFN 599.01 Preambles num detected 4
15:57:41.161874 I [DRV.PRACH] SFN 599.01 #0 prmbIndex 2 prmbDelay 0.000000
  → prmbPower -2.878487
15:57:41.161880 I [DRV.PRACH] SFN 599.01 #1 prmbIndex 5 prmbDelay 0.000000
  → prmbPower -2.801307
15:57:41.161883 I [DRV.PRACH] SFN 599.01 #2 prmbIndex 8 prmbDelay 0.000000
  → prmbPower -3.207683
15:57:41.161886 I [DRV.PRACH] SFN 599.01 #3 prmbIndex 11 prmbDelay 0.000000
  → prmbPower -3.423241
15:57:41.161901 I [DRV.PRACH] RO 1 SFN 599.01 Preambles num detected 4
1.2.4.5.3.6 NZP CSI_RS

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 4001 --channels CSI_RS
sudo ./ru_emulator nrSim 4001 --channels CSI_RS
# Expect RU Emulator to report 100 CSI_RS per second

1.2.4.5.3.7 PDSCH + ZP CSI_RS

To run TC 3323, 3338, and 3339, add --channels CSI_RS+PDSCH in the test_mac and ru_emulator commands.

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 3323 --channels CSI_RS+PDSCH
sudo ./ru_emulator nrSim 3323 --channels CSI_RS+PDSCH
# Expect RU Emulator to count 100 CSI_RS and 100 PDSCH per second
1.2.4.5.3.8 Precoding

# Below steps are applicable to precoding test for PDSCH, PDCCH, PBCH, and CSI_RS
# In l2_adapter_config_nrSim.SC.yml, set enable_precoding to 1
sed -i -z "s∕enable_precoding: 0∕enable_precoding: 1∕" $cuBB_SDK/cuPHY-CP/cuphycontroller/config/l2_adapter_config_nrSim.SC.yml
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim.SC
sudo ./test_mac nrSim 3248 --channels PDSCH
# Reset enable_precoding to 0
sed -i -z "s∕enable_precoding: 1∕enable_precoding: 0∕" $cuBB_SDK/cuPHY-CP/cuphycontroller/config/l2_adapter_config_nrSim.SC.yml
# In ru-emulator/config/config.yaml, set dl_approx_validation to 1
sed -i -z "s∕dl_approx_validation: 0∕dl_approx_validation: 1∕" $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml
sudo ./ru_emulator nrSim 3248 --channels PDSCH
# Expect testMAC and RU Emulator both see 1.36 Mbps 100 Slots per second
# Reset dl_approx_validation to 0
sed -i -z "s∕dl_approx_validation: 1∕dl_approx_validation: 0∕" $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml

1.2.4.5.3.9 PUCCH HARQ

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim.SC
sudo ./test_mac nrSim 6001 --channels PUCCH
sudo ./ru_emulator nrSim 6001 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per second

1.2.4.5.3.10 PUCCH Format 2

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim.SC
sudo ./test_mac nrSim 6201 --channels PUCCH
sudo ./ru_emulator nrSim 6201 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per second

1.2.4.5.3.11 PUCCH HARQ/SR

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim.SC
sudo ./test_mac nrSim 6049 --channels PUCCH
sudo ./ru_emulator nrSim 6049 --channels PUCCH
# Expect testMAC to report 300 HARQ + 300 SR and ru-emulator to report 100 PUCCH per second
1.2.4.5.3.12 PUCCH Format 3

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 6301 --channels PUCCH
sudo ./ru_emulator nrSim 6301 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per second

1.2.4.5.3.13 UCI on PUSCH

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7501
sudo ./ru_emulator nrSim 7501
# Expect testMAC to report 100 HARQ/s and UL slots/s

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7502
sudo ./ru_emulator nrSim 7502
# Expect testMAC to report 100 HARQ/s and UL slots/s

# Restart MPS
# UCI on PUSCH CSI part 2
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7517
sudo ./ru_emulator nrSim 7517 --channel PUSCH

For 7517-7519, 7524-26, 7528-29
# Expect testMAC to report 100 CSI part2/s and 100 UL slots/s
# Expect cuphycontroller to report 0 CRC for 100 slots/s and 1.61 Mbps UL throughput

For 7520-7523, 7527, 7530
# Expect testMAC to report 100 CSI part2/s
# Expect cuphycontroller to report 0 CRC for 100 slots/s

1.2.4.5.3.14 SRS

To enable FAPI 10.04 fields for the SRS test, add -DSCF_FAPI_10_04=ON in the cmake options and do a clean build. The test cases for SRS validation are 8301 and 8302.

In cuphycontroller_nrSim_SCF.yaml - enable_srs: 1

# Restart MPS
# Running 8301
sudo ./ru_emulator nrSim 8301 --channels SRS or ./ru_emulator nrSim 8301 (default support all channels)
sudo ./test_mac nrSim 8301 --channels SRS or ./test_mac nrSim 8301 (default support all channels)
sudo -E ./cuphycontroller_scf nrSim_SCF
# Expect the testMac to report the number of received SRS is between 97 and 103 and INV values per second to be 0.

(continues on next page)
# If the INV Values are greater than 0, there is either a SRS report mismatch or SRS
→ report parameter mismatch.

# Restart MPS
# Running 8302
sudo ./ru_emulator nrSim 8302 --channels SRS or ./ru_emulator nrSim 8302 (default
→ support all channels)
sudo ./test_mac nrSim 8302 --channels SRS or ./test_mac nrSim 8302 (default support
→ all channels)
sudo -E ./cuphycontroller_scf nrSim_SCF
# Expect the testMac to report the number of received SRS is between 97 and 103 and INV
→ values per second to be 0.
# If the INV Values are greater than 0, there is either a SRS report mismatch or SRS
→ report parameter mismatch.

1.2.4.5.3.15 S-slot

# Restart MPS
sudo ./ru_emulator nrSim 90013 --channels 0x1ff
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 90013 --channels 0x1ff
# Expect RU Emulator to report 50 DL and PDCCH_DL per second, testMAC to report 50 HARQ
→ per second

# Restart MPS
sudo ./ru_emulator nrSim 90015 --channels 0x1ff
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 90015 --channels 0x1ff
# Expect RU Emulator to report 50 DL and PDCCH_DL per second, testMAC to report 50 HARQ
→ per second

1.2.4.5.3.16 Multiple SSB

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 1104 --channels PBCH
sudo ./ru_emulator nrSim 1104 --channels PBCH
# Expect RU Emulator to report 100 PBCH per second

1.2.4.5.3.17 PUSCH TDI

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF_tdi
sudo ./test_mac nrSim 7411 --channels PUSCH
sudo ./ru_emulator nrSim 7411 --channels PUSCH
# Expect testMAC and RU Emulator both see 1.79 Mbps 100 Slots per second
### 1.2.4.5.3.18 PUSCH SINR and Noise

- **For TCs 7265, 7266, 7268, 7269, 7271, 7272**
  - Change `cuphycontroller_nrSim_SCF.yaml` file to have 8 eAxIds for PUSCH
  ```
  eAxC_id_pusch: [8,0,1,2,3,4,5,6]
  ```
  - Use `sed` to modify the file
  ```
  sed -i s/^eAxC_id_pusch: \[8,0,1,2\]$/eAxC_id_pusch: \[8,0,1,2,3,4,5,6\]$/
  ```
  - **For TCs 7264, 7267, 7270** no change to `cuphycontroller_nrSim_SCF.yaml`

- **Restart MPS**
  ```
  sudo ./test_mac nrSim 7265 --channels PUSCH
  sudo ./ru_emulator nrSim 7265 --channels PUSCH
  ```
  - Revert if changed earlier
  ```
  sed -i s/^eAxC_id_pusch: \[8,0,1,2,3,4,5,6\]$/eAxC_id_pusch: \[8,0,1,2\]$/
  ```

### 1.2.4.5.3.19 mSlot_mCell Test Cases

TCs 90001, 90002, 90003, 90004, 90005, 90006, 90011, 90012, 90013, 90014, 90015

- **nrSim config generation**
  ```
  cd ${cuBB_SDK}/cubb_scripts/autoconfig
  python3 auto_controllerConfig.py -i ../../testVectors/ -t  ../../../cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o  ../../../cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
  ```
  - **Auto RuEmulatorConfig**
  ```
  python3 auto_RuEmulatorConfig.py -i ../../../cuPHY-CP/cuphycontroller/config -t  ../../../cuPHY-CP/ru-emulator/config/config.yaml -o  ../../../cuPHY-CP/ru-emulator/config/config.yaml
  ```
  - **Backup default nrSim config**
  ```
  cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
  ```
  - **Copy nrSim_SCF_900xx config**
  ```
  cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
  ```
  - **Test Mac Config**
  ```
  python3 auto_TestMacConfig.py -t  ../../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml -c 900xx -p CG1 -o  ../../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
  ```
  - **Restart MPS**
  ```
  sudo . ./cuphycontroller_scf nrSim_SCF
  ```
  - **Restore nrSim config file**
  ```
  cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
  ```
1.2.4.5.3.20 TDD Pattern DSUUU

The test cases for uplink heavy TDD pattern (DSUUU) validation are 90061 to 90063.

In `cuphycontroller_nrSim_SCF.yaml` set `pusch_aggr_per_ctx` to 9, `prach_aggr_per_ctx` to 4 and `ul_input_buffer_per_cell` to 15

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 90061 --channels 0x1ff
sudo ./test_mac nrSim 90061 --channels 0x1ff
# Expect the testMac to report 800 PDSCH and 1600 PUSCH slots per second.
```

1.2.4.5.3.21 32T32R SRS + Dynamic Beamforming Weights Test Cases

Here are the steps to build and run the 32T32R SRS and dynamic beamforming weights related tests.

Build options:

```
cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native -DCSCF_FAPI_10_04=ON
cmake --build build
```

Verify all of the following launch patterns for DL-BFW+PDSCH and UL-BFW+PUSCH:

- TC's 100Mhz: 90070, 90073, 90074, 90079, 90082, 90083
- TC's 30Mhz: 90075, 90076
- TC's 50Mhz: 90077, 90078

The following are the TV's that you must uses for the above launch patterns:

100Mhz TV's:

- TVnr_9000_gNB_FAPI_s0.h5
- TVnr_9001_gNB_FAPI_s0.h5
- TVnr_9226_gNB_FAPI_s0.h5
- TVnr_9227_gNB_FAPI_s0.h5
- TVnr_3850_gNB_FAPI_s0.h5
- TVnr_3853_gNB_FAPI_s0.h5
- TVnr_7851_gNB_FAPI_s0.h5

30Mhz TV's:

- TVnr_9228_gNB_FAPI_s0.h5
- TVnr_9229_gNB_FAPI_s0.h5
- TVnr_3851_gNB_FAPI_s0.h5
- TVnr_7852_gNB_FAPI_s0.h5

50Mhz TV's:

- TVnr_9230_gNB_FAPI_s0.h5
- TVnr_9231_gNB_FAPI_s0.h5
For 32T32R SRS 84xx, the TV's need to be executed. You can generate the config using the autoconfig scripts for the above launch patterns, with the exception that only the following parameters need to be explicitly modified in the generated config file:

In `cuphycontroller_nrSim_SCF.yaml` - `enable_srs`: 1, `mMIMO_enable`: 1, `mtu`: 8192
In `ru-emulator: config.yaml` - `aerial_fh_mtu`: 8192

```
# nrSim config generation
cd ${cubb_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../../testVectors/ -t ../../../cuPHY-CP/
    cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../../cuPHY-CP/
    cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../../../cuPHY-CP/cuphycontroller/config -t ../../../cuPHY-CP/cuphycontroller/config/config.yaml -o ../../../cuPHY-CP/cuphycontroller/config

# backup default nrSim config
cp ${cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cubb_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cubb_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig
cp ${cubb_SDK}/cuPHY-CP/testMAC/testMAC/config/config.yaml ${cubb_SDK}/cuPHY-CP/testMAC/testMAC/config/config.yaml.orig

# Use nrSim_SCF_900xx config
cp ${cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml ${cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cubb_SDK}/cuPHY-CP/ru-emulator/config/ru_simulator_config_900xx.yaml ${cubb_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
python3 auto_TestMacConfig.py -t ../../../cuPHY-CP/testMAC/testMAC/config/config.yaml.orig -c 900xx -p CG1 -o ../../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 900xx --channels 0x7ff
sudo ./test_mac nrSim 900xx --channels 0x7ff

# Restore nrSim config file
cp ${cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig ${{cubb_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cubb_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${{cubb_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
cp ${cubb_SDK}/cuPHY-CP/testMAC/testMAC/testMAC/test_mac_config.yaml.orig ${{cubb_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```
Here are the steps to build and run the 64T6R SRS and dynamic beamforming weights related tests.

Build options:
```
cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native -DCSF_FAPI_10_...04=ON
cmake --build build
```

Verify all of the following launch patterns for DL-BFW+PDSCH and UL-BFW+PUSCH:
- TC's 100Mhz: 90090, 90091, 90092, 90093, 90094, 90095, 90096, 90097, 90098, 90099, 90100, 90101, 90102

The following are the TV's that you must uses for the above launch patterns:

100 MHz DL 16 layer (16 UE's) full allocation:
- TVnr_9236_gNB_FAPI_s0.h5
- TVnr_3870_gNB_FAPI_s0.h5

100 MHz UL 8 layer (8 UE's) full allocation:
- TVnr_9237_gNB_FAPI_s0.h5
- TVnr_7870_gNB_FAPI_s0.h5

100 MHz DL 1 layer (1 UE) full allocation:
- TVnr_9238_gNB_FAPI_s0.h5
- TVnr_3871_gNB_FAPI_s0.h5

100 MHz UL 1 layer (1 UE) full allocation:
- TVnr_9239_gNB_FAPI_s0.h5
- TVnr_7871_gNB_FAPI_s0.h5

100 MHz DL 2 layers (1 UE) full allocation:
- TVnr_9240_gNB_FAPI_s0.h5
- TVnr_3872_gNB_FAPI_s0.h5

100 MHz UL 2 layers (1 UE) full allocation:
- TVnr_9241_gNB_FAPI_s0.h5
- TVnr_7872_gNB_FAPI_s0.h5

100 MHz DL 1+1 layers (2 UE's) full allocation:
- TVnr_9242_gNB_FAPI_s0.h5
- TVnr_3873_gNB_FAPI_s0.h5

100 MHz UL 1+1 layers (2 UE's) full allocation:
- TVnr_9243_gNB_FAPI_s0.h5
- TVnr_7873_gNB_FAPI_s0.h5

100 MHz DL 2+2 layers (2 UE's) full allocation:
- TVnr_9244_gNB_FAPI_s0.h5
For 64T64R SRS 85xx, the TV’s need to be executed. You can generate the config using the autoconfig scripts for the above launch patterns, with the exception that only the following parameters need to be explicitly modified in the generated config file:

In cuphycontroller_nrSim_SCF.yaml - enable_srs: 1, mMIMO_enable: 1, mtu: 8192
In ru-emulator: config.yaml - aerial_fh_mtu: 8192

# nrSim config generation
cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../..//testVectors/ -t ../..//cuPHY-CP/
--cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../..//cuPHY-CP/
--cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../..//cuPHY-CP/cuphycontroller/config -t ../..//
--cuPHY-CP/ru-emulator/config/config.yaml -o ../..//cuPHY-CP/ru-emulator/config

# backup default nrSim config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
--emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
--testMAC/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_900xx config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml $${cuBB_}
--SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml
python3 auto_TestMacConfig.py -t ../..//cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.
--orig -c 900xx --p CG1 -o ../..//cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 900xx --channels 0x7ff
sudo ./test_mac nrSim 900xx --channels 0x7ff

(continues on next page)
### FAPI Message Reference Check

The cuBB software supports the FAPI message reference check. The values and payloads of RX_DATA.ind, CRC.ind, UCI.ind, and RACH.ind are compared with the related INDx PDU of the TV. If validation fails, a “mismatch” WARN level log is printed to testmac.log by testMAC.

**Note:** Some validation failures are not fixed yet. The current known validation failures are not reported with “INV > 0” by default.

The following configurations are implemented to configure test_mac reporting. The default configuration for FAPI validation is as follows:

```plaintext
# FAPI indication validating
# validate_enable: 0 - disabled; 1 - report error level; 2 - report error and warning levels
validate_enable: 1
# validate_log_opt: 0 - no print; 1 - print per MSG; 2 - print per PDU; 3 - force print all
validate_log_opt: 1
```

The following is an example validation failure log with default configuration:

```
09:35:02.205513 W [MAC.FAPI] SFN 0.5 Cell 6 CRC.ind mismatch: 0 err 6 warn [crc->num_cb=192 tv.NumCb=4] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=880]
```

One FAPI message can may contain multiple PDUs, and one PDU can contain multiple validation failures.

- Set “validate_enable: 1” to report only some validation failures with “INV > 0” in test_mac console. Known validation failures are not reported with “INV > 0” (but can still be seen in the “mismatch” WARN log).
- Set “validate_enable: 2” to report all validation failures with “INV > 0” in test_mac console.
- Set “validate_log_opt: 1” to print one line “mismatch” log with at most three mismatched values per FAPI message, and print the total mismatched PDU count (e.g. “0 err, 6 warn”) per FAPI message (avoids performance dropping).
- Set “validate_log_opt: 2” to print all validation failures in the “mismatch” WARN log, one line per PDU.

Example log with “validate_log_opt: 2”:

```
07:32:09.407972 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind PDU0 mismatch: [crc->num_cb=0 tv.NumCb=5] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=1280]
07:32:09.407976 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind PDU1 mismatch: [crc->num_cb=0 tv.NumCb=5] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=1280]
```
The current recommended test instructions:

▶ Use the default configuration to test, then grep “mismatch” in phy.log to check whether there is a validation failure.
▶ Configure “validate_log_opt: 2” to print all validation failures, if required.

1.2.4.5.5 Running testMAC + cuPHYController_SCF + RU Emulator P5G PRACH

This use case runs the Private 5G SIB1 and PRACH demo msg1-4 between the RU Emulator and the testMAC.

You need additional modifications to the default cuPHYController_P5G.yaml to test against RU emulator. Ensure it matches the configs here. You must also set the PCIe NIC address that is currently in use:

1.2.4.5.5.1 Server#1

```
sed -i "s/nic:.*/nic: 0000:b5:00.0/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G.yaml`
sed -i "s/dl_iq_data_fmt.*/dl_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G.yaml`
sed -i "s/ul_iq_data_fmt.*/ul_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G.yaml`
sed -i "s/ppc.*/ppc: 7/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G.yaml`
sed -i "0,/dst_mac_addr.*/{s/dst_mac_addr.*/dst_mac_addr: 20:04:9B:9E:27:A3/}" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G.yaml`
sed -i "s/enableTickDynamicSfnSlot.*/enableTickDynamicSfnSlot: 0/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/12_adapter_config_P5G.yaml`
sed -i "s/enableTickDynamicSfnSlot.*/enableTickDynamicSfnSlot: 0/" `${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/12_adapter_config_P5G_R750.yaml`
```

1.2.4.5.5.2 Server#2

Replace 0000:b5:00.0 with the PCIe address of the NIC to use. Also, replace the MAC address with the MAC address of the NIC used in Server#1 (the server running cuPHYController and testMAC):

```
sed -i "s/nic_interface.*/nic_interface: 0000:b5:00.0/" `${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml`
```

Change the dl_iq_data_fmt/ul_iq_data_fmt to BFP 16. Ensure you change it back to BFP 14 for other tests.

```
sed -i "s/dl_iq_data_fmt.*/dl_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" `${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml`
sed -i "s/ul_iq_data_fmt.*/ul_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" `${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml`
```

(continues on next page)
Run the emulator:

```bash
sudo ./ru_emulator P5G PRACH --channels 0x1FF
```

Run the cuPHY controller and the testMAC:

```bash
sudo -E ./cuphycontroller_scf P5G
dsudo ./test_mac P5G PRACH --channels 0x1FF
```

Expected RU emulator console:

<table>
<thead>
<tr>
<th>Time</th>
<th>Cell</th>
<th>DL</th>
<th>UL</th>
<th>PBCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:44:12.169849</td>
<td>0</td>
<td>0.17 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:13.169848</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:14.169849</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
</tbody>
</table>

Expected testMAC console:

<table>
<thead>
<tr>
<th>Time</th>
<th>Cell</th>
<th>DL</th>
<th>UL</th>
<th>PBCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:44:11.565232</td>
<td>0</td>
<td>0.26 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:12.565230</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:13.565230</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
</tbody>
</table>

Expected cuPHYController logs to be flooded with preamble detection:

<table>
<thead>
<tr>
<th>Time</th>
<th>Cell</th>
<th>DL</th>
<th>UL</th>
<th>PBCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:44:11.565224</td>
<td>0</td>
<td>0.26 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:12.565224</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
<tr>
<td>00:44:13.565224</td>
<td>0</td>
<td>0.03 Mbps</td>
<td>50 Slots</td>
<td>50 Slots</td>
</tr>
</tbody>
</table>

### 1.2.4.5.6 Running End-to-End with Full Stack

This section provides a guide on reference cuPHYController YAML to be used when using Aerial CUDA-Accelerated RAN with Full Stack application.

When running full stack Aerial CUDA-Accelerated RAN on Aerial Devkit, use the following files as a starting point to be modified according to your lab configuration.

1. When using Keysight RU-SIM as a Radio Unit, use `cuphycontroller_P5G.yaml` as a reference.
2. When using Foxconn O-RU as a Radio Unit, use `cuphycontroller_P5G_FXN.yaml` as a reference.
When running full stack Aerial CUDA-Accelerated RAN on Dell R750, use the following files as a starting point to be modified according to your lab configuration.

1. When using Keysight RU-SIM as a Radio Unit, use `cuphycontroller_P5G_R750.yaml` as a reference.
2. When using Foxconn O-RU as a Radio Unit, use `cuphycontroller_P5G_FXN_R750.yaml` as a reference.

When running full stack Aerial CUDA-Accelerated RAN on Grace Hopper, use the following file as a starting point to be modified according to your lab configuration.

1. When using Keysight RU-SIM as a Radio Unit, use `cuphycontroller_P5G_GH.yaml` as a reference.
2. When using Foxconn O-RU as a Radio Unit, use `cuphycontroller_P5G_FXN_GH.yaml` as a reference.

Note: You need to modify the above mentioned reference files based on your End-to-End setup.

### 1.2.4.5.7 Capture Logs

Collect the text logs after testing.

1. By default, the logs get stored in the `/tmp` location. You can use the `AERIAL_LOG_PATH` environment variable to set the logfile path.
2. When the log size exceeds 50GB, a new file gets created (e.g. `phy.log`, `phy.log.1`, `phy.log.2` ... `phy.log.7`).
   a. The test MAC logs are named as `testmac.log`, `testmac.log.1`, etc.
   b. The RU logs are named as `ru.log`, `ru.log.1`, etc.
3. These file segments are reused in a cyclic manner by overwriting the oldest files.

For SHM IPC, if you see the IPC buffer pool full during testing, run the following command to dump IPC status after test:

```bash
# For SHM IPC, dump nvipc message queues after test
sudo ./build/cuPHY-CP/gt_common_libs/nvIPC/tests/dump/ipc_dump

# If not using default nvipc configurations, need input the nvipc "prefix" and yaml config file like below.
# For Multi-L2 case, the "prefix" names are different for each L2 instance, see related nvipc config yaml files.
sudo ./build/cuPHY-CP/gt_common_libs/nvIPC/tests/dump/ipc_dump nvipc ./cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08.yaml
```

To capture PCAP log, please run "export NVIPC_DEBUG_EN=1" in command line of cuphycontroller or config below pcap_enable=1. Max size limitation of PCAP logs can be configured in the yaml file.

```bash
# Transport settings for nvIPC
transport:
  type: shm
  app_config:
    pcap_enable: 1
    pcap_cpu_core: -1 # CPU core of background pcap log save thread
```

(continues on next page)
pcap_cache_size_bits: 29 \# 2^{29} = 512MB, size of /dev/shm/${prefix}_pcap
pcap_file_size_bits: 31 \# 2^{31} = 2GB, max size of /dev/shm/${prefix}_pcap
pcap_max_data_size: 8000 \# Max DL/UL FAPI data size to capture reduce pcap size.

After cuphycontroller exits, run below command with nvipc "prefix" to collect PCAP logs. The "prefix" comes from l2adapter config yaml. It is "nvipc" by default. In Multi-L2 case, different L2 instance has different "prefix" names.

# Usage: sudo ./pcap_collect <prefix> [destination path]
sudo $cuBB_SDK/build/cuPHY-CP/gt_common_libs/nvIPC/tests/pcap/pcap_collect nvipc

# The nvipc.pcap can be seen at current directory (by default) or the inputed "destination path".

1.2.4.6 Run in Test Mode (TM)

To run any test where at least one cell is in Test Mode (TM), you need to ensure cmake was run with -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON.

This option is needed for nrSIM TCs 2036 (PDCCH) and 3296 (PDSCH), as well as for DLMIX TCs 120-128 with both PDSCH and PDCCH present. This requirement extends to any multi-cell launch pattern with at least one of these TM test vectors present.

For test cases where no TM cell is present, the cmake option value is not relevant for the functional correctness of cuBB tests.

1.2.4.7 Mixed O-RAN IOT Profiles (CAT-A-NoBF + CAT-A-DBF)

To run mixed one cell with CAT-A-NoBF and another cell with CAT-A-DBF, use the nrSIM TC 90019 and run the following:

# nrSim config generation

cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../../cuPHY-CP/ --cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../cuPHY-CP/ --cuphycontroller/config

python3 auto_RuEmulatorConfig.py -i ../../cuPHY-CP/cuphycontroller/config -t ../../cuPHY-CP/ --cuPHY-CP/ru-emulator/config/config.yaml -o ../../cuPHY-CP/ru-emulator/config

# backup default nrSim config

cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig

cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig

cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_90019 config

cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_90019.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_90019.yaml

cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml

python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml -c 90019 -p CG1 -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
# Restart MPS

```
sudo -E ./cuphycontroller_scfd nrSim_SCF
sudo ./ru_emulator nrSim 90019 --channels 0x1ff
sudo ./test_mac nrSim 90019 --channels 0x1ff
```

# Restore nrSim config file

```
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml

cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```

Expected result:

# Expected Tput and passing criteria

- Expected thrput: Cell 0: [DL=1.36/100]
- Expected thrput: Cell 1: [DL=2.72/100]
- Pass criterion low: Cell 0: [DL=1.31/97]
- Pass criterion high: Cell 0: [DL=1.40/103]
- Pass criterion low: Cell 1: [DL=2.63/97]
- Pass criterion high: Cell 1: [DL=2.80/103]

# Example ru-emulator output

```
16:24:15.218189 Cell 0 DL 1.36 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C
   --ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% | Seconds 45
16:24:15.218201 Cell 1 DL 2.72 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C
   --ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% | Seconds 45
16:24:16.218191 Cell 0 DL 1.36 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C
   --ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% | Seconds 46
16:24:16.218204 Cell 1 DL 2.72 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C
   --ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% | Seconds 46
```

1.2.4.8 Mixed BFP9/BFP14

# nrSim config generation

```
# nrSim config generation

cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../cuPHY-CP/
   --cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../cuPHY-CP/
   --cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../cuPHY-CP/cuphycontroller/config -t ..
   --cuPHY-CP/ru-emulator/config/config.yaml -o ../cuPHY-CP/ru-emulator/config
```

# backup default nrSim config

```
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
   -emulator/config/config.yaml.orig

cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
   --testMAC/testMAC/test_mac_config.yaml.orig
```

# Use nrSim_SCF_90020 config

```
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_90020.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
```
Example testMAC output

Expected slots: Cell=0 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0
  CSI_RS=0 SRS=0

Expected data: Cell=0 DL=0.000000 UL=41.797600 Prmb=0 HARQ=0 SR=0 CSI1=0 CSI2=0 ERR=0
  INV=0

Expected slots: Cell=1 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0
  CSI_RS=0 SRS=0

Expected data: Cell=1 DL=0.000000 UL=41.797600 Prmb=0 HARQ=0 SR=0 CSI1=0 CSI2=0 ERR=0
  INV=0

Example testMAC output

<table>
<thead>
<tr>
<th>Time</th>
<th>Cell</th>
<th>DL</th>
<th>UL</th>
<th>PRMB</th>
<th>HARQ</th>
<th>SR</th>
<th>CSI1</th>
<th>CSI2</th>
<th>ERR</th>
<th>INV</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:09:34.600006</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>100 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>Slots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:34.600006</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:35.600006</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:35.600014</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:36.600006</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:36.600013</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:37.600008</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:37.600017</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:38.600006</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:38.600014</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:39.600008</td>
<td>0</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:09:39.600017</td>
<td>1</td>
<td>0.00 Mbps</td>
<td>0 Slots</td>
<td>UL</td>
<td>41.00 Mbps</td>
<td>100 Slots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 1.2.4.9 Mixed IQ data format for F08 Test Case

Here is an example to run the mixed compression using F08 test case for 1 16-bit Fixed point + 1 BFP9 + N BFP 14 cells, where N = 1, 2, 3. Set the value for `dl_iq_data_fmt` and `ul_iq_data_fmt` to 16-bit fixed point for the 1st cell and BFP 9 for the 2nd cell in both the `cuPHYController_F08_*`.yaml file and RU emulator config.yaml file. Set the value for `dl_iq_data_fmt` and `ul_iq_data_fmt` to BFP 14 for all other cells.

```yaml
# First cell
dl_iq_data_fmt: {comp_meth: 0, bit_width: 16}
ul_iq_data_fmt: {comp_meth: 0, bit_width: 16}

# Second cell
dl_iq_data_fmt: {comp_meth: 1, bit_width: 9}
ul_iq_data_fmt: {comp_meth: 1, bit_width: 9}

# All other cells
dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
```

The throughput levels must be the same as non-mixed case with only BFP 14.

```
16:17:05.609792 C [SCF.PHY] Cell 0 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
-400 Slots CRC 0 ( 0 ) | Tick 16000
16:17:05.609808 C [SCF.PHY] Cell 1 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
-400 Slots CRC 0 ( 0 ) | Tick 16000
16:17:05.609814 C [SCF.PHY] Cell 2 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
-400 Slots CRC 0 ( 0 ) | Tick 16000
16:17:05.609822 C [SCF.PHY] Cell 3 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
-400 Slots CRC 0 ( 0 ) | Tick 16000
```

### 1.2.4.10 Displaying PHY Processing Latencies

To extract the latencies, run the test with `shm_log_level: 5`:

```
nvlog:
name: phy
shm_log_level: 5 # SHM log level
```

It is possible to parse the cuphycontroller PHY log (`cuphycontroller PHY.log`) to get average latencies (with standard deviation) of key cuphydriver and aerial-fh tasks:

```
pip3 install matplotlib
python3 ${cuBB_SDK}/cuPHY-CP/aerial-fh-driver/scripts/phy_latencies/phy_latencies.py --phy.log -a
```
1.2.4.10.1 Aerial-FH Latencies

- **C-plane**: Preparing all C-plane packets for a slot and sending them.
- **U-plane prepare**: Preparing all U-plane packets for a slot.
- **U-plane TX**: Sending all U-plane packets for a slot.
- **U-plane poll TX complete**: Checking if previous U-plane TX has completed.
  - Because of accurate TX scheduling, packets are not sent immediately.
  - To reuse GPU buffers, there must be verification of when the U-plane packets got sent.
  - U-plane TX does 'completions polling' as well.
- **U-plane RX**: Receiving all U-plane packets for a slot.
- **U-plane Free**: Freeing aerial-fh data structures used for U-plane RX.

1.2.4.11 UL Measurements

To enable UL measurements in PHY, set the the following to 1 in `cuphycontroller_nrSim_SCF.yaml` and all measurements are in dBm unit.

```yaml
pusch_sinr: 1  # 0 - Disabled; 1 - PostEq value; 2 - PreEq value
pusch_rssi: 1
pusch_tdi: 1
pusch_cfo: 1
```

**Note:** From release 22-2.3 onwards, SINR reporting can be configured to report pre- or post-equalizer values from the `cuphycontroller_nrSim_SCF.yaml` file, as shown above.

To enable FAPI 10.04 fields, add `-DSCF_FA10_04=ON` in the cmake options and do a clean build.

To enable RSSI and RSRP measurements, L2 has to send Measurement Config TLV in `config.request` with a value of 1 for dBm as described in table 3-27 of FAPI 10.02 and table 3-48 of FAPI 10.04. To enable the same in `testMac`:

- RSSI is enabled by default.
- For RSRP, set the following to 1 in the `$cuBB_SDK/testMAC/testMAC/test_mac_config.yaml` file:

```yaml
rsrpMeasurement: 1
```

L2 vendors have requested additional interference level reporting for PUSCH, UCI on PUSCH, and PUCCH (PF2,3 only supported). For this purpose, vendor specific messages have been defined to indicate the Aerial instance that reports these measurements. To enable this reporting, L2 has to send 2 additional TLVs in `config.request` as mentioned in `CONFIG.request`.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA012</td>
<td>PNMeasurement</td>
<td>uint8_t</td>
<td>Post equalisation noise variance measurement Value: 0: Do not report 1: dBm</td>
</tr>
<tr>
<td>0xA014</td>
<td>PF_234_Interference</td>
<td>uint8_t</td>
<td>Interference power per UE. Value: 0: Do not report 1: dBm</td>
</tr>
</tbody>
</table>
After it is enabled, for every CRC.indication, Aerial sends an additional RX_PE_Noise_Variance.indication. For every UCI.indication carrying PF2,3, Aerial sends a PF_234_Interference.indication.

To enable interference reporting in testMac, set the following to 1 in the $cuBB_SDK/testMAC/test_mac_config.yaml file:

```yaml
pf_234_interference: 1
pnMeasurement: 1
```

Enable DEBUG level log for tag SCF.PHY as follows in the cuPHY/nvlog/config/nvlog_config.yaml file:

```yaml
- 333: "SCF.PHY"
  shm_level: 6  # Example: overlay shm_log_level for a tag
```

The following example shows results in the phy.log log:

```
  dbm 733 numMeasurements 1
05:22:56.350664 I [MAC.SCF] SFN 375.0 <<< SCF_FAPI_RX_PF_234_INTEFERNCE_INDICATION:
  num=0 meas=733
```

For DTX detection for UCI on PUSCH, look for “detection status”. Sample below.

```
03:39:11.983670 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: HARQ detection status 4
  detection status 4 CSI P1 bit len 10
```

### 1.2.4.11.1 Verification of PUSCH Measurement Reporting for BFP-9/14/16

Change the value of BFP in the matlab file 5GModel/nr_matlab/config. Generate cuPHY and FAPI TV and run the test.

```bash
% BFP setting for cuPHY UL TV generation and UL performance simulation
SimCtrl.BFPforCuphy = 16; % 16, 14 or 9 for FP16, BFP14 or BFP9
```

Set log level to 6 and take h5dump of IND1 in FAPI TV.

```
h5dump -d IND1 TVnr_7427_gNB_FAPI_s0.h5_BFP9
```

Compare RSSI, RSRP, sinrdB and noiseVar values against the FAPI values in logs -

```bash
>>> SCF_FAPI_CRC_INDICATION 10.04 ul-sinr=20000 ta=63 ta-ns=16803 rssi=853 rsrp=912
[SCF.PHY] >>> RX_PE_NOISE_VARIANCE_INDICATION: PHY sfn=0x153 slot=0x0 num_meas=1
  meas[0]=633
```
For comparing the raw values of UL measurements against the TV, take the h5dump of the following values from cuPHY TV. Then compare reference_sinrdB, reference_rssi, reference_rsrpdB, and reference_noiseVardB values against the raw values in logs.

```
```

### 1.2.4.11.2 Verification of PUCCH Measurement Reporting for BFP-9/14/16

Change the value of BFP in the 5GModel/nr_matlab/config matlab file. Generate cuPHY and FAPI TV and run the test.

```
% BFP setting for cuPHY UL TV generation and UL performance simulation
SimCtrl.BFPforCuphy = 16; % 16, 14 or 9 for FP16, BFP14 or BFP9
```

Set log level to 6.

Match the value in logs against these fields in cuPHY TV.

Format 0 - FOUCisOutRef. Compare RSSI and RSRP values against corresponding value in logs.

```
SCF_FAPI_UCI_INDICATION 10.04: PUCCH : Raw SINR=0.000000 RSSI=16.824944 RSRP=10.0804343
```

Format 1 - F1UcisOutRef. Compare “SinrDB”, “RSSI”, and “RSRP” values against the corresponding value in logs.

```
```

Format 2/3 - pucchF234_refSnrBuffer, pucchF234_refRsrpBuffer, pucchF234_refRssiBuffer, and pucchF234_refInterfBuffer. Compare them against relevant values in logs.

```
[SCF.PHY] Raw SINR=28.154160 RSRP=-0.132790 ul_configured_gain=48.680000
[SCF.PHY] Raw RSSI=5.887811 ul_configured_gain=48.680000
>>> SCF_FAPI_UCI_INDICATION: PUCCH interference Raw=-28.286949 dbm 750
    --numMeasurements 1
```

### 1.2.4.11.3 Verification of PRACH Interference Level Report for BFP-9/14/16

Enable config in test_mac_config.yaml.

```
prach_interference: 1
```

Run the nrSim 5013 test.

```
nrSim 5013 --channels PRACH
```

Get “nOcc=x Raw PRACH interference” (x=0~3) from phy.log and get “PDUx_noise” (x=1~4) from TV:
Aerial CUDA-Accelerated RAN, Release 24-2

# phy.log

grep -o "PHY nOcc=[0-9] Raw PRACH interference=.*" phy.log
PHY nOcc=0 Raw PRACH interference=-16.046867 ul_configured_gain=48.680000 FAPI
→ value=872
PHY nOcc=1 Raw PRACH interference=-16.921370 ul_configured_gain=48.680000 FAPI
→ value=863
PHY nOcc=2 Raw PRACH interference=-17.524746 ul_configured_gain=48.680000 FAPI
→ value=857
PHY nOcc=3 Raw PRACH interference=-18.472067 ul_configured_gain=48.680000 FAPI
→ value=848

# TV

h5ls -ld TVnr_5013_gNB_FAPI_s1.h5∕PDU1_noise TVnr_5013_gNB_FAPI_s1.h5∕PDU2_noise TVnr_
→ 5013_gNB_FAPI_s1.h5∕PDU3_noise TVnr_5013_gNB_FAPI_s1.h5∕PDU4_noise

PDU1_noise Dataset {1, 1}
  Data: (0, 0) -16.0463

PDU2_noise Dataset {1, 1}
  Data: (0, 0) -16.0842

PDU3_noise Dataset {1, 1}
  Data: (0, 0) -16.0449

PDU4_noise Dataset {1, 1}
  Data: (0, 0) -16.294

Expected result:

abs(Raw PRACH interference - PDUx_noise) < 3
# Example, in above log the 4th occasion: abs(-18.472067 + 16.294) = 2.178067 < 3

1.2.4.12 Cell Life-Cycle Test

To restart all cells while multiple cells are running

In the test_mac_config.yaml file, set the following:

# testMAC/test_mac_config.yaml

# Total slot number in test
test_slots: 8000  # When 1 slot = 0.5 ms, 8000 slots = 4 seconds.
# Restart interval after test_slots finished. Unit is second
restart_interval: 5

This instructs the testMAC to schedule 8000 slots then send cell stop request to all cells. After waiting
5 seconds, TestMAC sends a config request and cell start request to all cells.

Use the following commands to verify with the F08 4C pattern A case. The expected result is full
throughput runs for approximately 4 seconds, test_mac throughput stops, and ru-emulator through-
put reduces to 0 for about 5 seconds, then the procedure repeats.

sudo ./cuPHY-CP/ru-emulator/ru_emulator F08 4C 60
sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08
sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 4C 60

1.2. cuBB Quickstart Guide
1.2.4.13 Terminate cuphycontroller Using a gRPC Message

Run the F08 E2E test case as usual:

```
sudo -E ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08
sudo ${cuBB_SDK}/build/cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 1C 60
sudo ${cuBB_SDK}/build/cuPHY-CP/testMAC/testMAC/test_mac F08 1C 60
```

Terminate cuphycontroller while the E2E test is running:

```
cd ${cuBB_SDK}/build/cuPHY-CP/cuphyoam
python3 ../../../cuPHY-CP/cuphyoam/examples/aerialTerminate_cuphycontroller.py
```

Verify that the cuphycontroller stops running, and that `aerialTerminate_cuphycontroller.py` prints the following output:

```
12:23:32 Terminating cuphycontroller...
12:23:36 cuphycontroller terminated successfully!
```

1.2.4.14 Update M-plane Parameters Using gRPC Message

Dynamically changing M-plane parameters via gRPC message are often used with cell life during the initial cell setup with RUs, as well as to replace the RU while cells are running. See List of parameters supported by dynamic OAM via gRPC and CONFIG.request (M-plane).

The following sequence diagram shows an example of both scenarios:

- **Initial cell and M-plane setup:** After launching cuphycontroller, L1 is initialized and all cells are in idle state to be configured. The max number of cells is defined by the `cell_group_num` parameter in the cuphycontroller YAML config. In the example sequence diagram, the OAM sends a gRPC message to update M-plane parameters so that L1 gets the details to connect to the right RU for each cell. Then L2 sends `CONFIG.request` to configure the cell.

  **Note:** In the current implementation, all cells must be configured before any cell `Start.request`.

- **RU replacement while other cells are running:** The example sequence diagram shows the sequence to move the cell-1 traffics from RU1 to RU5. Firstly, the L2 must stop scheduling traffics on cell-1 and send cell `Stop.request` to cell-1. After that, OAM sends the new M-plane parameters via gRPC message for L1 to connect to RU5. Then L2 sends Config.request and Start.request to bring cell-1 to a running state again.

  **Note:** In the current implementation, the cell `Config.request` after the first cell `Start.request` has no effect.
1.2.4.14.1 X2 Launch Pattern Files Generation

In subsequent sections, X2 launch pattern files are needed for a related test. The $cuBB_SDK/cuPHY-CP/cuphyoam/examples/launch_pattern_x2_update.py script is used to generate them.

Here is the usage:

```
usage: launch_pattern_x2_update.py [-h] -f LAUNCH_PATTERN_FILE -o OUTPUT_DIR
launch_pattern_x2_update.py: error: the following arguments are required: -f/--launch_pattern_file, -o/--output_dir
```

For example, to generate the X2 launch pattern file for TC “F08 2C 59”, run the following in container. This generates the corresponding `$cuBB_SDK/testVectors/multi-cell/launch_pattern_F08_2C_59_X2.yaml` file.

```
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/launch_pattern_x2_update.py -f $cuBB_SDK/testVectors/multi-cell/launch_pattern_F08_2C_59.yaml -o $cuBB_SDK/testVectors/multi-cell/
```

1.2.4.14.2 Initial OAM Update

Here is an example of a 4 cell test. Run cuphycontroller with the wrong initial configurations, then use the gRPC message to update them to the right values.

1.2.4.14.2.1 DST MAC Address OAM Initial Update Test - Single Cell

Update configs:

```
# Update `cell_group_num` to 1 in cuphycontroller yaml config
sed -i "s/cell_group_num.*/cell_group_num: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml

# Use below settings for "Cell1" in ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml, note that only the eth mac address is changed cell_configs:
```

```
- name: "Cell1"
  eth: "20:04:9B:9E:27:B3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
```

If you don’t perform the OAM update to change the cell 1 destination MAC address, the following test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```
If you perform the OAM update to change the cell 1 destination MAC address, the following test passes. The expected test result is throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08
# Below OAM update command should be executed on the same server as cuphycontroller
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

1.2.4.14.2.2 VLAN ID OAM Initial Update Test - Single Cell

**Update configs:**

```
# Update 'cell_group_num' to 1 in cuphycontroller yaml config
sed -i "s[cell_group_num].*/cell_group_num: 1/" $cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml
# Use below settings for "Cell1" in $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml, note that only the VLAN ID is changed
cell_configs:
    -
      name: "Cell1"
      eth: "20:04:9B:9E:27:A3"
      eAxC_UL: [8,0,1,2]
      eAxC_DL: [8,0,1,2]
      eAxC_prach_list: [15,7,0,1]
      dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
      ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
      peer: 0
      nic: 0
      vlan: 3
      pcp: 7
```

If you don’t perform the OAM update to change the cell 1 VLAN ID, the following test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

If you perform the OAM update to change the cell 1 VLAN ID, the following test passes. The expected test result is throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08
# Below OAM update command should be executed on the same server as cuphycontroller
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```
1.2.4.14.2.3 VLAN PCP OAM Initial Update Test - Single Cell

Update configs:

```bash
# Update 'cell_group_num' to 1 in cuphycontroller yaml config
sed -i "s/\(cell_group_num\).*/cell_group_num: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
   --config/cuphycontroller_F08_*.yaml
# Use below settings for "Cell1" in ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml,
   → note that only the PCP is changed cell_configs:
   -
      name: "Cell1"
      eth: "20:04:9B:9E:27:A3"
      eAxC_UL: [8,0,1,2]
      eAxC_DL: [8,0,1,2]
      eAxC_prach_list: [15,7,0,1]
      dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
      ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
      peer: 0
      nic: 0
      vlan: 2
      pcp: 4
```

If you don't perform the OAM update to change the cell 1 PCP, the following test fails. The expected test result is no throughput on the ru-emulator side.

```bash
sudo -E ./cuphycontroller_scf F08
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

If you perform the OAM update to change the cell 1 PCP, the following test passes. The expected test result is throughput on the ru-emulator side.

```bash
sudo -E ./cuphycontroller_scf F08
# Below OAM update command should be executed on the same server as cuphycontroller
cd ${cuBB_SDK}/build/cuPHY-CP/cuphyoam && python3 ${cuBB_SDK}/cuPHY-CP/cuphyoam/examples/
   aerial_cell_param_net_update.py 1 20:04:9B:9E:27:A3 8002
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

1.2.4.14.2.4 DST MAC + VLAN ID + PCP OAM Initial Update Test - Multi-Cells

```bash
# Update 'cell_group_num' to 4 in cuphycontroller yaml config
sed -i "s/\(cell_group_num\).*/cell_group_num: 4/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
   --config/cuphycontroller_F08_*.yaml
# Change eth, vlan, pcp of Cell 1~4 to any wrong values (here only show the values
   → which require change) in ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_/
   →F08_*.yaml
   -
      cell_id: 1
      vlan: 3
      pcp: 2
   -
      cell_id: 2
```
(continues on next page)
If you don’t perform the OAM update, the E2E test fails. The expected test result is no throughput on the ru-emulator side.

```bash
sudo -E ./cuphycontroller_scf F08
sudo ../ru_emulator F08 4C 59
sudo ../test_mac F08 4C 59
```

If you perform the OAM update for MAC + VLAN + PCP of the 4 cells to correct values, the E2E test passes. The expected test result is normal throughputs for all cells.

```bash
sudo -E ./cuphycontroller_scf F08
# OAM update MAC + VLAN + PCP of the 4 cells after cuphycontroller_scf started
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
  aerial_cell_param_net_update.py 1 20:04:9B:9E:27:A3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
  aerial_cell_param_net_update.py 2 26:04:9D:9E:29:B3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
  aerial_cell_param_net_update.py 3 20:34:9A:9E:29:B3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
  aerial_cell_param_net_update.py 4 22:34:9C:9E:29:A3 E002
sudo ../ru_emulator F08 4C 59
sudo ../test_mac F08 4C 59
```

### 1.2.4.14.3 Dynamic OAM Update

#### 1.2.4.14.3.1 DST MAC Address OAM On-the-Fly Update Test - Single Cell

Update the ‘restart_interval’ and ‘test_cell_update’ sections with the following values in the testMac config file ($cuBB_SDK/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml).

**testMAC configs:** Add cell_id 0 to the “test_cells” list to enable the test. Notice ‘vlan’ and ‘pcp’ is the same, but ‘dst_mac’ is different here. Change ‘test_sequence’ if required to test more cases.

```yaml
restart_interval: 3
# For cell net parameters update test
# Configs of slot_point=0 only runs at init, other configs will run repeatably.
test_cell_update:
test_cells: [0]
```

(continues on next page)
test_sequence:

- slot_point: 20000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}

- slot_point: 40000
  configs:
  - {cell_id: 0, dst_mac: 26:04:9D:9E:29:B3, vlan: 2, pcp: 7}

testMAC automatically calls the following script to change the net parameters during the testMAC initialization and before cell restarting (Note: m-plane cell_id = testMAC cell_id + 1).

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

In the ru-emulator config file ($cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml), change "Cell2" parameters to be the same as "Cell1", except for "eth" (the only difference is the eth MAC address).

```
- name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7

- name: "Cell2"
  eth: "26:04:9D:9E:29:B3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
```

Run normal F08 Pattern 0 1C E2E test commands, except change the ru-emulator parameter “1C” to “1C_X2”. The following only shows the test case parameters; refer to the F08 cases for full instructions:

```
sudo ./ru_emulator F08 1C_59_X2
sudo -E ./cuphycontroller_scf F08
sudo ./test_mac F08 1C 59
```

Test result:

- ru-emulator throughput first starts on cell 1.
- ru-emulator throughput switches between cell 0 to cell 1, and repeats.
- The switching time points are decided by the above “slot_point” in testMAC configurations. Currently 20000 slots = 10 seconds.
1.2.4.14.3.2 VLAN ID OAM On-the-Fly Update Test - Single Cell

Update 'restart_interval' and 'test_cell_update' section with the following in the testMac config file $cuBB_SDK/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.

**testMAC configs:** Add cell_id 0 to “test_cells” list to enable the test. Notice ‘dst_mac’ and ‘pcp’ are same, ‘vlan’ is different here. Change test_sequence if required to test more cases.

```yaml
# For cell net parameters update test
test_cell_update:
test_sequence:
  - slot_point: 20000
    configs:
      - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 3, pcp: 7}
  - slot_point: 40000
    configs:
      - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}
```

testMAC automatically calls the following script to change the net parameters, and stop then restart the cell. (Note: m-plane cell_id = testMAC cell_id + 1)

```bash
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

Update the ‘cell_configs’ section with the following for “Cell1” and “Cell2” in the ru-emulator config file $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml. Note: The only difference is the vlan id.

```yaml
- name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8, 0, 1, 2]
  eAxC_DL: [8, 0, 1, 2]
  eAxC_prach_list: [15, 7, 0, 1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
- name: "Cell2"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8, 0, 1, 2]
  eAxC_DL: [8, 0, 1, 2]
  eAxC_prach_list: [15, 7, 0, 1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 3
  pcp: 7
```

Run normal F08 Pattern 0 1C E2E test commands except change ru-emulator parameter “1C” to “1C_X2”. This example only shows the test case parameters, refer to F08 cases for full instructions:
Expected test result: ru-emulator to have throughput changed between cell 0 to cell 1, and repeat. The change time points are decided by the above “slot_point” in testMAC configurations. Currently 20000 slots = 10 seconds.

1.2.4.14.3.3 VLAN PCP OAM On-the-Fly Update Test - Single Cell

Update ‘restart_interval’ and ‘test_cell_update’ sections with the following in the testMac config file $cuBB_SDK/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.

**testMAC configs:** add cell_id 0 to “test_cells” list to enable the test. Notice ‘dst_mac’ and ‘vlan’ are same, ‘pcp’ is different here. Change test_sequence if requires to test more cases.

```
restart_interval: 3
# For cell net parameters update test
test_cell_update:
test_cells: [0]
test_sequence:
  - slot_point: 20000
    configs:
      - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 4}
  - slot_point: 40000
    configs:
      - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}
```

testMAC automatically calls the following script to change the net parameters, and stop then restart the cell. (Note: m-plane cell_id = testMAC cell_id + 1)

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

Update the ‘cell_configs’ section with the following for “Cell1” and “Cell2” in the ru-emulator config file $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml. Note: The only difference is the PCP value.

```
- name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
- name: "Cell2"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
```

(continues on next page)
Run normal F08 Pattern 0 1C E2E test commands except change ru-emulator parameter “1C” to “1C_X2” The following example only shows the test case parameters, refer to F08 cases for full instructions:

```
sudo ./ru_emulator F08 1C_59_X2
sudo -E ./cuphycontroller_scf F08
sudo ./test_mac F08 1C 59
```

Expected test result: ru-emulator has throughput changed between cell 0 to cell 1, and repeat. The change time points are decided by above “slot_point” in testMAC configurations. Currently 20000 slots = 10 seconds.

1.2.4.14.3.4 DST MAC OAM On-the-Fly Update Test (with OAM Cell Ctrl Command) - Multi-Cells

The following sequence diagram shows the capability of updating Dst_MAC/VLAN/PCP on the fly with multi-cell running.
# Save original configuration before the test

```bash
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml $cuBB_SDK/cuPHY-CP/testMAC/...testMAC/test_mac_config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml.orig
```

# Update config

```bash
sed -i "s∕oam_cell_ctrl_cmd:.*∕oam_cell_ctrl_cmd: 1∕" $cuBB_SDK/cuPHY-CP/testMAC/
...testMAC/test_mac_config.yaml

sed -i "s∕eAxC_UL:.*∕eAxC_UL: \[0,1,2,3\]∕" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
...config.yaml

sed -i "s∕eAxC_DL:.*∕eAxC_DL: \[0,1,2,3\]∕" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
...config.yaml

sed -i "s∕eAxC_prach_list:.*∕eAxC_prach_list: \[5,6,7,10\]∕" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
...config.yaml

sed -i "s∕cell_group_num:.*∕cell_group_num: 4∕" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
...config/cuphycontroller_F08_R750.yaml

sed -i "s∕\[.*∕∕g

${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml

sed -i "s∕eAxC_id_.*∕eAxC_id_prach: \[5, 6, 7, 10\]∕" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
...cuphycontroller_F08_R750.yaml
```

# Restore the configuration after the test

```bash
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig $cuBB_SDK/cuPHY-CP/
...testMAC/test_mac_config.yaml

cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml

cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml.orig $cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

RU-Emulator will use launch pattern file "xC_59_X2" for test:

- launch_pattern_F08_4C_59_X2.yaml
- launch_pattern_F08_8C_59_X2.yaml

**Note:** There is a known issue with running launch_pattern_F08_8C_59_X2.yaml.

Run normal F08 4C 59 E2E test commands except change ru-emulator parameter “4C” to “4C_59_X2”. The following example only shows the test case parameters, refer to F08 cases for full instructions:

```bash
sudo ./ru_emulator F08 4C_59_X2
sudo -E ./cuphycontroller_scf F08
sudo ./test_mac F08 4C 59
```

Init CONF.req is sent to all cells (executed on DU server):

```bash
cd ${cuBB_SDK}/build/cuPHY-CP/cuphyoam/
for i in \{0..3\}; do python3 ${cuBB_SDK}/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
...py --server_ip localhost --cell_id $i --cmd 3 && sleep 1; done;
```
START.req sent to all cells (executed on DU server):

```bash
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in \{0..3\}; do python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --cmd 1 && sleep 1; done;
```

At this point, validate that the RU emulator sees cell 0~3 have tput:

STOP.req sent to all cells (executed on DU server):

```bash
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in \{0..3\}; do python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --cmd 0 && sleep 1; done;
```

OAM update Cell i destination MAC updated to target cell i+4 on RU-Emulator side. That is: 0→4, 1→5, 2→6, 3→7) (executed on DU server):

```bash
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 1 20:04:00:9B:9E:27:05 E002 && sleep 1
python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 2 20:04:00:9B:9E:27:06 E002 && sleep 1
python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 3 20:04:00:9B:9E:27:07 E002 && sleep 1
python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 4 20:04:00:9B:9E:27:08 E002 && sleep 1
```

START.req sent to all cells (executed on DU server):

```bash
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in \{0..3\}; do python3 $cuBB_SDK/build/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --cmd 1 && sleep 1; done;
```

At this point, validate that the RU-Emulator sees cell 4~7 have tput:
1.2.4.15 Dynamic PRACH Configuration and Init Sequence Test

This sequence shows the changing of the PCI and 4 PRACH parameters after the initial config of a cell. There is also a possibility of changing the RU's VLAN ID and MAC address connected to the cell.
L1 Init and Adding RU Dynamically

1) Initial Cell and M-plane Setup
   - M-plane (RU: dummy dest MAC addr., )
     - Cell-1 Config request
     - M-plane (RU: dummy dest MAC addr., )
     - Cell-2 Config request
     - M-plane (RU: dummy dest MAC addr., )
     - Cell-3 Config request
     - M-plane (RU: dummy dest MAC addr., )
     - Cell-4 Config request

2) Dynamically Add RU-1
   - DHCP
     - M-plane (RU1: dest MAC addr., VLAN ID, PCP)
     - Cell-1 Config request with new PRACH params
     - Cell-1 Start request
     - cell-1 traffic

3) Dynamically Add RU-2
   - DHCP
     - M-plane (RU2: dest MAC addr., VLAN ID, PCP)
     - Cell-2 Config request with new PRACH params
     - Cell-2 Start request
     - cell-2 traffic

4) Dynamically Add RU-3
   - DHCP
     - M-plane (RU3: dest MAC addr., VLAN ID, PCP)
     - Cell-3 Config request with new PRACH params
     - Cell-3 Start request
     - cell-3 traffic

5) Dynamically Add RU-4
   - DHCP
     - M-plane (RU4: dest MAC addr., VLAN ID, PCP)
     - Cell-4 Config request with new PRACH params
     - Cell-4 Start request
     - cell-4 traffic

6) Replace RU for Cell-1
   - Cell-1 Stop request
   - DHCP
     - M-plane (RU: dest MAC addr., VLAN ID, PCP)
     - Cell-1 Start request
     - cell-1 traffic

RU power on

1.2. cuBB Quickstart Guide
To support the sequence above, testMac has been enhanced to send CONFIG.req and START.req using OAM commands. Aerial has been enhanced to support dynamic PRACH parameter configuration and change of PCI in release 22-2.3. Changing the VLAN-id and DST MAC address was supported in previous releases and is used to support the Init sequence as shown above. The six PRACH parameters that can be changed are as follows:

- `prachRootSequenceIndex`
- `restrictedSetConfig`
- `prachConfigIndex`
- `prachZeroCorrConf`
- `numPrachFdOccasions`
- `K1`
- `prachConfigIndex`
- `restrictedSetConfig`

To test this feature, testMac and ru-emulator are started with a higher number of cells from the cuphyController, and then OAM commands are used to change the configuration of a given cell.

Enable testMac to take OAM commands for CONFIG and START of a cell - change the `test_mac_config.yaml` file as follows:

```yaml
# Send cell config/start/stop request via OAM command
oam_cell_ctrl_cmd: 1
```

To test the sequence with n cells, change `cell_group_num` to n in `cuphycontroller_F08_*.yaml` and other corresponding files.

```yaml
cell_group: 1
cell_group_num: n
fix_beta_dl: 0
```

For example, for 8C -

```yaml
cell_group: 1
cell_group_num: 8
fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```bash
sed -i "s/eAxC_UL::*/eAxC_UL: \[[0,1,2,3]\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/eAxC_DL::*/eAxC_DL: \[[0,1,2,3]\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/eAxC_prach_list::*/eAxC_prach_list: \[[5,6,7,10]\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/\[[.]*\]/g" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
sed -i "s/eAxC_id_.*/&\[0, 1, 2, 3\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
sed -i "s/eAxC_id_prach_.*/eAxC_id_prach: \[[5, 6, 7, 10]\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

Run cuphycontroller, testMac for > nC and ru-emulator for > nC. For example, for 8C:
After testMac has created the gRPC Server and after you see the following logs on the testMac console, you can issue the OAM commands from the OAM window.

```plaintext
gRPC Server listening on 0.0.0.0:50052
20:33:56.124414 C [NVIPC:SHM] shm_ipc_open: forward_enable=0 fw_max_msg_buf_count=0
20:33:56.124434 C [MAC.PROC] set_launch_pattern_and_configs: fapi_type=1 tb_loc=1
20:33:56.124439 C [MAC.PROC] test_mac: create SCF FAPI interface
```

Execute the OAM commands for testMac from an OAM window:

- CONFIG.req command for all n cells. cmd=3 is for CONFIG.req
- Start cell-0 (cmd=1)

For example, for 8C:

```plaintext
export cuBB_SDK=$(pwd)
cd build/cuPHY-CP/cuphyoam/
for i in {0..7}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --cmd 3 &amp;&amp; sleep 1; done; //Send CONFIG.req
```

At this time you can see traffic running only for cell-0 on testMac, cuphycontroller and ru-emulator console:
Now give OAM commands to switch the change PCI and PRACH parameters for cell-1 to cell ‘n+1’.

For example, the following command triggers testMac to send another CONFIG.req for cell-1 with parameters for cell-9. The DST MAC address in the parameters for aerial_cell_param_new_update.py script must be the DST MAC address of n+1 cell in the cuphycontroller YAML file. For example, for 8C testcase, the DST MAC address for cell-9 in the cuphycontroller YAML file is:
Now testMAC and cuphycontroller see traffic for cell-0 and cell-1 while RU-Emulator sees traffic for cell-0 and cell-8.

```
# testMac console
20:35:00.125020 C [MAC.SCF] cell_start: cell_id=1 fapi_type=SCF global_tick=61130
20:35:00.560041 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
 | Prmb 100 | HARQ 1200 | SR 0 | CSI1 1200 | CSI2 2400 | ERR 0 | INV 0
20:35:00.560053 Cell 1 | DL 752.17 Mbps 695 Slots | UL 6.63 Mbps 174 Slots |
 | Prmb 43 | HARQ 5220 | SR 0 | CSI1 1044 | CSI2 1044 | ERR 0 | INV 174
20:35:00.560058 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:00.560063 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:01.560039 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:01.560050 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:01.560066 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:02.560041 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:02.560053 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:02.560066 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560040 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560051 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560065 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560066 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560043 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560054 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560059 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560064 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
 | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
```

# cuPhyController console

(continues on next page)
# ru-emulator console

12:15: 46.760099 Cell 8 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
   → 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_Cundance
   → 100.00% DL_U_ON 100.00% UL_C_ON 100.00% | Seconds 513

12:15: 46.760025 Cell 0 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
   → 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_Cundance
   → 100.00% DL_U_ON 100.00% UL_C_ON 100.00% | Seconds 514

12:15: 46.760041 Cell 1 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760049 Cell 2 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760054 Cell 3 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760068 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760073 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760078 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

12:15: 46.760083 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
   → 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_Cundance
   → 0.00% DL_U_ON 0.00% UL_C_ON 0.00% | Seconds 514

(continues on next page)
1.2.4.16 Duplicate Configuration and Init Sequence Test

Duplicate Cell Config.request is a feature that enables dynamically configuring and starts a cell on an individual basis. The Config.request for all the cells need not be sent before a Start.Req is issued. To enable this feature, the following configuration in L2Adapter and testMac must be provisioned.

```bash
sed -i "s/duplicate_config_all_cells.*/duplicate_config_all_cells: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08_R750.yaml

sed -i "s/duplicate_config_all_cells.*/duplicate_config_all_cells: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08_R750.yaml

sed -i "s/cell_config_wait.*/cell_config_wait: 1000/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/testMac_module.yaml

sed -i "s/oam_cell_ctrl_cmd.*/oam_cell_ctrl_cmd: 1/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/testMac_module.yaml
```

To test the sequence with n cells, change cell_group_num to n in cuphycontroller_F08_*.yaml and other corresponding files.
cell_group: 1
cell_group_num: n
fix_beta_dl: 0

For example, for 8C:

```bash
cell_group: 1
cell_group_num: 8
fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```bash
sed -i "s∕eAxC_UL::.*∕eAxC_UL: [0,1,2,3]∕" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
--config.yaml
sed -i "s∕eAxC_DL::.*∕eAxC_DL: [0,1,2,3]∕" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
--config.yaml
sed -i "s∕eAxC_prach_list::.*∕eAxC_prach_list: [5,6,7,10]∕" ${cuBB_SDK}/cuPHY-CP/ru-
--emulator/config/config.yaml
sed -i "s∕\[.*∕g" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_-
--R750.yaml
sed -i "s∕eAxC_id::.*∕eAxC_id: [5,6,7,10]∕" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
--cuphycontroller_F08_R750.yaml
```

Run cuphycontroller, testMac for > nC and ru-emulator for > nC. For example, for 8C:

```bash
sudo .∕cuPHY-CP∕ru-emulator∕ru_emulator∕ru_emulator F08 8C
sudo -E .∕cuPHY-CP∕cuphycontroller∕examples∕cuphycontroller_scf F08
sudo .∕cuPHY-CP∕testMAC∕testMAC∕test_mac F08 8C
```

After testMac has created the gRPC Server and after you see the following logs on the testMac con-
sole, you can issue the OAM commands from the OAM window:

```plaintext
gRPC Server listening on 0.0.0.0:50052
20:33:56.124414 C [NVIPC:SHM] shm_ipc_open: 0 fw_max_msg_buf_count=0
--fw_max_data_buf_count=0
20:33:56.124434 C [MAC.PROC] set_launch_pattern_and_configs: fapi_type=1 tb_loc=1
20:33:56.124439 C [MAC.PROC] test_mac: create SCF FAPI interface
```

Execute the OAM commands for testMac from a OAM window:

1. CONFIG.req command for 1 cell at a time. cmd=3 is for CONFIG.req
2. Start cell-0 (cmd=1)
3. Repeat the above sequence for all cells

```bash
export cuBB_SDK=$(pwd)
cd build/cuPHY-CP/cuphyoam/
#Note that the config&start of cells can be in any order
for i in [0..7]; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
--py --server_ip localhost --cell_id $i --cmd 3 && sleep 3 && python3 $cuBB_SDK/cuPHY-
--CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --
--cmd 1; done; //Send CONFIG.req and Start.req for cell 0~7
```
1.2.4.17 How to Get Aerial Metrics

Run the following on gNB Server#1. Make sure `-DAERIAL_METRICS=1` added in cmake config:

curl localhost:8081/metrics

Set the Prometheus thread to a proper CPU core number. For testing on R750 with non-HT setup, change the F08_R750 config file so that DPDK-related metrics are updated, then launch cuphycontroller:

```bash
sed -i "s/prometheus_thread.*/prometheus_thread: 23/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

```bash
sudo -E numactl -N 1 -m 1 -- ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08_R750
```

(continues on next page)
Do NOT start test_mac yet. Query the metrics. All metrics should be 0 except for:

- `aerial_cuphycp_net_tx_accu_sched_clock_queue_jitter_ns`
- `aerial_cuphycp_net_tx_accu_sched_clock_queue_wander_ns`

Launch RU emulator:
```
sudo ${cuBB_SDK}/build/cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 8C_59
```

Run testMAC with 20000 slots:
```
sed -i "s/test_slots.*/test_slots: 20000/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
sudo numactl -N 1 -m 1 -- ${cuBB_SDK}/build/cuPHY-CP/testMAC/testMAC/test_mac F08 8C_59
```

Let the test finish. Wait until you see that the test_mac output shows 20000 slots finished:
```
13:16:37.244835 C [MAC.FAPI] Finished running 20000 slots test
```

Don't kill the cuphycontroller yet. Query the metrics again, see the example log as follows:

```...
# HELP aerial_cuphycp_slots_total Aerial cuPHY-CP total number of processed Downlink slots
# TYPE aerial_cuphycp_slots_total counter
aerial_cuphycp_slots_total{cell="8",type="UL"} 4000
aerial_cuphycp_slots_total{cell="8",type="DL"} 16000
aerial_cuphycp_slots_total{cell="7",type="UL"} 4000
aerial_cuphycp_slots_total{cell="7",type="DL"} 16000
aerial_cuphycp_slots_total{cell="6",type="UL"} 4000
aerial_cuphycp_slots_total{cell="6",type="DL"} 16000
aerial_cuphycp_slots_total{cell="3",type="UL"} 4000
aerial_cuphycp_slots_total{cell="3",type="DL"} 16000
aerial_cuphycp_slots_total{cell="2",type="UL"} 4000
aerial_cuphycp_slots_total{cell="2",type="DL"} 16000
aerial_cuphycp_slots_total{cell="1",type="UL"} 4000
aerial_cuphycp_slots_total{cell="1",type="DL"} 16000
aerial_cuphycp_slots_total{cell="4",type="UL"} 4000
aerial_cuphycp_slots_total{cell="4",type="DL"} 16000
aerial_cuphycp_slots_total{cell="5",type="UL"} 4000
aerial_cuphycp_slots_total{cell="5",type="DL"} 16000
```

```...
# HELP aerial_cuphycp_on_time_uplane_rx_packets_total Aerial cuPHY-CP Uplink U-plane packets which arrived within their receive windows
# TYPE aerial_cuphycp_on_time_uplane_rx_packets_total counter
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="7"} 1680000
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="6"} 1680000
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="2"} 1680000
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="3"} 1680000
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="4"} 1680000
aerial_cuphycp_on_time_uplane_rx_packets_total{cell="5"} 1680000
```

```...
# HELP aerial_cuphycp_cplane_tx_bytes_total Aerial cuPHY-CP C-plane TX bytes
```
```(continues on next page)```
1.2.4.18 Run an Additional Logging Stream Container

Note: The nvlog_observer and nvlog_collect are deprecated in 23-1.

1) By default the logs are stored in '/tmp' location. You can set the environment variable AERIAL_LOG_PATH to define a customized logfile path.

2) The moment the log size crosses 20GB, a new file gets created. Like phy.log, phy.log.1, phy.log.2 … phy.log.7.

1.2.4.19 Run Multiple L2 Instances with Single L1 Instance

Rel-23-3 support static cell allocation for different L2 instances.

There’s a known limitation that all cells need to be configured (by FAPI CONFIG.req) before any cell starts scheduling. With the duplicate Cell Config.request feature introduced in 23-4, the dynamic L2 instances can be supported without the above limitation but the cell config on each L2 instance must be the same.

Example: Run two L2 instances with 4 cells for each and one L1 instance with 8 cells.

1) Assign a different “prefix” in nvipc config for each L2 instance. The “prefix” is a string whose length should be less than 32.

# nvipc config yaml for each L2 instance

# For L2 instance 0: test_mac_config.yaml
prefix: nvipc

# For L2 instance 1: test_mac_config_1.yaml
prefix: nvipc1

The first testMAC instance uses the default test_mac_config.yaml. After it is configured properly, make a copy of test_mac_config.yaml and configure it for the second testMAC instance. To run multiple testMAC instances on the same machine, CPU cores, logger name, and OAM server port must be changed. The following are the example commands to configure the 2nd testMAC instance:

```
cp '${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml' '${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml'
sed -i 's/prefix:.*/prefix: nvipc1/g' '${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml'
```
2) Switch nvipc config to nvipc_multi_instances.yaml in L1.

```sh
# l2_adapter_config_XXX.yaml
nvipc_config_file: nvipc_multi_instances.yaml
```

3) Config "prefix" in L1 and assign L1 cells for each L2 instance.

Assume 8 cells are configured in cuphycontroller_XXX.yaml, the indexes for them are 0 ~ 7.

L1 cells: 0 ~ 7
The 1st L2 instance cells: 0 ~ 3
The 2nd L2 instance cells: 4 ~ 7

Then configure:

```sh
# nvipc_multi_instances.yaml
transport:
- transport_id: 0
  phy_cells: [0, 1, 2, 3]
  type: shm
  shm_config:
    prefix: nvipc

- transport_id: 1
  phy_cells: [4, 5, 6, 7]
  type: shm
  shm_config:
    prefix: nvipc1
```

The cell_id map between L1 and L2 is maintained in cuphycontroller:
4) Run the test.

Use F08 8C_60 for example:

```bash
sudo ./ru_emulator F08 8C_60
sudo -E ./cuphycontroller_scf F08_R750
# Run the 1st test_mac instance with default config file: test_mac_config.yaml
sudo ./test_mac F08 8C_60 --cells 0x0F
# Run the 2nd test_mac instance with another config file: test_mac_config_1.yaml
sudo ./test_mac F08 8C_60 --cells 0xF0 --config test_mac_config_1.yaml
```

5) Review the 8 cells of throughput in the L1 and 4 cells of throughput in each L2 instance.
1.2.4.20 OAM Commands in Multiple L2 Instances

OAM commands have no change in multiple L2 instances case.

Notes:

1) The “schedule_total_time” tolerance in Multi-L2 cases is a bit lower than Single-L2 cases. Please set "schedule_total_time" to 0 ~ 450000 for Multi-L2 cases.

The minor difference in FAPI timing tolerance is expected because there are multiple NVIPC instances working in difference processes and additional SLOT.ind messages are added.

2) There are two types of cell IDs used in L1:
   - FAPI cell_id: cell instance index in each app. It starts from 0 and is unique in each L1/L2 app instance (but can be duplicated in different L2 app instances). It is also used as cell_id / handle_id in FAPI message.
   - mplane_id: an arbitrary integer value that is configurable in cuphycontroller_XXX.yaml and is used in cuPHYDriver.

The “cell_id” in OAM commands is the mplane_id not the FAPI cell_id.

```yaml
# cuphycontroller_XXX.yaml

cells:
  - name: O-RU 0  # FAPI cell_id is the cell instance index. For the first cell, FAPI cell_id = 0
    cell_id: 1  # Here "cell_id" is actually "mplane_id" in source code. Current default config is: mplane_id = FAPI cell_id + 1
```

In the following OAM command example, pass mplane_id = 1 to select the first cell.

```bash
# Usage: aerial_cell_param_net_update.py cell_id dst_mac_addr vlan_tci
```

1.2.5. cuBB on NVIDIA Cloud Native Stack

NVIDIA Cloud Native Stack (formerly known as Cloud Native Core) is a collection of software that runs cloud native workloads on NVIDIA GPUs. This section describes how to install and run the Aerial cuBB software examples on NVIDIA Cloud Native Stack and related components to run Aerial cuBB.

1.2.5.1 Installation of NVIDIA Cloud Native Stack

Prerequisite: The server must already have the OS, NVIDIA Driver, and other configuration as described in Installing Tools on Grace Hopper or Installing Tools on Dell R750.

The steps to install NVIDIA Cloud Native Stack follows the NVIDIA Cloud Native Stack v13.0 installation guide on GitHub, starting with section "Installing Container Runtime", with the following additional notes:

- Select containerd when given the choice between containerd or CRI-O in the install guide.
- For running an ru-emulator on a server without a GPU, it is necessary to remove/comment out the "BinaryName" field from /etc/containerd/config.toml on that server.
If this step is not done, an ru-emulator failed to start error message can occur

<table>
<thead>
<tr>
<th>State: Terminated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason: StartError</td>
</tr>
<tr>
<td>Exit Code: 128</td>
</tr>
<tr>
<td>Started: Thu, 01 Jan 1970 00:00:00 +0000</td>
</tr>
<tr>
<td>Finished: Wed, 17 Jan 2024 05:25:27 +0000</td>
</tr>
</tbody>
</table>

Enable k8s CPU Manager, Topology Manager, and Memory Manager.

1. Update each worker node's /var/lib/kubelet/config.yaml. The file to use depends on the server type.

   # For Aerial Devkit servers
   $ cat <<EOF | sudo tee -a /var/lib/kubelet/config.yaml
   # Additional Configuration
   
   # Feature Gates
   featureGates:
       MemoryManager: true
   
   # CPU Manager Configuration
   cpuManagerPolicy: "static"
   cpuManagerPolicyOptions:
       full-pcpus-only: "true"
   reservedSystemCPUs: 0-2,22-23
   
   # Topology Manager Configuration
   topologyManagerPolicy: "restricted"
   topologyManagerScope: "container"
   
   # Memory Manager Configuration
   memoryManagerPolicy: "Static"
   reservedMemory:
       - numaNode: 0
         limits:
           memory: 100Mi
   EOF

   # for Dell R750 servers
   $ cat <<EOF | sudo tee -a /var/lib/kubelet/config.yaml
   # Additional Configuration
   
   # Feature Gates
   featureGates:
       MemoryManager: true
   
   # CPU Manager Configuration
   cpuManagerPolicy: "static"
   cpuManagerPolicyOptions:
       full-pcpus-only: "true"
   reservedSystemCPUs: 0-3

   (continues on next page)
# Topology Manager Configuration

topologyManagerPolicy: "restricted"
topologyManagerScope: "pod"

# Memory Manager Configuration

memoryManagerPolicy: "Static"
reservedMemory:
- numaNode: 0
  limits:
    memory: 50Mi
- numaNode: 1
  limits:
    memory: 50Mi

EOF

2. Drain each worker node.

    # Run from k8s master or other server where you have the kube config
    kubectl drain $nodeName --force --ignore-daemonsets

3. Restart each worker node's kubelet.

    # Run on worker node
    sudo systemctl stop kubelet
    sudo rm -f /var/lib/kubelet/cpu_manager_state
    sudo rm -f /var/lib/kubelet/memory_manager_state
    sudo systemctl start kubelet
    sudo systemctl status kubelet

4. Confirm kubelet status, verify that it is healthy.

    $ systemctl status kubelet
    kubelet.service - kubelet: The Kubernetes Node Agent
    Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
    10-kubeadm.conf
    Active: active (running) since Thu 2023-10-12 19:36:05 UTC; 5s ago

5. Uncordon the node.

    # Run from k8s master or other server where you have the kube config
    kubectl uncordon $nodeName

6. Setup a registry secret called "regcred" to be able to pull from $YourPrivateRegistry container registry. Follow the procedure described in the Kubernetes documentation. If you are using $YourPrivateRegistry=nvcr.io, please remember to generate an API Key from the NGC API-Key Setup Portal if you don't already have one.
1.2.5.2 Building Aerial Binary Container

This section describes how to build an Aerial binary container for the cuphycontroller_scf, test_mac, and ru_emulator applications, along with some example scenario test vectors.

1. Extract the build script.

```bash
mkdir -p cuPHY-CP/
docker pull nvcr.io/ea_aerial_sdk/aerial:24-2-cubb
docker run --rm -u $(id -u) -v .:/staging nvcr.io/ea_aerial_sdk/aerial:24-2-cubb cp -a /opt/nvidia/cuBB/cuPHY-CP/container /staging/cuPHY-CP/
```

Install dependencies

```bash
sudo apt update
sudo apt install python3-pip -y
pip3 install hpccm
```

2. Build the binary container and push to your private container repository.

```
AERIAL_BUILD_IMAGE=nvcr.io/ea_aerial_sdk/aerial:24-2-cubb
AERIAL_RELEASE_REPO=→$YourPrivateRepo
AERIAL_RELEASE_VERSION_TAG=$YourTag
./cuPHY-CP/container/build_binary.sh
docker push $YourPrivateRepo/aerial_binary:$YourTag-amd64
```

1.2.5.3 Deploying Binary Container using Helm Chart

Configure the NGC cli tool - follow the steps in https://ngc.nvidia.com/setup/installers/cli

Login to NGC

```
$ ngc config set
```

2. Fetch the Helm Chart from NGC.

```
neg registry chart pull ea_aerial_sdk/aerial-l1
neg registry chart pull ea_aerial_sdk/aerial-ru-emulator
```

3. Create value overload files specific to your environment. You must change the following values:

- YourRUEmulatorNodeName
- YourPrivateRepo
- YourTag
- <MAC Address of DU's FH Port>
- YourDUNodeName

```
$ cat <<EOF | tee override-ru-emulator-binary.yaml
# Deployment customization
extraSpec:
  nodeName: "YourRUEmulatorNodeName"
image:
  repository: YourPrivateRepo/
```

(continues on next page)
pullPolicy: Always
# Overrides the image tag whose default is the chart appVersion.
tag: "YourTag"

peerethaddr: "<MAC Address of DU's FH Port>"

# Spacing is critical below
extraSetup: |
\`\`ru-emulator extra setup\`
    \
    sed -i "s∕enable_beam_forming:.*∕enable_beam_forming: 1∕" ..∕cuPHY-CP/ru-emulator/config/config_dyn.yaml

\`\`# Configure NIC PCIe Address\`
    sed -i "s∕nic_interface.*∕nic_interface: 0000:3b:00.0∕" ..∕cuPHY-CP/ru-emulator/config/config_dyn.yaml

EOF

$ cat <<EOF | tee override-l1-binary.yaml
# Deployment customization
extraSpec: 
    nodeName: "YourDUNodeName"

image: 
    repository: YourPrivateRepo/
    name: aerial_binary
pullPolicy: Always
# Overrides the image tag whose default is the chart appVersion.
tag: "YourTag"

enableTestMACContainer: 1

# Spacing is critical below
extraSetup: |
\`\`Aerial L1 extra setup\`
        \
        \`\`# Launch pattern related configuration\`
        sed -i "s∕cell_group_num: .*∕cell_group_num: 16∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;
        sed -i "s∕pusch_nMaxPrb: .*∕pusch_nMaxPrb: 136∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;

        \`\`# 3GPP conformance\`
        sed -i "s∕pusch_tdi:.*∕pusch_tdi: 1∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;
        sed -i "s∕pusch_cfo:.*∕pusch_cfo: 1∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;
        sed -i "s∕pusch_to:.*∕pusch_to: 1∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;

        \`\`# Configure NIC PCIe Address\`
        sed -i "s∕ nic:.*∕ nic: 0000:cc:00.1∕" ..∕cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;
(continues on next page)
# Spacing is critical below
extraTestMACSetup: |
  \# testMAC extra setup\`

  \#sed -i "s/test_slots: 0/test_slots: 100000/" ..\cuPHY-CP\testMAC\testMAC/\n  test_mac_config_dyncore.yaml;  
  sed -i "s/schedule_total_time: 0/schedule_total_time: 470000/" ..\cuPHY-CP\testMAC\testMAC/\n  test_mac_config_dyncore.yaml;  
  sed -i "s/fapi_delay_bit_mask: 0/fapi_delay_bit_mask: 0xF/" ..\cuPHY-CP\testMAC\testMAC/\n  test_mac_config_dyncore.yaml;  
  sed -i "s/builder_thread_enable: 0/builder_thread_enable: 1/" ..\cuPHY-CP\testMAC\testMAC/\n  test_mac_config_dyncore.yaml;  
EOF

4. Deploy the Helm Chart.

helm install aerial-ru-emulator-test aerial-ru-emulator-0.20234.0.tgz -f override-ru- 
  emulator-binary.yaml  
helm install aerial-l1-test aerial-l1-0.20234.0.tgz -f override-l1-binary.yaml

5. View the logs for each container.

# Run in separate windows
kubectl logs aerial-l1-test -f  
kubectl logs aerial-l1-test -c aerial-testmac-ctr -f  
kubectl logs aerial-ru-emulator-test -f

6. Remove the Helm Chart and destroy the pods when finished.

helm uninstall aerial-l1-test  
helm uninstall aerial-ru-emulator-test

1.2.5.4 Theory of Operation

At pod deployment time, k8s dynamically assigns dedicated CPU cores to the Aerial L1 cuphycon- 
troller_scf container and the testMAC container (if it is deployed). When the container starts up, 
the $cuBB_SDK/cuPHY-CP/cuphycontroller/config/autoconfig/auto_assign_cores.py script runs to map the k8s-assigned 
cores to the various Aerial functions. The following template configuration YAML files are used by the 
auto_assign_cores.py script:

▶ $cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_$configL1.yaml -> $cuBB_SDK/cuPHY- 
  CP/cuphycontroller/config/cuphycontroller_dyncore.yaml  
▶ $cuBB_SDK/cuPHY-CP/cuphycontroller/config/\$l2adapter_filename -> $cuBB_SDK/cuPHY- 
  CP/cuphycontroller/config/l2_adapter_dyncore.yaml  
▶ $cuBB_SDK/cuPHY-CP/testMAC/testMAC/$configMAC -> $cuBB_SDK/cuPHY- 
  CP/testMAC/testMAC/test_mac_config_dyncore.yaml

The variables used above come from:

▶ $cuBB_SDK: Environment variable defined in container  
▶ $configL1: Helm chart aerial-l1/values.yaml (or override-l1-binary.yaml if overridden) variable 
  ‘configL1’
<table>
<thead>
<tr>
<th>Core Number</th>
<th>Primary</th>
<th>Sibling</th>
<th>Uses</th>
<th>Core Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sibling Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low priority threads (inc. DPDK EAL)</td>
<td>53</td>
<td>H2D copy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UL Worker</td>
<td>55</td>
<td>[idle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UL Worker</td>
<td>57</td>
<td>[idle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL Worker</td>
<td>59</td>
<td>[idle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL Worker</td>
<td>61</td>
<td>[idle]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL Worker</td>
<td>63</td>
<td>[idle]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An example run of the auto_assign_cores.py script for the aerial-l1-ctr container is:

Detected HT Enabled
Detected Multiple NUMA Nodes: [0, 1]. Will use node 1 for scheduling.
OS core affinity: [19, 21, 23, 67, 69, 71]
OS core affinity for numa node 1: [19, 21, 23, 67, 69, 71]
OS isolated cores: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95]
Tentative primary cores: [19, 21, 23]
testMAC core assignment strategy:
* 1 low priority primary core, HT sibling idle
* 1 mac_recv thread primary core, HT sibling idle
* 1 builder thread primary core, HT sibling idle
Need 3 physical cores (plus 0 reserved), potential affinity for 3 isolated physical cores

Parsing testmac configuration template: /opt/nvidia/cuBB/cuPHY-CP/testMAC/testMAC/...test_mac_config.yaml
Writing testmac configuration: /opt/nvidia/cuBB/cuPHY-CP/testMAC/config/l2_adapter_config_F08_R750.yaml

<table>
<thead>
<tr>
<th>Core Number</th>
<th>Uses</th>
<th>Core Number</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td></td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>L2A timer</td>
<td></td>
<td>L2A msg</td>
<td></td>
</tr>
<tr>
<td>processing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parsing cuphycontroller configuration template: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config/cuphycontroller_dyncore.yaml
Writing cuphycontroller configuration: /opt/nvidia/cuBB/cuphycontroller/config/cuphycontroller_F08_R750.yaml
Parsing l2adapter configuration template: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08_R750.yaml
Writing l2adapter configuration: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config/l2_adapter_config_dyncore.yaml

(continued from previous page)
1.3. Aerial cuPHY

cuPHY is the 5G L1 library of the Aerial CUDA-Accelerated RAN. It is designed as an inline accelerator to run on NVIDIA GPUs and it does not require any additional hardware accelerator.

1.3.1. cuPHY Features Overview

This section provides an overview of supported features in cuPHY.

1.3.1.1 Supported Features

1.3.1.1.1 Aerial CUDA-Accelerated RAN Layer 1

Aerial CUDA-Accelerated RAN adheres to 3GPP Release 15 standard specifications to deliver the necessary Layer 1 capabilities for a gNB.

1.3.1.1.1.1 3GPP Release 15

Aerial cuPHY adheres to 3GPP Release 15 standard specifications to deliver the following capabilities for gNB Layer 1.

Overall PHY capabilities include:

- Error detection on the transport channel and indication to higher layers
- FEC encoding/decoding of the transport channel
- Hybrid ARQ soft-combining
- Rate matching of the coded transport channel to physical channels
- Mapping of the coded transport channel onto physical channels
- Power weighting of physical channels
- Modulation and demodulation of physical channels including:
  - Frequency and time synchronization
  - Radio characteristics measurements and indication to higher layers
  - Multiple Input Multiple Output (MIMO) antenna processing
  - Transmit Diversity (TX diversity)
  - Digital and Analog Beamforming
  - RF processing
### 1.3.1.1.2 PHY FH Interface

#### 1.3.1.1.2.1 Aerial CUDA-Accelerated RAN PHY Overall Capabilities

<table>
<thead>
<tr>
<th>Features</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard support</td>
<td>3GPP 5G NR Rel 15</td>
<td>P</td>
</tr>
<tr>
<td>Duplexing Mode</td>
<td>TDD</td>
<td>Y</td>
</tr>
<tr>
<td>Narrow Bandwidth (MHz)</td>
<td>30MHz, 40 MHz, 50MHz, 80MHz</td>
<td>P</td>
</tr>
<tr>
<td>Channel Bandwidth (MHz)</td>
<td>100 MHz</td>
<td>Y</td>
</tr>
<tr>
<td>Subcarrier Spacing (kHz)</td>
<td>30kHz</td>
<td>Y</td>
</tr>
<tr>
<td>Maximum Number of Subcarriers (Max number of RBs x Num of Subcarriers per RB) = 273 x 12</td>
<td>3276</td>
<td>Y</td>
</tr>
<tr>
<td>Downlink Waveform</td>
<td>CP-OFDM</td>
<td>Y</td>
</tr>
<tr>
<td>Uplink Waveform</td>
<td>CP-OFDM</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DFT-s-OFDM (for data and control)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Configurable to DFT-s-OFDM (for data &amp; Control)</td>
<td>Y</td>
</tr>
<tr>
<td>Number of Downlink SU-MIMO layers</td>
<td>Up to 4</td>
<td>Y</td>
</tr>
<tr>
<td>Number of Uplink SU-MIMO layers</td>
<td>1, 2</td>
<td>Y</td>
</tr>
<tr>
<td>Number of Tx physical antennas</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>Y</td>
</tr>
<tr>
<td>Number of Rx physical antennas</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>Y</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Features</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot format</td>
<td>DDDSUUDDDD S = 6:4:4 (DL: G: UL)</td>
<td>Y</td>
</tr>
<tr>
<td>Carrier Aggregation</td>
<td>Configurable component carriers</td>
<td>Y</td>
</tr>
<tr>
<td>Configurable BW Parts</td>
<td>Up to 4</td>
<td>Y</td>
</tr>
<tr>
<td>BBU-RRU split option</td>
<td>7.1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>N</td>
</tr>
<tr>
<td>Maximum Downlink throughput per user (Mbps) 4T4R configuration</td>
<td>1870</td>
<td>Y</td>
</tr>
<tr>
<td>Maximum Uplink throughput per user (Mbps) 4T4R configuration</td>
<td>467</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.1.1.3 TS 38.211 Numerologies, Physical Resources, Modulation, Sequence, Signal Generation

1.3.1.1.3.1 Aerial CUDA-Accelerated RAN PHY Numerologies

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerologies:Normal CP</td>
<td>μ=0 : SCS=15kHz, 14symbol/slot, 10slot/frame, 1slot/subframe, Normal CP</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>μ=1 : SCS=30kHz, 14symbol/slot, 20slot/frame, 2slot/subframe, Normal CP</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>μ=2 : SCS=60kHz, 14symbol/slot, 40slot/frame, 4slot/subframe, Normal CP</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>μ=3 : SCS=120kHz, 14symbol/slot, 80slot/frame, Normal CP</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>μ=4 : SCS=240kHz, 14symbol/slot, 160slot/frame, Normal CP</td>
<td>N</td>
</tr>
<tr>
<td>Numerologies:Extended CP</td>
<td>μ=2 : SCS=60kHz, 12symbol/slot, 40slot/frame, 4slot/subframe, Extended CP</td>
<td>N</td>
</tr>
</tbody>
</table>
### 1.3.1.1.3.2 Aerial CUDA-Accelerated RAN Overall PHY Physical Resources

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antenna Ports</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Grid</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Elements</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Block</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Block - Common Resource Block (CRB)</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Block - Physical Resource Block (PRB)</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Block - Virtual Resource Block (VRB)</td>
<td>Y</td>
</tr>
<tr>
<td>Bandwidth Part (BWP)</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Bandwidth Part (BWP)**

Dynamically adapt the carrier bandwidth and numerology in which a UE operates.

A bandwidth part is a subset of contiguous common resource blocks for a given numerology \( i \) in bandwidth part \( i \) on a given carrier.

A UE can be configured with up to four bandwidth parts in UL and DL.

---

### 1.3.1.1.3.3 Aerial CUDA-Accelerated RAN PHY Physical Resources – BWP

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth Part (BWP)</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Bandwidth Part (BWP)**

Dynamically adapt the carrier bandwidth and numerology in which a UE operates.

A bandwidth part is a subset of contiguous common resource blocks for a given numerology \( i \) in bandwidth part \( i \) on a given carrier.

A UE can be configured with up to **four** bandwidth parts in **both** UL and DL.

---

Default Aerial CUDA-Accelerated RAN startup configuration to not use BWP, can be enabled to support BWP on a per carrier basis (while cell OOS)

N

Default Aerial CUDA-Accelerated RAN startup configuration to not use BWP, can be enabled to support BWP on a per carrier basis at startup

N
1.3.1.1.3.4 Aerial CUDA-Accelerated RAN Overall Carrier Aggregation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Supported (emulated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Aggregation</td>
<td>Transmissions in multiple cells can be aggregated to support inter-band and intra-band configurations</td>
<td>Y</td>
</tr>
<tr>
<td>100MHz</td>
<td>Up to 2 cells aggregation(1CC,2CC)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Up to 4 cells aggregation(1CC,2CC,3CC, 4CC)</td>
<td>Y</td>
</tr>
<tr>
<td>Narrowband Carrier Aggregation</td>
<td>Configurable upto 4 component carriers</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.1.1.3.5 Aerial CUDA-Accelerated RAN PHY Modulation Mapper

<table>
<thead>
<tr>
<th>Modulation Scheme</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pi/2 BPSK</td>
<td>Y</td>
</tr>
<tr>
<td>BPSK</td>
<td>Y</td>
</tr>
<tr>
<td>QPSK</td>
<td>Y</td>
</tr>
<tr>
<td>16QAM</td>
<td>Y</td>
</tr>
<tr>
<td>64QAM</td>
<td>Y</td>
</tr>
<tr>
<td>256QAM</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.1.1.3.6 Aerial CUDA-Accelerated RAN PHY Sequence Generation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Supported</th>
</tr>
</thead>
</table>
| Sequence Generation| Pseudo-random sequence generation  
Generic pseudo-random sequences are defined by a length-31 Gold sequence | Y         |
|                    | Low-PAPR sequence generation type 1                                         | Y         |
|                    | Low-PAPR sequence generation type 2                                         | Y         |
### 1.3.1.1.3.7 OFDM Baseband Signal Generation (UL DFT-S-OFDM)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal generation for all channels except PRACH &amp; RIM-RS</td>
<td>RU support expected</td>
<td></td>
</tr>
<tr>
<td>PRACH</td>
<td>RU support expected</td>
<td></td>
</tr>
<tr>
<td>RIM-RS</td>
<td>RU support expected</td>
<td></td>
</tr>
<tr>
<td>Uplink waveform</td>
<td>DFT-S-OFDM for UL.</td>
<td>Y</td>
</tr>
<tr>
<td>Support concurrent UE configuration to use CP-OFDM or DFT-S-OFDM on same cell.</td>
<td>Some specific parameters:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Support for PUSCH and for PUCCH format 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Support 0.5 pi-BPSK for Modulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Support DMRS group hopping</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Support DMRS sequence hopping</td>
<td></td>
</tr>
</tbody>
</table>
### Aerial CUDA-Accelerated RAN, Release 24-2

1.3.1.1.4 TS 38.211 Channels

1.3.1.1.4.1 Aerial CUDA-Accelerated RAN Physical Overall Channels and Reference Signals

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downlink Channels (TX)</td>
<td>PDSCH processing</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PDCCH processing</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PBCH processing</td>
<td>Y</td>
</tr>
<tr>
<td>Downlink signals (TX)</td>
<td>DMRS for PDSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DMRS for PDCCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DMRS for PBCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PSS, SSS</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CSI-RS, TRS</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PT-RS</td>
<td>N</td>
</tr>
<tr>
<td>Downlink Physical Resources</td>
<td>Antenna ports starting with 1000 for PDSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna ports starting with 2000 for PDCCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna ports starting with 3000 for channel-state information reference signals</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna ports starting with 4000 for SS/PBCH block transmission</td>
<td>Y</td>
</tr>
<tr>
<td>Uplink Channels (RX)</td>
<td>PUSCH processing</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PUCCH processing</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PRACH processing</td>
<td>Y</td>
</tr>
<tr>
<td>Uplink signals (RX)</td>
<td>DMRS for PUSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DMRS for PUCCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>SRS</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PT-RS</td>
<td>N</td>
</tr>
<tr>
<td>Uplink physical Resources</td>
<td>Antenna ports starting with 0 for PUSCH and associated demodulation reference signals</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna ports starting with 1000 for SRS</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna ports starting with 2000 for PUCCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Antenna port 4000 for PRACH</td>
<td>Y</td>
</tr>
</tbody>
</table>
### 1.3.1.1.4.2 Aerial CUDA-Accelerated RAN Overall Channel - PUSCH (Physical Uplink Shared Channel)

<table>
<thead>
<tr>
<th>Features</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of codewords</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>Scrambling</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Modulation schemes</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>- Pi/2-BPSK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- QPSK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 16 QAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 64 QAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 256 QAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUSCH transform precoding mode</td>
<td>Disable</td>
<td>Y</td>
</tr>
<tr>
<td>Precoding</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>HARQ process</td>
<td>Number of HARQ process = 1</td>
<td>Y</td>
</tr>
<tr>
<td>Mapping to virtual resource blocks</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>VRB to PRB mapping Type</td>
<td>Non-interleaved</td>
<td>Y</td>
</tr>
<tr>
<td>Transmission Mode</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>- SU-MIMO up to 4 layers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- MU-MIMO 2 up to 4 layers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- MU-MIMO up to 8 layers</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>PUSCH DMRS CDM group without data</td>
<td>PUSCH DMRS CDM group without data 1</td>
<td>Y</td>
</tr>
<tr>
<td>PUSCH users per TTI</td>
<td>16</td>
<td>Y</td>
</tr>
<tr>
<td>Uplink algorithm</td>
<td>UL HARQ control</td>
<td>Y</td>
</tr>
<tr>
<td>Rate Matching</td>
<td>I_LBRM = 1 (Limited Buffer Rate Matching)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>I_LBRM = 0 (Limited Buffer Rate Matching)</td>
<td>Y</td>
</tr>
</tbody>
</table>

Aerial CUDA-Accelerated RAN Overall Channel - PUCCH (Physical Uplink Control Channel)
<table>
<thead>
<tr>
<th>Format</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>N</td>
</tr>
<tr>
<td>UCI sched coding, AFC, DFT (Format 1)</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Modulation schemes</td>
<td>Pi/2-BPSK, BPSK, QPSK</td>
<td>Y</td>
</tr>
<tr>
<td>Scheduling Request SR</td>
<td>Support needed</td>
<td>Y</td>
</tr>
<tr>
<td>Group hopping</td>
<td>neither</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>disable</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>enable</td>
<td>Y</td>
</tr>
<tr>
<td>Sequence cyclic shift</td>
<td>Zadoff-Chu sequence</td>
<td>Y</td>
</tr>
<tr>
<td>Intra-slot Frequency hopping/second hop PRB</td>
<td>Support</td>
<td>Y</td>
</tr>
<tr>
<td>Inter-slot Frequency hopping/second hop PRB</td>
<td>Support</td>
<td>Y</td>
</tr>
<tr>
<td>PUCCH over multiple slots</td>
<td>Number of slots - 2,4,8</td>
<td>N</td>
</tr>
<tr>
<td>Frequency Offset Correction</td>
<td>PUCCH format 1, 3</td>
<td>N</td>
</tr>
<tr>
<td>Multi-UE support</td>
<td>24 UEs / TTI</td>
<td>Y</td>
</tr>
<tr>
<td>PUCCH UCI HARQ-ACK Polar</td>
<td>codeblock CB size &lt; 359, lifsize = 8</td>
<td>Y</td>
</tr>
</tbody>
</table>
### PRACH (PHY Random Access Channel)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>A1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>A2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>A3</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>B1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>B2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>B3</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>B4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>C0</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>Subcarrier Spacing (kHz)</td>
<td>1.25</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>Y</td>
</tr>
<tr>
<td>Sequence cyclic shift</td>
<td>Zadoff-Chu sequence</td>
<td>Y</td>
</tr>
<tr>
<td>Preamble length</td>
<td>839</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>139</td>
<td>Y</td>
</tr>
<tr>
<td>Number of PRACH occasions per TTI</td>
<td>4 FDM</td>
<td>Y</td>
</tr>
<tr>
<td>Contention based Random Access</td>
<td>Configurable non-contention based Random Access</td>
<td>N</td>
</tr>
</tbody>
</table>
### 1.3.1.1.4.4 Aerial CUDA-Accelerated RAN Overall PHY - UL Reference Signals

<table>
<thead>
<tr>
<th>Configuration</th>
<th>PUSCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSCH DMRS sequence generation when transform precoding is disabled</td>
<td>Neither group, nor sequence hopping enabled</td>
</tr>
<tr>
<td>PUSCH DMRS sequence generation when transform precoding is enabled</td>
<td>Group hopping is enabled and sequence hopping is disabled</td>
</tr>
<tr>
<td>Demodulation reference signal for PUSCH Mapping to physical resources</td>
<td>DM-RS configuration type 1</td>
</tr>
<tr>
<td></td>
<td>DM-RS configuration type 2</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-max-len=1</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-max-len=2</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-add-pos=0</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-add-pos=1</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-add-pos=2</td>
</tr>
<tr>
<td></td>
<td>UL-DMRS-add-pos=3</td>
</tr>
<tr>
<td>Phase-tracking reference signals for PUSCH Sequence generation</td>
<td>transform precoding is not enabled</td>
</tr>
<tr>
<td></td>
<td>transform precoding is enabled</td>
</tr>
<tr>
<td>Phase-tracking reference signals for PUSCH Mapping to physical resources</td>
<td>transform precoding is disabled</td>
</tr>
<tr>
<td></td>
<td>transform precoding is enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuration</th>
<th>PUCCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demodulation reference signal for PUCCH format 1</td>
<td>no intra-slot frequency hopping</td>
</tr>
<tr>
<td></td>
<td>intra-slot frequency hopping enabled</td>
</tr>
<tr>
<td>Demodulation reference signal for PUCCH format 2</td>
<td></td>
</tr>
<tr>
<td>Demodulation reference signal for PUCCH format 3</td>
<td>No additional DM-RS, No hopping</td>
</tr>
<tr>
<td>Format 4 not supported</td>
<td>No Additional DM-RS, hopping</td>
</tr>
<tr>
<td></td>
<td>Additional DM-RS, No hopping</td>
</tr>
<tr>
<td></td>
<td>Additional DM-RS, hopping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuration</th>
<th>SRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sounding reference signal resource</td>
<td>Antenna ports=1, 1OFDM symbols</td>
</tr>
<tr>
<td></td>
<td>Antenna ports=1, 2OFDM symbols</td>
</tr>
<tr>
<td></td>
<td>Antenna ports=1, 4OFDM symbols</td>
</tr>
<tr>
<td></td>
<td>Antenna ports=2, 1OFDM symbols</td>
</tr>
<tr>
<td></td>
<td>Antenna ports=2, 2OFDM symbols</td>
</tr>
</tbody>
</table>
### Table 2 – continued from previous page

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antenna ports=2, 4OFDM symbols</td>
<td>Y</td>
</tr>
<tr>
<td>Antenna ports=4, 1OFDM symbols</td>
<td>Y</td>
</tr>
<tr>
<td>Antenna ports=4, 2OFDM symbols</td>
<td>Y</td>
</tr>
<tr>
<td>Antenna ports=4, 4OFDM symbols</td>
<td>Y</td>
</tr>
<tr>
<td>Sounding reference signal Sequence generation</td>
<td>KTC=2</td>
</tr>
<tr>
<td></td>
<td>KTC=4</td>
</tr>
<tr>
<td></td>
<td>KTC=8</td>
</tr>
<tr>
<td>Sounding reference signal Mapping to physical resources</td>
<td>CSRS=0~63</td>
</tr>
<tr>
<td>Sounding reference signal slot configuration</td>
<td>Indicated by higher layer parameter SRS-Config</td>
</tr>
</tbody>
</table>

**PTRS**

<table>
<thead>
<tr>
<th>PTRS Support</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td></td>
</tr>
</tbody>
</table>

### 1.3.1.1.4.5 Aerial CUDA-Accelerated RAN Overall Channel - PDSCH(PHY DL Shared Channel)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrambling</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Modulation schemes</td>
<td>QPSK</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>16 QAM</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>64 QAM</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>256 QAM</td>
<td>Y</td>
</tr>
<tr>
<td>Transmission Mode</td>
<td>4T4R SU-MIMO up to 4 layers</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>32T32R SU-MIMO up to 8 layers</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>64T64R SU-MIMO up to 16 layers</td>
<td>P</td>
</tr>
<tr>
<td>Number of codewords</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td>Number of antenna ports</td>
<td>1000 - 1011</td>
<td>Y</td>
</tr>
<tr>
<td>Number of physical antennas</td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>P</td>
</tr>
<tr>
<td>Beam Forming weights computation</td>
<td>BF m2</td>
<td>N</td>
</tr>
<tr>
<td>Precoding</td>
<td>non-codebook</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>pre-coding weight</td>
<td>N</td>
</tr>
</tbody>
</table>

continues on next page
Table 3 – continued from previous page

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I Single-Panel Codebook</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Type I Multi-Panel Codebook</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Type II Codebook</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Type II Port Selection Codebook</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>PDSCH mapping type</td>
<td>Type A</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Type B</td>
<td>Y</td>
</tr>
<tr>
<td>Resource allocation type</td>
<td>Type 0</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Type 1</td>
<td>Y</td>
</tr>
<tr>
<td>VRB to PRB mapping Type</td>
<td>Non-interleaved</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Interleaved</td>
<td>N</td>
</tr>
<tr>
<td>PDSCH DMRS CDM groups without data</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>Number PDSCH users per TTI</td>
<td>16</td>
<td>Y</td>
</tr>
<tr>
<td>Power Control</td>
<td>PDSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DMRS - PDSCH</td>
<td>Y</td>
</tr>
</tbody>
</table>
### 1.3.1.4.6 Aerial CUDA-Accelerated RAN Overall Channel - PDCCH (Physical DL Control Channel)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrambling</td>
<td>Up to 2 codewords</td>
<td>N</td>
</tr>
<tr>
<td>CORESET</td>
<td>Normal</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>RMSI CORESET</td>
<td>Y</td>
</tr>
<tr>
<td>SSB - RMSI CORESET multiplexing pattern</td>
<td>Pattern 1</td>
<td>Y</td>
</tr>
<tr>
<td>Aggregation Level</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Y</td>
</tr>
<tr>
<td>Modulation schemes</td>
<td>QPSK</td>
<td>Y</td>
</tr>
<tr>
<td>Layer mapping</td>
<td>Supported</td>
<td>Y</td>
</tr>
<tr>
<td>Antenna port mapping</td>
<td>Supported</td>
<td>Y</td>
</tr>
<tr>
<td>Mapping to virtual resource blocks</td>
<td>Supported</td>
<td>Y</td>
</tr>
<tr>
<td>Mapping from virtual to physical resource blocks</td>
<td>Non-interleaved VRB-to-PRB mapping</td>
<td>Y</td>
</tr>
<tr>
<td>Polar code</td>
<td>Block length up to 128 bits</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS (Demodulation Reference Signal)</td>
<td>m-sequence</td>
<td>Y</td>
</tr>
<tr>
<td>CCE To REG Mapping Type</td>
<td>Non-interleaved</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Interleaved</td>
<td>Y</td>
</tr>
<tr>
<td>Number OFDM symbol of CORESET</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Y</td>
</tr>
<tr>
<td>Power Control</td>
<td>PDCCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DMRS-PDCCH</td>
<td>Y</td>
</tr>
<tr>
<td>DCI format</td>
<td>0_0</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>0_1</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>1_0</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>1_1</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>2_x</td>
<td>NA</td>
</tr>
<tr>
<td>Precoding</td>
<td>Precoding Matrix Idx based precoding in</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>the DU</td>
<td></td>
</tr>
</tbody>
</table>
### 1.3.1.1.4.7 Aerial CUDA-Accelerated RAN Overall Channel - PBCH (Physical Broadcast Channel)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>cuBB Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precoding</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Scrambling</td>
<td>SS/PBCH block index Lmax=4</td>
<td>N</td>
</tr>
<tr>
<td>Scrambling</td>
<td>SS/PBCH block index Lmax=8</td>
<td>N</td>
</tr>
<tr>
<td>Scrambling</td>
<td>SS/PBCH block index Lmax=64</td>
<td>N</td>
</tr>
<tr>
<td>Modulation schemes</td>
<td>QPSK</td>
<td>Y</td>
</tr>
<tr>
<td>Mapping to Physical Resources</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DMRS Support</td>
<td>Support</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS config type</td>
<td>Type 1</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS config type</td>
<td>Type 2</td>
<td>N</td>
</tr>
<tr>
<td>DMRS type A Pos</td>
<td>Pos2</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS type A Pos</td>
<td>Pos3</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS max length</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS max length</td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS Additional Position</td>
<td>Pos0</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS Additional Position</td>
<td>Pos1</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS Additional Position</td>
<td>Pos2</td>
<td>Y</td>
</tr>
<tr>
<td>DMRS Additional Position</td>
<td>Pos3</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 1.3.1.1.4.8 Aerial CUDA-Accelerated RAN Overall - PHY DL Reference Signals

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demodulation reference signals for PDSCH</td>
<td>Sequence generation</td>
</tr>
<tr>
<td>Demodulation reference signals for PDSCH</td>
<td>Mapping to physical resources</td>
</tr>
<tr>
<td></td>
<td>DM-RS configuration type 1</td>
</tr>
<tr>
<td></td>
<td>DM-RS configuration type 2</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-max-len=1</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-max-len=2</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-add-pos=0</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-add-pos=1</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-add-pos=2</td>
</tr>
<tr>
<td></td>
<td>DL-DMRS-add-pos=3</td>
</tr>
</tbody>
</table>
### Table 4 – continued from previous page

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase-tracking reference signals (PTRS) for PDSCH Mapping to physical resources</td>
<td>LPT-RS=1</td>
</tr>
<tr>
<td></td>
<td>LPT-RS=2</td>
</tr>
<tr>
<td></td>
<td>LPT-RS=4</td>
</tr>
<tr>
<td><strong>PDCCH</strong></td>
<td></td>
</tr>
<tr>
<td>Demodulation reference signals for PDCCH Sequence generation</td>
<td></td>
</tr>
<tr>
<td>Demodulation reference signals for PDCCH Mapping to physical resources</td>
<td></td>
</tr>
<tr>
<td><strong>PBCH</strong></td>
<td></td>
</tr>
<tr>
<td>Demodulation reference signals for PBCH Sequence generation</td>
<td></td>
</tr>
<tr>
<td>Demodulation reference signals for PBCH Mapping to physical resources</td>
<td></td>
</tr>
<tr>
<td>CSI reference signals</td>
<td></td>
</tr>
<tr>
<td>CSI reference signals</td>
<td>Zero-power</td>
</tr>
<tr>
<td></td>
<td>non-zero-power</td>
</tr>
<tr>
<td>CSI reference signals Sequence generation</td>
<td>nID equals the higher-layer parameter</td>
</tr>
<tr>
<td>CSI reference signals Mapping to physical resources</td>
<td></td>
</tr>
<tr>
<td>Row 1: 1 port, Density = 3, CDMtype = No CDM</td>
<td></td>
</tr>
<tr>
<td>Row 2: 1 port, Density = 1, 0.5 CDMtype = No CDM</td>
<td></td>
</tr>
<tr>
<td>Row 3: 2 port, Density = 1, 0.5 CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 4: 4 port, Density = 1, CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 5: 4 port, Density = 1, CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 6: 8 port, Density = 1, CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 7: 8 port, Density = 1, CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 8: 8 port, Density = 1, CDMtype = CDM4 (FD2, TD2)</td>
<td></td>
</tr>
<tr>
<td>Row 9: 12 port, Density = 1, CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 10: 12 port, Density = 1, 0.5 CDMtype = CDM4 (FD2, TD2)</td>
<td></td>
</tr>
<tr>
<td>Row 11: 16 port, Density = 1, 0.5 CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 12: 16 port, Density = 1, 0.5 CDMtype = CDM4 (FD2, TD2)</td>
<td></td>
</tr>
<tr>
<td>Row 13: 24 port, Density = 1, 0.5 CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 14: 24 port, Density = 1, 0.5 CDMtype = CDM4 (FD2, TD2)</td>
<td></td>
</tr>
<tr>
<td>Row 15: 24 port, Density = 1, 0.5 CDMtype = CDM8 (FD2, TD4)</td>
<td></td>
</tr>
<tr>
<td>Row 16: 32 port, Density = 1, 0.5 CDMtype = FD-CDM2</td>
<td></td>
</tr>
<tr>
<td>Row 17: 32 port, Density = 1, 0.5 CDMtype = CDM4 (FD2, TD2)</td>
<td></td>
</tr>
<tr>
<td>Row 18: 32 port, Density = 1, 0.5 CDMtype = CDM8 (FD2, TD4)</td>
<td></td>
</tr>
<tr>
<td><strong>RIM</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

**1.3. Aerial cuPHY**
Table 4 – continued from previous page

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIM reference signal General</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The first RIM-RS type can be used</td>
</tr>
<tr>
<td></td>
<td>The second RIM-RS type depends on configuration only</td>
</tr>
<tr>
<td>RIM reference signal Sequence generation</td>
<td></td>
</tr>
<tr>
<td>RIM reference signal Mapping to physical resources</td>
<td></td>
</tr>
<tr>
<td>RIM reference signal RIM-RS configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enough Indication is disabled</td>
</tr>
<tr>
<td></td>
<td>Enough Indication is enabled</td>
</tr>
<tr>
<td>Positioning Reference</td>
<td></td>
</tr>
<tr>
<td>Positioning reference signal Sequence generation</td>
<td></td>
</tr>
<tr>
<td>Positioning reference signal Mapping to physical resources</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LPRS = 2, Kcomb = 2</td>
</tr>
<tr>
<td></td>
<td>LPRS = 4, Kcomb = 2</td>
</tr>
<tr>
<td></td>
<td>LPRS = 6, Kcomb = 2</td>
</tr>
<tr>
<td></td>
<td>LPRS = 12, Kcomb = 2</td>
</tr>
<tr>
<td></td>
<td>LPRS = 4, Kcomb = 4</td>
</tr>
<tr>
<td></td>
<td>LPRS = 12, Kcomb = 4</td>
</tr>
<tr>
<td></td>
<td>LPRS = 6, Kcomb = 6</td>
</tr>
<tr>
<td></td>
<td>LPRS = 12, Kcomb = 6</td>
</tr>
<tr>
<td></td>
<td>LPRS = 12, Kcomb = 12</td>
</tr>
<tr>
<td><strong>Synchronization signals</strong></td>
<td></td>
</tr>
<tr>
<td>SSB numerology</td>
<td>30 kHz</td>
</tr>
<tr>
<td>SSB precoding</td>
<td>supported</td>
</tr>
<tr>
<td>SSB burst set configuration</td>
<td>2 SS blocks w/ single SSB burst set</td>
</tr>
<tr>
<td>Synchronization signal generation</td>
<td>PSS generation and mapping to</td>
</tr>
<tr>
<td></td>
<td>SSS generation and mapping to</td>
</tr>
<tr>
<td>SS/PBCH block</td>
<td>Mapping of PSS within an SS/PBCH block</td>
</tr>
<tr>
<td></td>
<td>Mapping of SSS within an SS/PBCH block</td>
</tr>
<tr>
<td></td>
<td>Mapping of PBCH and DM-RS within an SS/PBCH block</td>
</tr>
</tbody>
</table>

**Chapter 1. Aerial modulation**
### 1.3.1.5 TS 38.212 Multiplexing and Channel Coding

#### 1.3.1.5.1 Aerial CUDA-Accelerated RAN Overall Multiplexing and Channel Coding

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Procedures</td>
<td>CRC calculation&lt;br&gt; All CRC len supported (6, 11, 16, 24)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Code block segmentation and code block CRC attachment&lt;br&gt; ▶ Polar coding&lt;br&gt; ▶ Low density parity check coding</td>
<td>Y</td>
</tr>
<tr>
<td>Transport to physical channel mapping - UL</td>
<td>UL-SCH -&gt; PUSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>RACH -&gt; PRACH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>UCI -&gt; PUCCH, PUSCH</td>
<td>Y</td>
</tr>
<tr>
<td>Transport to physical channel mapping - DL</td>
<td>DL-SCH -&gt; PDSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>BCH -&gt; PBCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PCH -&gt; PDSCH</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>DCI -&gt; PDCCH</td>
<td>Y</td>
</tr>
<tr>
<td>Channel coding schemes</td>
<td>Polar coding</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Low density parity check coding (LDPC)</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Channel coding of small block lengths</td>
<td>Y</td>
</tr>
<tr>
<td>Rate matching</td>
<td>Rate matching for Polar code</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Rate matching for LDPC code</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Rate matching for channel coding of small block lengths</td>
<td>Y</td>
</tr>
<tr>
<td>Code block concatenation</td>
<td>sequentially concatenating the rate matching outputs for the different code blocks&lt;br&gt; ▶ LDPC&lt;br&gt; ▶ Polar Coding</td>
<td>Y</td>
</tr>
<tr>
<td>uplink transport channels and control information</td>
<td>Random access channel</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Uplink shared channel&lt;br&gt; ▶ LDPC graph selection&lt;br&gt; ▶ Rate Matching&lt;br&gt; ▶ Code block concatenation</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Uplink control information&lt;br&gt; ▶ Uplink control information</td>
<td>Y</td>
</tr>
</tbody>
</table>
1.3.1.1.6 TS 38.213 Physical Layer Procedures for Control

1.3.1.1.6.1 Aerial CUDA-Accelerated RAN Overall - PHY Control Procedures

UE procedures (Not applicable to base station)

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronization procedures</td>
<td>Cell search</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Transmission timing adjustments</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Timing for secondary cell activation / deactivation</td>
<td>NA</td>
</tr>
<tr>
<td>Radio link monitoring</td>
<td>SSB based</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>CSI-RS based</td>
<td>NA</td>
</tr>
<tr>
<td>Link recovery procedures</td>
<td>radio link failure</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>beam failure recovery</td>
<td>NA</td>
</tr>
<tr>
<td>Uplink power control</td>
<td>Physical uplink shared channel</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Physical uplink control channel</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Sounding reference signal</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Physical random access channel</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Power ramping counter suspension</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Dual connectivity</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Power headroom report</td>
<td>NA</td>
</tr>
</tbody>
</table>

PHY RACH

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-1 random access procedure</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Type-2 random access procedure</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

UE procedures (Not applicable to base station)
### HARQ-ACK codebook determination

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBG-based HARQ-ACK codebook determination</td>
<td>CBG-based HARQ-ACK codebook determination</td>
<td>NA</td>
</tr>
<tr>
<td>Type-1 HARQ-ACK codebook determination in physical up-link control channel</td>
<td>Type-1 HARQ-ACK codebook determination in physical up-link control channel</td>
<td>NA</td>
</tr>
<tr>
<td>Type-1 HARQ-ACK codebook determination in physical up-link shared channel</td>
<td>Type-1 HARQ-ACK codebook determination in physical up-link shared channel</td>
<td>NA</td>
</tr>
<tr>
<td>Type-2 HARQ-ACK codebook determination in physical up-link control channel</td>
<td>Type-2 HARQ-ACK codebook determination in physical up-link control channel</td>
<td>NA</td>
</tr>
<tr>
<td>Type-2 HARQ-ACK codebook determination in physical up-link shared channel</td>
<td>Type-2 HARQ-ACK codebook determination in physical up-link shared channel</td>
<td>NA</td>
</tr>
<tr>
<td>Type-3 HARQ-ACK codebook determination</td>
<td>Type-3 HARQ-ACK codebook determination</td>
<td>NA</td>
</tr>
</tbody>
</table>

### UCI reporting on PUSCH

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short block codes for UCI</td>
<td>Input: 1 - 11 bits output 32 bits</td>
<td>Y</td>
</tr>
<tr>
<td>Multiplexing of coded UCI bits to PUSCH</td>
<td>CSI part 1, support maximum 48 bit</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CSI part 1 and CSI part 2, support maximum 48 bit</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Decoding UCI on PUSCH with PUSCH data (UCI-ON-PUSCH scaling) 0.5/0.65/0.8/1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Decoding UCI on PUSCH without PUSCH data (UCI-ON-PUSCH scaling) 0.5/0.65/0.8/1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>HARQ information length maximum 128</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Semi-static offset</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Dynamic offset</td>
<td>N</td>
</tr>
</tbody>
</table>

### UCI Reporting on PUCCH

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCI reporting on PUCCH</td>
<td>PUCCH Resource Sets before RRC connection establishment</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>PUCCH Resource Sets for RRC connected UE</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>UE procedure for reporting multiple UCI types</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>PUCCH repetition procedure</td>
<td>N</td>
</tr>
</tbody>
</table>

### UE Procedures (Not applicable to base station)
### UE procedure for determining physical downlink control channel assignment

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type0-PDCCH search space</td>
<td>common</td>
</tr>
<tr>
<td></td>
<td>Type0A-PDCCH search space</td>
<td>common</td>
</tr>
<tr>
<td></td>
<td>Type1-PDCCH search space</td>
<td>common</td>
</tr>
<tr>
<td></td>
<td>Type2-PDCCH search space</td>
<td>common</td>
</tr>
<tr>
<td></td>
<td>Type3-PDCCH search space</td>
<td>common</td>
</tr>
<tr>
<td></td>
<td>UE-specific search space</td>
<td>NA</td>
</tr>
</tbody>
</table>

### UE Procedure for Receiving Control Information

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDCCH validation for DL SPS and UL grant Type 2</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>PDCCH validation for DL SPS and UL grant Type 2</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>PDCCH monitoring indication and dormancy/non-dormancy behaviour for SCells</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>Search space set group switching</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>HARQ-ACK information for PUSCH transmissions</td>
<td></td>
<td>NA</td>
</tr>
</tbody>
</table>

### UE-Group Common Signaling

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE-group common signalling</td>
<td>Slot configuration</td>
<td>N</td>
</tr>
<tr>
<td>UE procedure for determining slot format</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Interrupted transmission indication</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Cancellation indication</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Group TPC commands for PUCCH/PUSCH</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>SRS switching</td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>

### Bandwidth Part Operation
<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWP</td>
<td>Configurable up to 4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({15, 15}) kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({15, 15}) kHz for frequency bands operated with shared spectrum channel access</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({15, 30}) kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({30, 15}) kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({30, 30}) kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({30, 15}) kHz for frequency bands operated with shared spectrum channel access</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({30, 30}) kHz for frequency bands operated with shared spectrum channel access</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({30, 30}) kHz for frequency bands operated with shared spectrum channel access</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({120, 60}) kHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({120, 120}) kHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({240, 60}) kHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Set of resource blocks and slot symbols of CORESET for Type0-PDCCH search space set when (\text{SS/PBCH block, PDCCH}) SCS is ({240, 120}) kHz</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Parameters for PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 1 and FR1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Parameters for PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 2 and (\text{SS/PBCH block, PDCCH}) SCS ({120, 120}) kHz</td>
<td>N</td>
</tr>
</tbody>
</table>
### 1.3.1.1.7 TS 38.214 Physical Layer Procedures for Data

#### 1.3.1.1.7.1 Aerial CUDA-Accelerated RAN Overall PHY Data Procedures

<table>
<thead>
<tr>
<th>Category</th>
<th>L1 Requirement</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>UL PUSCH Procedures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission Scheme</td>
<td>Codebook-based</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Non-codebook-based</td>
<td>Y</td>
</tr>
<tr>
<td>Resource allocation</td>
<td>Type 0</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Type 1</td>
<td>Y</td>
</tr>
<tr>
<td>Modulation order, redundancy version and transport block size determination</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Code block group based PUSCH transmission</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>MCS Table</td>
<td>Table64QAM Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Table256QAM Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Table64QAMLowSE</td>
<td>Y</td>
</tr>
<tr>
<td>PUSCH mapping type</td>
<td>Type A</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Type B</td>
<td>Y</td>
</tr>
<tr>
<td>CBG retransmission bitmap</td>
<td>Enable</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Disable</td>
<td>Y</td>
</tr>
</tbody>
</table>
### 1.3.1.1.8 FH Interfaces

#### 1.3.1.1.8.1 Aerial CUDA-Accelerated RAN Overall 4T4R L1 - L2 Layer Interface Based on SCF FAPI

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration (10.02)</th>
<th>Supported (Emulated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCF control interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Config.response</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Start.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Stop.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Stop.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Error.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Param.request (cap query)</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Param.response</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>SCF data interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL_TTI.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>UL_TTI.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>UL_DCI.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>SLOT errors</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>TX_Data.request</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>Rx_Data.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>CRC.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>UCI.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>SRS.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>RACH.indication</td>
<td>4T4R</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Aerial CUDA-Accelerated RAN Overall PHY FH Interface

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IOT Profiles</strong></td>
<td><strong>Simultaneous</strong> support of TDD profile(s) and TDD pattern on single GPU</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ NR TDD IOT Profile 1: NR-TDD-FR1-CAT-A-NoBF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ NR TDD IOT Profile 2: NR-TDD-FR1-CAT-A-DBF</td>
<td></td>
</tr>
<tr>
<td><strong>O-RAN CUS plane features with fronthaul 7.2-x split:</strong> [10][11]</td>
<td><strong>Simultaneous</strong> O-RU category support on same GPU/DU</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ CAT-A (precoding supported for PDCSH)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ CAT-B</td>
<td></td>
</tr>
<tr>
<td><strong>Beamforming</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Predefined beamID based beamforming</td>
<td></td>
</tr>
<tr>
<td><strong>IQ compression &amp; bit-width</strong></td>
<td><strong>Simultaneous support for</strong></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Static-bit-width Fixed point IQ (14 bit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ BFP IQ Compression (9 bit)</td>
<td></td>
</tr>
<tr>
<td><strong>O-DU timing</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Defined transport delay method</td>
<td></td>
</tr>
<tr>
<td><strong>Synchronization</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ G8275.1 (full timing support)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ LLS-C3 with PTP + SyncE</td>
<td></td>
</tr>
<tr>
<td><strong>Transport features</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ eCPRI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Application layer fragmentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ QoS over fronthaul</td>
<td></td>
</tr>
<tr>
<td><strong>Section types</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Section Type 1 (DL/UL channels)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Section Type 3 (PRACH)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Multiple sections within a single C-plane message</td>
<td></td>
</tr>
<tr>
<td><strong>Digital power scaling</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ UL gain correction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ DL reference level adjustment</td>
<td></td>
</tr>
<tr>
<td><strong>Rx window monitoring, Counters like</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Data received too early</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Data received too late</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Data received on-time</td>
<td></td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>▶ Support for upto 8 peak - 16 avg 100Mhz carriers</td>
<td></td>
</tr>
</tbody>
</table>
### 1.3.1.1.9 Measurements

#### 1.3.1.1.9.1 Aerial CUDA-Accelerated RAN Overall PHY Measurements 4T4R

<table>
<thead>
<tr>
<th>Measurements</th>
<th>Cfg Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PUSCH measurements</strong></td>
<td></td>
</tr>
<tr>
<td>RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>Pn+I pre-eq (Noise+Interference power)</td>
<td>4T4R</td>
</tr>
<tr>
<td>Pn+I post-eq (Noise+Interference power)</td>
<td>4T4R</td>
</tr>
<tr>
<td>SINR pre-eq</td>
<td>4T4R</td>
</tr>
<tr>
<td>SINR post-eq</td>
<td>4T4R</td>
</tr>
<tr>
<td>Timing Advance</td>
<td>4T4R</td>
</tr>
<tr>
<td><strong>PUCCH measurements</strong></td>
<td></td>
</tr>
<tr>
<td>PUCCH Format 0</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF0 RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF0 RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF0 Pn+i</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF0 timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td>PUCCH Format 1</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF1 RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF1 RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF1 Pn+i</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF1 timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td>PUCCH Format 2</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF2 RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF2 RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF2 Pn+i</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF2 timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td>PUCCH Format 3</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF3 RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF3 RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF3 Pn+i</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF3 timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td>PUCCH Format 4</td>
<td>4T4R</td>
</tr>
</tbody>
</table>
Table 5 – continued from previous page

<table>
<thead>
<tr>
<th>Measurements</th>
<th>Cfg Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF4 RSS</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF4 RSRP</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF4 Pn+i</td>
<td>4T4R</td>
</tr>
<tr>
<td>PF4 timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td><strong>PRACH measurements</strong></td>
<td></td>
</tr>
<tr>
<td>Pn+i (Noise+Interference power)</td>
<td>4T4R</td>
</tr>
<tr>
<td>Preamble signal strength</td>
<td>4T4R</td>
</tr>
<tr>
<td><strong>SRS measurements</strong></td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>4T4R</td>
</tr>
<tr>
<td>Received signal strength</td>
<td>4T4R</td>
</tr>
<tr>
<td>Timing advance</td>
<td>4T4R</td>
</tr>
<tr>
<td><strong>All channels measurements</strong></td>
<td></td>
</tr>
<tr>
<td>Both pre-equalization and post-equalization across all channels should be configurable and supported</td>
<td>4T4R</td>
</tr>
</tbody>
</table>
1.3.1.1.10  TS 38.104 (base station radio Tx and Rx) Base Station (BS) Radio Transmission and Reception

1.3.1.1.10.1  Aerial CUDA-Accelerated RAN Overall PHY Performance Conformance

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUSCH with transform precoding disabled</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>PUSCH with transform precoding enabled</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>UCI multiplexed on PUSCH</td>
<td>4T4R</td>
<td>Y</td>
</tr>
<tr>
<td>PUCCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTX to ACK probability</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for PUCCH format 0</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for PUCCH format 1</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for PUCCH format 2</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for PUCCH format 3</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for PUCCH format 4</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>Performance requirements for multi-slot PUCCH</td>
<td>4T4R</td>
<td>N</td>
</tr>
<tr>
<td>PRACH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance requirements for PRACH</td>
<td>PRACH False alarm probability</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>PRACH detection requirements</td>
<td>N</td>
</tr>
</tbody>
</table>

1.3.2.  Aerial CUDA-Accelerated RAN Features for 5G gNB

The 5G gNB capabilities, procedures, and interfaces have dependencies on Aerial CUDA-Accelerated RAN PHY Layer. The purpose of this section is to ensure that the Aerial CUDA-Accelerated RAN provides support for gNB capabilities, procedures, and interfaces.

1.3.2.1  Highlights

- PUCCH Format 1 I+N and SINR, DTX for UCI on PUSCH
- Predefined BeamId support
- Foxconn O-RU support
- Cell life cycle management
- 4T4R TDD 7 beam support
- 8-port CSI-RS
- Dynamic OAM supporting out-of-service updates:
1. Dest MAC and VLAN ID
2. exponent_dl
3. dl_iq_data_fmt
4. ul_iq_data_fmt
5. exponent_ul
6. max_amp_ul
7. section_3_time_offset
8. pusch_prb_stride
9. prach_prb_stride
10. fh_len_range
11. lower_guard_bw
12. gps_alpha (Shared across cells)
13. gps_beta (Shared across cells)
14. prachRootSequenceIndex
15. prachZeroCorrConf
16. numPrachFdOccasions
17. restrictedSetConfig
18. prachConfigIndex
19. K1

► Fronthaul Extension to 50km
► Simultaneous fronthaul ports for higher fronthaul bandwidth
► Multiple BandWidth Part (BWP) support
► 4T4R TDD bandwidth 10MHz, 30MHz, 40MHz, 50MHz and 80MHz
► Carrier aggregation:
  1. 100MHz + 80MHz
  2. 100MHz + 40MHz
  3. 80MHz + 40MHz
  4. 100MHz + 80MHz + 40MHz
► L1 startup time within 30sec
► Support for multiple L2 on a single converged card
► Cell-Id starts from 0 for all pods
1.3.2.1.1 Capabilities

1.3.2.1.1.1 Homogeneous Cell Lifecycle Mgmt - Cell State Mgmt (IS/OOS)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to support cell activation and de-activation. This is commonly referred to as taking a carrier OOS (Out of Service) and bringing it to IS (In Service) states</td>
<td>Y</td>
</tr>
</tbody>
</table>
## 1.3.2.1.2 Procedures

### 1.3.2.1.2.1 Aerial CUDA-Accelerated RAN Overall Beam and Carrier Mobility

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-gNB Handover</td>
<td>UE moves from 1 gNB to another gNB</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>► UL RRC transfer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► UE Context Modification Request/Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► UE Context Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► Serving and Target gNB cells can support different frequencies</td>
<td></td>
</tr>
<tr>
<td>Intra-DU Handover</td>
<td>Cell-level Mobility - UE establishes new connection to new carrier (inter-cell) supported by UE context modification procedure</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>► UE Context Modification Request/Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► UE Context Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► Serving and Target Cells can support different frequencies</td>
<td></td>
</tr>
<tr>
<td>Beam Mobility</td>
<td>UE establishes data path to new beam within carrier coverage (intra-cell)</td>
<td>N</td>
</tr>
<tr>
<td>Mobility at low speeds</td>
<td>Aerial CUDA-Accelerated RAN shall support pedestrian mobility by modeling the 3GPP channels and 38.104 requirements</td>
<td>N</td>
</tr>
<tr>
<td>Mobility at vehicular speeds</td>
<td>Aerial CUDA-Accelerated RAN shall support mobility at high vehicular speeds - upto 70mph (Doppler Shift = 400Hz)</td>
<td>N</td>
</tr>
</tbody>
</table>
1.3.2.1.2.2 UL Power Control

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single UE Power Control</td>
<td>BS initiated power control for single UEs</td>
<td>Y</td>
</tr>
<tr>
<td>UE Group Power Control</td>
<td>BS initiated power control for UE groups</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.2.1.2.3 Carrier Aggregation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Aggregation</td>
<td>Transmissions in multiple cells can be aggregated to support inter-band and intra-band configurations</td>
<td>Y</td>
</tr>
<tr>
<td>100MHz</td>
<td>Up to 2 cells aggregation(1CC,2CC) intra-band contiguous</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Up to 2 cells aggregation(1CC,2CC) intra-band non-contiguous</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Up to 4 cells aggregation(1CC,2CC,3CC,4CC) inter-band non contiguous</td>
<td>Y</td>
</tr>
<tr>
<td>Narrowband Carrier Aggregation (ZMHz)</td>
<td>Configurable upto 4 component carriers</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.2.1.3 Interfaces

1.3.2.1.3.1 gNB Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>NG Interface (TS 38.410)</td>
<td>Y</td>
</tr>
<tr>
<td>Xn interface (TS 38.420)</td>
<td>N</td>
</tr>
<tr>
<td>F1 interface (TS 38.470)</td>
<td>N</td>
</tr>
<tr>
<td>E1 interface (TS 38.460)</td>
<td>N</td>
</tr>
<tr>
<td>Front Haul interface - ORAN 7.2 Split (CUS version 3)</td>
<td>Y</td>
</tr>
<tr>
<td>E2 interface</td>
<td>N</td>
</tr>
<tr>
<td>O1 interface</td>
<td>N</td>
</tr>
</tbody>
</table>
1.3.2.2 Network, Services, and KPIs

This section includes E2E integration configuration and KPIs for appropriate NEs across 5G RAN, CN, and 5G infrastructure.

1.3.2.2.1 Highlights

- 3 Peak Cells validated in eCPRI setup. 8 Average cells (50% traffic) also validated in eCPRI setup
- 4 DL Layers and 2 UL Layers supported in 4T4R configuration
- 6 UE/TTI Supported
- Simultaneous Front Haul capability. Multi L2 also validated with each L2 supporting different cells.
- 1 Cell OTA verified

1.3.2.2.2 E2E Summary

- 3 Peak Cell in E2E configuration (CN + RAN + UE-EM) via eCPRI connection to test equipment (Achieving aggreagte DL throughput of 4.2Gbps)
- 1 Peak Cell in E2E configuration (CN + RAN + UE-EM) via RF cable connection to O-RU (Achieving DL throughput of 1.3Gbps and UL throughput of 100Mbps)
- Simultaneous Front Haul capability (8 peak cells) (4 Peak cells per Front Haul port)
- 1 Cell OTA in E2E configuration (CN + RAN + CUE) via OTA connection to UE device (Achieving DL throughput of 871Mbps and UL throughput of 99Mbps)
- 1 Cell OTA in E2E configuration (CN + RAN + CUE) via OTA connection to UE devices (Achieving 8 CUEs connected for greater than 8 hours)
- 1 Cell OTA testing in Bands n78 and CBRS
### 1.3.2.2.3 4T4R EA Overall Configuration and KPIs

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 15 SA</td>
<td>TDD 7.2 CatA</td>
<td>Y</td>
</tr>
<tr>
<td>Subcarrier spacing (SCS)</td>
<td>30kHz</td>
<td>Y</td>
</tr>
<tr>
<td>sub-6 frequency spectrum</td>
<td>n78 Germany (3700 - 3800 MHz)</td>
<td>Y</td>
</tr>
<tr>
<td>sub-6 frequency spectrum</td>
<td>n48 US CBRS (3550 - 3700 MHz)</td>
<td>Y</td>
</tr>
<tr>
<td>sub-6 frequency spectrum</td>
<td>n79</td>
<td>N</td>
</tr>
<tr>
<td>Channel bandwidth</td>
<td>100 Mhz</td>
<td>Y</td>
</tr>
<tr>
<td>MIMO Layers support</td>
<td>DL : 4 layer</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>UL : 2 layer</td>
<td></td>
</tr>
<tr>
<td>100MHz cells per GPU [GH200]</td>
<td>Up to 10 peak cells</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Up to 20 average cells (50%)</td>
<td></td>
</tr>
<tr>
<td>Peak throughput per cell</td>
<td>DL : 1.38 Gbps per cell</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>UL : 210 Mbps per cell</td>
<td></td>
</tr>
<tr>
<td>Number of RRC Connected UEs per cell</td>
<td>100</td>
<td>Y</td>
</tr>
<tr>
<td>Number of active data transmitting UEs per cell</td>
<td>256</td>
<td>N</td>
</tr>
<tr>
<td>Number of UEs/TTI</td>
<td>DL : 16 UE/TTI</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>UL : 16 UE/TTI</td>
<td></td>
</tr>
<tr>
<td>Frame structure and slot format</td>
<td>DDDSUUDDDD</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>S = 6:4:4 (DL: G: UL)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSUUUDSUUU</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>DDDDSU</td>
<td>Y</td>
</tr>
<tr>
<td>User plane latency (RRC connected mode)</td>
<td>10ms one way for DL and UL</td>
<td>Y</td>
</tr>
<tr>
<td>Synchronization and Timing support</td>
<td>IEEE 1588v2 PTP / SyncE</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>ORAN LLS-C3</td>
<td></td>
</tr>
<tr>
<td>MTU size</td>
<td>1500 bytes</td>
<td>Y</td>
</tr>
<tr>
<td>Modulation</td>
<td>256 QAM DL</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>256 QAM UL</td>
<td></td>
</tr>
<tr>
<td>Soak Testing</td>
<td>8 hours</td>
<td>Y</td>
</tr>
</tbody>
</table>
1.3.2.4 Aerial CUDA-Accelerated RAN Overall ORU Ecosystem

<table>
<thead>
<tr>
<th>ORU</th>
<th>Configuration</th>
<th>Freq Band</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foxconn RPQN-7801E</td>
<td>4T4R</td>
<td>3.7GHz - 3.8GHz (indoors)</td>
<td>Y</td>
</tr>
<tr>
<td>Fujitsu TA08029-B059</td>
<td>4T4R</td>
<td>3.6GHz - 3.7GHz</td>
<td>Y</td>
</tr>
<tr>
<td>Foxconn RP0N-7800</td>
<td>4T4R</td>
<td>3.7GHz - 3.8GHz (outdoors)</td>
<td>N</td>
</tr>
<tr>
<td>Fujitsu MU-MIMO</td>
<td>32T32R</td>
<td>3.7GHz - 3.8GHz</td>
<td>N</td>
</tr>
<tr>
<td>Foxconn RPQN-4800E</td>
<td>4T4R</td>
<td>CBRS 3.55GHz - 3.7GHz, indoor</td>
<td>N</td>
</tr>
</tbody>
</table>

1.3.2.5 Aerial CUDA-Accelerated RAN Overall UE Ecosystem

<table>
<thead>
<tr>
<th>UE</th>
<th>Configuration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>FourFaith Camera F-SC241-216-5G</td>
<td>4DL, 1UL</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>FourFaith Camera F-SC241-216-5G (EU)</td>
<td>4DL, 2UL</td>
<td></td>
</tr>
<tr>
<td>Handset</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>OnePlus Nord 5G AC2003 EU/UK Model</td>
<td>4DL, 1UL</td>
<td></td>
</tr>
<tr>
<td>Handset</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>Oppo Reno 5G</td>
<td>4DL, 1UL</td>
<td></td>
</tr>
<tr>
<td>Handset</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>Samsung S22</td>
<td>4DL, 1UL</td>
<td></td>
</tr>
<tr>
<td>Handset</td>
<td></td>
<td>SU-MIMO</td>
</tr>
<tr>
<td>Samsung S23</td>
<td>4DL, 1UL</td>
<td></td>
</tr>
</tbody>
</table>
## 1.3.2.2.6 5G Infrastructure Integration

### 1.3.2.2.6.1 5G RAN Integration

<table>
<thead>
<tr>
<th>Function</th>
<th>Features</th>
<th>Supported</th>
</tr>
</thead>
</table>
| gNB      | Baseband functions for  
► signal processing using multiple antennas  
► signal processing for detecting and correcting errors in the wireless transmission  
► signal processing to ensure that the wireless transmission is secure  
► managing the wireless resources efficiently between different devices in the network | Yes |
| O-RU     | Radio functions to convert digital information into signals that can be transmitted wirelessly, ensuring that the transmitted signals are in the right frequency bands and have the correct power levels. Includes antennas which radiate the electrical signals into radio waves | Yes |
| UE       | End user devices such as smartphones, routers, tablets, HMDs, CPEs | Yes |
### 1.3.2.2.6.2 5G Mobile Core (NGC) integration

<table>
<thead>
<tr>
<th>Function</th>
<th>Features</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMF Core Access and Mobility Management Function</td>
<td>Connection and reachability management, mobility management, access authentication and authorization, location services</td>
<td>Yes</td>
</tr>
<tr>
<td>SMF Session Management Function</td>
<td>UE session, including IP address allocation, selection of associated UP function, control aspects of QoS, and control aspects of UP routing.</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF Policy Control Function</td>
<td>Manage policy rules that other CP functions then enforce.</td>
<td>Yes</td>
</tr>
<tr>
<td>UDM Unified Data Management</td>
<td>Manage user identity, including generation of authentication credentials.</td>
<td>Yes</td>
</tr>
<tr>
<td>AUSF Authentication Server Function</td>
<td>Essentially an authentication server</td>
<td>Yes</td>
</tr>
<tr>
<td>UDR Unified Data Repository</td>
<td>Repository of subscriber information that can be used by other microservices. For example UDM</td>
<td>Yes</td>
</tr>
<tr>
<td>NCHF New Charging Function</td>
<td>Cover all the network's needs of charging and interaction with billing systems</td>
<td>Yes</td>
</tr>
<tr>
<td>CP - SDSF Structured Data Storage</td>
<td>“Helper” service used to store structured data.</td>
<td>Yes</td>
</tr>
<tr>
<td>CP - UDSF Unstructured Data Storage</td>
<td>“helper” service used to store unstructured data.</td>
<td>Yes</td>
</tr>
<tr>
<td>CP - NEF Network Exposure Function</td>
<td>Expose select capabilities to third-party services, including translation between internal and external representations for data. Could be implemented by an “API Server” in a microservices-based system.</td>
<td>N</td>
</tr>
<tr>
<td>CP - NRF NF Repository Function</td>
<td>A means to discover available services.</td>
<td>N</td>
</tr>
<tr>
<td>CP - NSSF Network Slicing Selector Function</td>
<td>A means to select a Network Slice to serve a given UE. Network slices are essentially a way to partition network resources in order to differentiate service given to different users.</td>
<td>N</td>
</tr>
<tr>
<td>UP - UPF User Plane Function</td>
<td>Forwards traffic between RAN and the Internet. In addition to packet forwarding, it is responsible for policy enforcement, lawful intercept, traffic usage reporting, and QoS policing</td>
<td>Y</td>
</tr>
</tbody>
</table>
## 1.3.2.2.7 5G NSE Overall Network Deployment Topologies

<table>
<thead>
<tr>
<th>Topology</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Prem Isolated Island</td>
<td>Co-located gNB + CN + MEC applications</td>
<td>Yes</td>
</tr>
<tr>
<td>Colocated 5G infra with low latency MEC applications + centralized 5GC</td>
<td>MEC applications + gNB + UPF with centralized 5G CN (CUPS support - with SBA and to minimize latency in user plane)</td>
<td>N</td>
</tr>
<tr>
<td>Campus Distributed MEC applications (latency tolerant)</td>
<td>Campus Distributed MEC applications + colocated (gNB + UPF + CN) - (Non latency sensitive applications can be distributed and leverage an existing enterprise network data stream)</td>
<td>N</td>
</tr>
<tr>
<td>CUPS Architecture Support</td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>
## 1.3.2.7.1 Aerial E2E Reference BOM and Component Manifest

<table>
<thead>
<tr>
<th>5G Infra Component</th>
<th>HW and SW Revision Manifest</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>gNB</td>
<td>SMC Grace Hopper MGX Serve with BF3 NIC</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Dell PowerEdge R750 Server with A100X</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Altran L2+</td>
<td>Y</td>
</tr>
<tr>
<td>CN</td>
<td>Dell PowerEdge R750 Server</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Altran CN</td>
<td>Y</td>
</tr>
<tr>
<td>FH Switch</td>
<td>Dell PowerSwitch S5248F-ON</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Adva switch FSP 150 XG400</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Spectrum switch SN3750X</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Ciena switch 5164</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Cisco switch N9K-C93180YC-FX3S</td>
<td>Y</td>
</tr>
<tr>
<td>GM</td>
<td>QULSAR Qg 2 Multi-Sync Gateway</td>
<td>Y</td>
</tr>
<tr>
<td>Cables</td>
<td>Dell C2G 1m LC-LC 50/125 Duplex Multimode OM4 Fiber Cable - Aqua - 3ft – Optical patch cable</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>NVIDIA MCP1600-C001E30N DAC Cable Ethernet 100GbE QSFP28 1m</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Beyondtech 5m (16ft) LC UPC to LC UPC Duplex OM3 Multimode PVC (OFNR) 2.0mm Fiber Optic Patch Cable</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CableCreation 3ft Cat5/Cat6 Ethernet Cables</td>
<td>Y</td>
</tr>
<tr>
<td>PDUs</td>
<td>Tripp Lite 1.4kW Single-Phase Monitored PDU with LX Platform Interface, 120V Outputs (8 5-15R), 5-15P, 12ft Cord, 1U Rack-Mount, TAA</td>
<td>Y</td>
</tr>
<tr>
<td>Transceivers</td>
<td>Finisar SFP-to-RJ45 Transceiver</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Intel Ethernet SFP+SR Optics</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Dell SFP28-25G-SR Transceiver</td>
<td>Y</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>Netgear ProSafe Plus JGS524E Rackmount</td>
<td>Y</td>
</tr>
</tbody>
</table>
1.3.2.2.7.2 Supported O-RUs

<table>
<thead>
<tr>
<th>ORU</th>
<th>Configuration</th>
<th>Freq Band</th>
<th>Supported</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foxconn RPQN-7801E</td>
<td>4T4R</td>
<td>3.7GHz - 3.8GHz (indoors)</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Fujitsu TA08029-B059</td>
<td>4T4R</td>
<td>3.6GHz - 3.7GHz</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Foxconn RP0N-7800</td>
<td>4T4R</td>
<td>3.7GHz - 3.8GHz (outdoors)</td>
<td>N, New</td>
<td></td>
</tr>
<tr>
<td>Fujitsu MU-MIMO</td>
<td>32T32R</td>
<td>3.7GHz - 3.8GHz</td>
<td>N, New</td>
<td></td>
</tr>
<tr>
<td>Foxconn RPQN-4800E</td>
<td>4T4R</td>
<td>CBRS 3.55GHz - 3.7GHz, indoor</td>
<td>Y, New</td>
<td></td>
</tr>
</tbody>
</table>

1.3.2.2.7.3 Supported UE

<table>
<thead>
<tr>
<th>UE</th>
<th>Configuration</th>
<th>Peak Tput</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>SU-MIMO 4DL, 1UL</td>
<td>DL NA UL NA</td>
<td>Y</td>
</tr>
<tr>
<td>Camera FourFaith Camera F-SC241-216-5G (EU)</td>
<td>SU-MIMO 4DL, 2UL</td>
<td>DL UL</td>
<td>Y</td>
</tr>
<tr>
<td>Handset OnePlus Nord 5G AC2003 EU/UK Model</td>
<td>SU-MIMO 4DL, 1UL</td>
<td>DL 850Mbps UL 55 Mbps</td>
<td>Y</td>
</tr>
<tr>
<td>Handset Oppo Reno 5G</td>
<td>SU-MIMO 4DL, 1UL</td>
<td>DL 850Mbps UL 55 Mbps</td>
<td>Y</td>
</tr>
<tr>
<td>Handset Samsung S22</td>
<td>SU-MIMO 4DL, 1UL</td>
<td>DL NA UL NA</td>
<td>Y</td>
</tr>
<tr>
<td>Handset Samsung S23</td>
<td>SU-MIMO 4DL, 1UL</td>
<td>DL NA UL NA</td>
<td>Y</td>
</tr>
</tbody>
</table>

1.3.3. cuPHY System Overview

Aerial cuPHY is a software-defined workload hosted on NVIDIA-certified EGX servers and a stack that uses the CUDA OS platform and GPU/NIC/CPU firmware and toolkits. This section highlights the Aerial cuPHY workload configuration interdependencies as part of the NVIDIA platform stack.
1.3.3.1 Highlights

- Grace Hopper MGX system supports 20 4T4R Peak cells / 20 4T4R average BFP9 cells
- Supports Massive MIMO: 64T64R (16DL | 8UL) @ 100MHz w/ SRS-based Beamforming
- Dell R750 with A100X supports 5 4T4R peak and 10 4T4R average BFP9 cells

1.3.3.2 Aerial CUDA-Accelerated RAN Overall Platform Qualification

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grace Hopper MGX Platform</td>
<td><img src="72-core%20NVIDIA%20Grace%20CPU" alt="72-core NVIDIA Grace CPU" /></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td><img src="NVIDIA%20H100%20Tensor%20Core%20GPU" alt="NVIDIA H100 Tensor Core GPU" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="480GB%20of%20LPDDR5X%20memory%20with%20ECC" alt="480GB of LPDDR5X memory with ECC" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="Supports%2096GB%20of%20HBM3" alt="Supports 96GB of HBM3" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="BF3%20NIC%20x2" alt="BF3 NIC x2" /></td>
<td></td>
</tr>
<tr>
<td>Platform for Converged Accelerator</td>
<td><img src="Dell%20R750" alt="Dell R750" /></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td><img src="Server%20Skew%2010-AYCG" alt="Server Skew 10-AYCG" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="Intel%20Xeon%20Gold%206336Y%202.4G,%2024C/48T" alt="Intel Xeon Gold 6336Y 2.4G, 24C/48T" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="PCle%20Gen4" alt="PCle Gen4" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="Memory%20512GB%20DDR4" alt="Memory 512GB DDR4" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="Storage%202TB" alt="Storage 2TB" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="GPU%20%2B%20NIC%20A100X" alt="GPU+NIC A100X" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="BF3%20NIC" alt="BF3 NIC" /></td>
<td></td>
</tr>
</tbody>
</table>

1.3.4. Operations, Administration, and Management (OAM) Guide

The Operations, Administration, and Management (OAM) guide covers Aerial OAM capabilities that include startup configuration using YAML configuration files, run-time configuration and status using remote procedure calls, high performance logging, and metrics reporting using the Prometheus framework.
1.3.4.1 OAM Operation

1.3.4.1.1 Cloud Native DevOps

Aerial CUDA-Accelerated RAN is based on cloud-native principles and supports a DevOps work-flow using industry standard tools such as Kubernetes, gRPC, and Prometheus.

1.3.4.1.2 Aerial Applications

The Aerial framework includes three primary applications for end to end L1 implementation and testing.

- **cuphycontroller** is the full L1 stack application. This application implements the adaptation layer from L2 to the cuPHY API, orchestrates the cuPHY API scheduling, and sends/receives ORAN compliant Fronthaul traffic over the NIC. Several independently configurable adaptation layers from L2 to the cuPHY API are available.

- **test_mac** application, for integration testing, implements a mock L2 that is capable of interfacing with cuphycontroller over the L2/L1 API.

- **ru-emulator** application, for integration testing, implements a mock O-RU + UE that is capable of interfacing with cuphycontroller over the ORAN compliant Fronthaul interface.

Every Aerial application supports the following:

- Configuration at startup through the use of YAML-format configuration files.
- Support for optionally-configured cloud-based logging and metrics backends.
- Support for optionally-deployed OAM clients for run-time configuration and status queries.
- When deployed as a Kubernetes pod:
  - Support for application monitoring and configuration auto-discovery through the Kubernetes API.
Configuration YAML files can optionally be mounted as a Kubernetes ConfigMap, separating the container image from the configuration.

Configuration YAML files can optionally be templatized using the Kubernetes kustomization.yaml format.

1.3.4.1.3 Deployment Scenarios

1.3.4.1.3.1 Functional Testing

For real-time functional correctness testing, test cases are generated offline in HDF5 binary file format, then played back in real-time through the testMAC and RU Emulator applications. The Aerial cuPHY-CP + cuPHY components under test, run in real-time to exercise GPU and Fronthaul Network interfaces. Test case sequencing is enabled through configurable launch pattern files read by testMAC and RU Emulator. The diagram below shows an example of downlink functional testing:

The diagram below shows an example of uplink functional testing:
1.3.4.1.3.2 End to End Testing

A variety of end to end testing scenarios are possible. Shown below is one example using an Aerial gNB system implementing the CU+DU, an ORAN compliant RU connected to the DU via the ORAN fronthaul interface, and UE test equipment from Keysight.

Another example is the all-digital eCPRI topology is shown below with an Aerial gNB system implementing the CU+DU with the Keysight test equipment implementing the O-RU and UE functions.

1.3.4.2 Fault Management

1.3.4.2.1 Logging

Aerial follows the best practices of Kubernetes (https://kubernetes.io/docs/concepts/cluster-administration/logging/) for implementing logging.

The cuphycontroller application outputs log messages, where the log level is less than or equal to the `nvlog.console_log_level` cuphycontroller YAML configuration parameter, directly to stdout using the logging at the node level pattern:
For high performance logs, Aerial uses a shared memory logger to offload the I/O bottleneck from the real-time threads. Log messages, where level is less than or equal to the `nvlog.shm_log_level` cuphycontroller YAML configuration parameter, are output to the shared memory logger. The shared memory logger outputs can be retrieved using either the **streaming sidecar** pattern with logs written directly to the local disk:

Or the **sidecar with logging agent** pattern to stream directly to an external logging backend:
1.3.4.2.2 nvlog message format

Each nvlog message is a string of the form “[Software Component Name] Msg” prefixed with the following space-separated optional fields:

- Date
- Timestamp
- Primary or Secondary nvlog process
- Log level
- Log event code id
- Log event code string
- CPU core number the calling thread is running on
- 64-bit sequence number
- Thread ID
- Thread Name

These fields are enabled in the nvlog_config.yaml.

An example nvlog message is:

```
20:58:09.036299 C [NVLOG.CPP] nvlog_create: name=phy shm_level=1
console_level=1 max_file_size=0x10000000 shm_cache_size=0x200000
log_buf_size=1024 prefix_opts=0x09
```

The message above had the following prefaces enabled:

- Timestamp
- Log level

Here are three more example nvlog messages, where all prefixed fields are enabled, taken at the start of the cuphycontroller process execution:

```
2021-09-15 21:29:22.926521 P C 0 SUCESS 1 0 140699056300032
cuphycontroller [NVLOG.CPP] nvlog_create: name=phy shm_level=1
console_level=1 max_file_size=0x10000000 shm_cache_size=0x200000
log_buf_size=1024 prefix_opts=0xFF
```

```
2021-09-15 21:29:22.926560 P C 0 SUCCESS 22 2 140699056300032
cuphycontroller [CTL.YAML] Standalone mode: No
```

Here is an example of an nvlog message at Fault level with Event Code AERIAL_MEMORY_EVENT:

```
20:58:09.036299 F MEMORY_EVENT Unable to allocate memory for FH buffers
```

The message above had the following prefaces enabled:

- Timestamp
- Log level
- Log event code string
The fields are further described herein:

Date is YYYY-MM-DD format, for example, 1970-01-01

Timestamp is HH:MM:SS.us, for example, 20:58:09.036299

Primary process is P, secondary process is S.

Log level is:

- F - Fatal
- E - Error
- C - Console
- W - Warning
- I - Info
- D - Debug
- V - Verbose

Log event code string or log event code id is a string (or a numerical id) that indicates the category of event that has occurred.

### 1.3.4.2.3 nvlog Components

Aerial implements the following default logging component tags:

**nvlog component:**

- 10: "NVLOG"
- 11: "NVLOG.TEST"
- 12: "NVLOG.ITAG"

**nvipc component:**

- 30: "NVIPC"

**cuPHY-CP Controller component:**

- 100: "CTL"
- 101: "CTL.SCF"
- 102: "CTL.ALTRAN"
- 103: "CTL.DRV"
- 104: "CTL.YAML"

**cuPHY-CP driver component:**

- 200: "DRV"
- 201: "DRV.SA"
- 202: "DRV.TIME"
- 203: "DRV.CTX"
- 204: "DRV.API"
- 205: "DRV.FH"
206: “DRV.GEN_CUDA”
207: “DRV.GPUDEV”
208: “DRV.PHYCH”
209: “DRV.TASK”
210: “DRV.WORKER”
211: “DRV.DLBUF”
212: “DRV.CSIRS”
213: “DRV.PBCH”
214: “DRV.PDCCH_DL”
215: “DRV.PDSCH”
216: “DRV.MAP_DL”
217: “DRV.FUNC_DL”
218: “DRV.HARQ_POOL”
219: “DRV.ORDER_CUDA”
220: “DRV.ORDER_ENTITY”
221: “DRV.PRACH”
222: “DRV.PUCCH”
223: “DRV.PUSCH”
224: “DRV.MAP_UL”
225: “DRV.FUNC_UL”
226: “DRV.ULBUF”
227: “DRV.MPS”
228: “DRV.METRICS”
229: “DRV.MEMFOOT”
230: “DRV.CELL”

cuPHY-CP cuphyl2adapter component:
230: “L2A”
231: “L2A.MAC”
232: “L2A.MACFACT”
233: “L2A.PROXY”
234: “L2A.EPOLL”
235: “L2A.TRANSPORT”
236: “L2A.MODULE”
237: “L2A.TICK”
238: “L2A.UEMD”

cuPHY-CP scfl2adapter component:
cuPHY-CP testMAC component:
- 400: "MAC"
- 401: "MAC.LP"
- 402: "MAC.FAPI"
- 403: "MAC.UTILS"
- 404: "MAC.SCF"
- 405: "MAC.ALTRAN"
- 406: "MAC.CFG"
- 407: "MAC.PROC"

cuPHY-CP ru-emulator component:
- 500: "RU"
- 501: "RU.EMULATOR"
- 502: "RU.PARSER"

cuPHY-CP aerial-fh-driver component:
- 600: "FH"
- 601: "FH.FLOW"
- 602: "FH.FH"
- 603: "FH.GPU_MP"
- 604: "FH.LIB"
- 605: "FH.MEMREG"
- 606: "FH.METRICS"
- 607: "FH.NIC"
- 608: "FH.PDUMP"
- 609: "FH.PEER"
- 610: "FH.QUEUE"
- 611: "FH.RING"
- 612: "FH.TIME"

cuPHY-CP compression_decompression component:
- 700: "COMP"
cuPHY-CP cuphyoam component:

▶ 800: “OAM”

cuPHY component:

▶ 900: “CUPHY”

Note: These strings can be changed using the nvlog_config.yaml.

1.3.4.2.4 Event codes

The following is the list of event codes (see aerial_event_code.h). The event strings match the event code names, minus the AERIAL_.

| AERIAL_SUCCESS = 0, |
| AERIAL_INVALID_PARAM_EVENT = 1, |
| AERIAL_INTERNAL_EVENT = 2, |
| AERIAL_CUDA_API_EVENT = 3, |
| AERIAL_DPDK_API_EVENT = 4, |
| AERIAL_THREAD_API_EVENT = 5, |
| AERIAL_CLOCK_API_EVENT = 6, |
| AERIAL_NVIPC_API_EVENT = 7, |
| AERIAL_ORAN_FH_EVENT = 8, |
| AERIAL_CUPHYDRV_API_EVENT = 9, |
| AERIAL_INPUT_OUTPUT_EVENT = 10, |
| AERIAL_MEMORY_EVENT = 11, |
| AERIAL_YAML_PARSER_EVENT = 12, |
| AERIAL_NVLOG_EVENT = 13, |
| AERIAL_CONFIG_EVENT = 14, |
| AERIAL_FAPI_EVENT = 15, |
| AERIAL_NO_SUPPORT_EVENT = 16, |
| AERIAL_SYSTEM_API_EVENT = 17, |
| AERIAL_L2ADAPTER_EVENT = 18, |
| AERIAL_RU_EMULATOR_EVENT = 19, |

1.3.4.3 OAM Configuration

1.3.4.3.1 Startup Configuration (cuphycontroller)

The application binary name for the combined cuPHY-CP + cuPHY is cuphycontroller. When cuphycontroller starts, it reads static configuration from configuration YAML files. This section describes the fields in the YAML files.
1.3.4.3.1.1 l2adapter_filename

This field contains the filename of the YAML-format config file for l2 adapter configuration.

1.3.4.3.1.2 aerial_metrics_backend_address

Aerial Prometheus metrics backend address.

1.3.4.3.1.3 low_priority_core

CPU core shared by all low-priority threads, isolated CPU core is preferred. Can be non-isolated CPU core but make sure no other heavy load task on it.

1.3.4.3.1.4 nic_tput_alert_threshold_mbps

This parameter is used to monitor NIC throughput. The units are in Mbps, that is, 85000 = 85 Gbps. This value is almost the max throughput that can be achieved with accurate send scheduling for a 100 Gbps link. A gRPC client (reference: $cuBB_SDK/cuPHY-CP/cuphyoam/examples/test_grpc_push_notification_client.cpp) needs to be implemented to receive the alert.

1.3.4.3.1.5 cuphydriver_config

This container holds configuration for cuphydriver.

1.3.4.3.1.6 standalone

0 - run cuphydriver integrated with other cuPHY-CP components
1 - run cuphydriver in standalone mode (no l2adapter, etc)

1.3.4.3.1.7 validation

Enables additional validation checks at run-time.

0 - Disabled
1 - Enabled
1.3.4.3.1.8 numSlots

Number of slots to run in cuphydriver standalone test.

1.3.4.3.1.9 logLevel

cuPHYDriver log level: DBG, INFO, ERROR.

1.3.4.3.1.10 profilerSec

Number of seconds to run the CUDA profiling tool.

1.3.4.3.1.11 dpdkThread

Sets the CPU core used by the primary DPDK thread. It does not have to be an isolated core. And the DPDK thread itself is defaulted to 'SCHED_FIFO+priority 95'.

1.3.4.3.1.12 dpdkVerboseLogs

Enable maximum log level in DPDK.
0 - Disable
1 - Enable

1.3.4.3.1.13 accuTxSchedResNs

Sets the accuracy of the accurate transmit scheduling, in units of nanoseconds.

1.3.4.3.1.14 accuTxSchedDisable

Disable accurate TX scheduling.
0 - packets are sent according to the TX timestamp
1 - packets are sent whenever it is convenient

1.3.4.3.1.15 fhStatsDumpCpuCore

Sets the CPU core used by the FH stats logging thread. It does not have to be an isolated core. And currently the default FH stats polling interval is 500ms.
1.3.4.3.1.16 pdump_client_thread

CPU core to use for pdump client. Set to -1 to disable fronthaul RX traffic PCAP capture.

See:
2. aerial-fh README.md

1.3.4.3.1.17 mps_sm_pusch

Number of SMs for PUSCH channel.

1.3.4.3.1.18 mps_sm_pucch

Number of SMs for PUCCH channel.

1.3.4.3.1.19 mps_sm_pusch

Number of SMs for PUSCH channel.

1.3.4.3.1.20 mps_sm_prach

Number of SMs for PRACH channel.

1.3.4.3.1.21 mps_sm_ul_order

Number of SMs for UL order kernel.

1.3.4.3.1.22 mps_sm_pdsch

Number of SMs for PDSCH channel.

1.3.4.3.1.23 mps_sm_pdcch

Number of SMs for PDCCH channel.
1.3.4.3.1.24  mps_sm_pbch

Number of SMs for PBCH channel.

1.3.4.3.1.25  mps_sm_srs

Number of SMs for SRS channel.

1.3.4.3.1.26  mps_sm_gpu_comms

Number of SMs for GPU comms.

1.3.4.3.1.27  nics

Container for NIC configuration parameters.

1.3.4.3.1.28  nic

PCIe bus address of the NIC port.

1.3.4.3.1.29  mtu

Maximum transmission size, in bytes, supported by the Fronthaul U-plane and C-plane.

1.3.4.3.1.30  cpu_mbufs

Number of preallocated DPDK memory buffers (mbufs) used for Ethernet packets.

1.3.4.3.1.31  uplane_tx_handles

The number of pre-allocated transmit handles that link the U-plane prepare() and transmit() functions.

1.3.4.3.1.32  txq_count

NIC transmit queue count.
Must be large enough to handle all cells attached to this NIC port.
Each cell uses one TXQ for C-plane and txq_count_uplane TXQs for U-plane.
1.3.4.3.1.33  rxq_count

Receive queue count.
This value must be large enough to handle all cell attached to this NIC port.
Each cell uses one RXQ to receive all uplink traffic.

1.3.4.3.1.34  txq_size

Number of packets that can fit in each transmit queue.

1.3.4.3.1.35  rxq_size

Number of packets that can be buffered in each receive queue.

1.3.4.3.1.36  gpu

CUDA device to receive uplink packets from this NIC port.

1.3.4.3.1.37  gpus

List of GPU device IDs. To use gpudirect, the GPU must be on the same PCIe root complex as the NIC.
To maximize performance, the GPU should be on the same PCIe switch as the NIC. Only the first entry
in the list is used.

1.3.4.3.1.38  workers_ul

List of pinned CPU cores used for uplink worker threads.

1.3.4.3.1.39  workers_dl

List of pinned CPU cores used for downlink worker threads.

1.3.4.3.1.40  debug_worker

For performance debug purpose, this is set to a free core to work with the enable_*_tracing logs.
1.3.4.3.1.41  workers_sched_priority

cuPHYDriver worker threads scheduling priority.

1.3.4.3.1.42  dpdk_file_prefix

Shared data file prefix to use for the underlying DPDK process.

1.3.4.3.1.43  wfreq

Filename containing the coefficients for channel estimation filters, in HDF5 (.h5) format.

1.3.4.3.1.44  cell_group

Enable cuPHY cell groups.

0 - disable 1 - enable

1.3.4.3.1.45  cell_group_num

Number of cells to be configured in L1 for the test.

1.3.4.3.1.46  enable_h2d_copy_thread

Enable/disable offloading of h2d copy in L2A to a separate copy thread.

1.3.4.3.1.47  h2d_copy_thread_cpu_affinity

CPU core on which the h2d copy thread in L2A should run. Applicable only if enable_h2d_copy_thread is 1.

1.3.4.3.1.48  h2d_copy_thread_sched_priority

h2d copy thread priority in L2A. Applicable only if enable_h2d_copy_thread is 1.

1.3.4.3.1.49  fix_beta_dl

Fix the beta_dl for local test with RU Emulator so that the output values are a bytematch to the TV.
1.3.4.3.1.50 prometheus_thread

Pinned CPU core for updating NIC metrics once per second.

1.3.4.3.1.51 start_section_id_srs

ORAN CUS start section ID for the SRS channel.

1.3.4.3.1.52 start_section_id_prach

ORAN CUS start section ID for the PRACH channel.

1.3.4.3.1.53 enable_ul_cuphy_graphs

Enable UL processing with CUDA graphs.

1.3.4.3.1.54 enable_dl_cuphy_graphs

Enable DL processing with CUDA graphs.

1.3.4.3.1.55 section_3_time_offset

Time offset, in units of nanoseconds, for the PRACH channel.

1.3.4.3.1.56 ul_order_timeout_cpu_ns

Timeout, in units of nanoseconds, for the uplink order kernel to receive any U-plane packets for this slot.

1.3.4.3.1.57 ul_order_timeout_gpu_ns

Timeout, in units of nanoseconds, for the order kernel to complete execution on the GPU.

1.3.4.3.1.58 pusch_sir

Enable pusch sir calculation (0 by default).
1.3.4.3.1.59  pusch_rssi

Enable PUSCH RSSI calculation (0 by default).

1.3.4.3.1.60  pusch_tdi

Enable PUSCH TDI processing (0 by default).

1.3.4.3.1.61  pusch_cfo

Enable PUSCH CFO calculations (0 by default).

1.3.4.3.1.62  pusch_dftsofdm

DFT-s-OFDM enable/disable flag: 0 - disable, 1 - enable.

1.3.4.3.1.63  pusch_to

It is only used for timing offset reporting to L2. If the timing offset estimate is not used by L2, it can be disabled.

1.3.4.3.1.64  pusch_select_eqcoeffalgo

Algorithm selector for PUSCH noise interference estimation and channel equalization. The following values are supported: 0: Regularized zero-forcing (RZF) 1: Diagonal MMSE regularization 2: Minimum Mean Square Error - Interference Rejection Combining (MMSE-IRC) 3: MMSE-IRC with RBLW covariance shrinkage 4: MMSE-IRC with OAS covariance shrinkage.

1.3.4.3.1.65  pusch_select_chestalgo

Channel estimation algorithm selection: 0 - legacy MMSE, 1 - multi-stage MMSE with delay estimation.

1.3.4.3.1.66  pusch_tbsizecheck

Tb size verification enable/disable flag: 0 - disable, 1 - enable.
1.3.4.3.1.67 pusch_deviceGraphLaunchEn

Static flag to allow device graph launch in PUSCH.

1.3.4.3.1.68 pusch_waitTimeOutPreEarlyHarqUs

Timeout threshold in microseconds for receiving OFDM symbols for PUSCH early-HARQ processing.

1.3.4.3.1.69 pusch_waitTimeOutPostEarlyHarqUs

Timeout threshold in microseconds for receiving OFDM symbols for PUSCH non-early-HARQ processing (essentially all the PUSCH symbols).

1.3.4.3.1.70 pusch_polarDcdrListSz

List size used in List Decoding of Polar codes.

1.3.4.3.1.71 enable_cpu_task_tracing

The flag is used to trace and instrument DL/UL CPU tasks running on existing cuphydriver cores.

1.3.4.3.1.72 enable_prepare_tracing

It's for tracing the U-plane packet preperation kernel durations and end times and need the debug worker to be enabled.

1.3.4.3.1.73 enable_dl_cqe_tracing

Enables tracing of DL CQEs (debug feature to check for DL U-plane packets' timing at the NIC).

1.3.4.3.1.74 ul_rx_pkt_tracing_level

This YAML param can be set to 3 different values: 0 (default, recommended) : Only keeps count of the early/ontime/late packet counters per slot as seen by the DU (Reorder kernel) for the Uplink U-plane packets. 1 : Also Captures and logs earliest/latest packet timestamp per symbol per slot as seen by the DU. 2 : Also Captures and logs timestamp of each packet received per symbol per slot as seen by the DU.
1.3.4.3.1.75  split_ul_cuda_streams

Keep default of 0. This allows back to back UL slots to overlap their processing. Keep disabled to maintain performance of first UL slot in every group of 2.

1.3.4.3.1.76  aggr_obj_non_avail_th

Keep the default value at 5. This param sets the threshold for successive non-availability of L1 objects (can be interpreted as L1 handler necessary to schedule PHY compute tasks to the GPU). Unavailability could imply the execution timeline falling behind the expected L1 timeline budget.

1.3.4.3.1.77  dl_wait_th_ns

This parameter is used for error handling in the event of GPU failure. You must keep the defaults.

1.3.4.3.1.78  sendCPlane_timing_error_th_ns

Keep the default value at 50000 (50 us). The threshold is used as a check for the proximity of the current time during C-plane task’s execution to the actual scheduled C-plane packet’s transmission time. Meeting the threshold check would result in C-plane packet transmission being dropped for the slot.

1.3.4.3.1.79  pusch_forcedNumCsi2Bits

Debug feature if > 0, overrides the number of PUSCH CSI-P2 bits for all CSI-P2 UCIs with the non-zero value provided. Recommend setting it to 0.

1.3.4.3.1.80  mMIMO_enable

Keep at default of 0. This flag is reserved for future capability.

1.3.4.3.1.81  enable_srs

Enable/disable SRS

1.3.4.3.1.82  enable_csip2_v3

Enable/disable the the support of CSI part2 defined by FAPI 10.03 Table 3-77
1.3.4.3.1.83 pusch_aggr_per_ctx

Number of PUSCH objects per context (3 by default).

1.3.4.3.1.84 prach_aggr_per_ctx

Number of PRACH objects per context (2 by default).

1.3.4.3.1.85 pucch_aggr_per_ctx

Number of PUCCH objects per context (4 by default).

1.3.4.3.1.86 srs_aggr_per_ctx

Number of SRS objects per context (2 by default).

1.3.4.3.1.87 ul_input_buffer_per_cell

Number of UL buffers allocated per cell (10 by default).

1.3.4.3.1.88 ul_input_buffer_per_cell_srs

Number of UL buffers allocated per cell for SRS (4 by default).

1.3.4.3.1.89 ue_mode

Flag for spectral efficiency feature. Must be enabled on the RU side YAML to emulate UE operation.

1.3.4.3.1.90 cplane_disable

Disable C-plane for all cells.
0 - Enable C-plane 1 - Disable C-plane

1.3.4.3.1.91 cells

List of containers of cell parameters.
1.3.4.3.1.92 name

Name of the cell

1.3.4.3.1.93 cell_id

ID of the cell.

1.3.4.3.1.94 src_mac_addr

Source MAC address for U-plane and C-plane packets. Set to 00:00:00:00:00:00 to use the MAC address of the NIC port in use.

1.3.4.3.1.95 dst_mac_addr

Destination MAC address for U-plane and C-plane packets.

1.3.4.3.1.96 nic

gNB NIC port to which the cell is attached. Must match the ‘nic’ key value in one of the elements of in the ‘nics’ list.

1.3.4.3.1.97 vlan

VLAN ID used for C-plane and U-plane packets.

1.3.4.3.1.98 pcp

QoS priority codepoint used for C-plane and U-plane Ethernet packets.

1.3.4.3.1.99 txq_count_uplane

Number of transmit queues used for U-plane.

1.3.4.3.1.100 eAxC_id_ssb_pbch

List of eAxC IDs to use for SSB/PBCH.
1.3.4.3.1.101  eAxC_id_pdcch
List of eAxC IDs to use for PDCCH.

1.3.4.3.1.102  eAxC_id_pdsch
List of eAxC IDs to use for PDSCH.

1.3.4.3.1.103  eAxC_id_csirs
List of eAxC IDs to use for CSI RS.

1.3.4.3.1.104  eAxC_id_pusch
List of eAxC IDs to use for PUSCH.

1.3.4.3.1.105  eAxC_id_pucch
List of eAxC IDs to use for PUCCH.

1.3.4.3.1.106  eAxC_id_srs
List of eAxC IDs to use for SRS.

1.3.4.3.1.107  eAxC_id_prach
List of eAxC IDs to use for PRACH.

1.3.4.3.1.108  dl_iq_data_fmt:comp_meth
DL U-plane compression method: 0: Fixed point 1: BFP

1.3.4.3.1.109  dl_iq_data_fmt:bit_width
Number of bits used for each RE on DL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16
1.3.4.3.1.110  ul_iq_data_fmt:comp_meth

UL U-plane compression method: 0: Fixed point 1: BFP

1.3.4.3.1.111  ul_iq_data_fmt:bit_width

Number of bits used per RE on uplink U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

1.3.4.3.1.112  fs_offset_dl

Downlink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.113  exponent_dl

Downlink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.114  ref_dl

Downlink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.115  fs_offset_ul

Uplink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.116  exponent_ul

Uplink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.117  max_amp_ul

Maximum full scale amplitude used in uplink U-plane scaling per ORAN CUS 6.1.3.

1.3.4.3.1.118  mu

3GPP subcarrier bandwidth index ‘mu’.
0 - 15 kHz 1 - 30 kHz 2 - 60 kHz 3 - 120 kHz 4 - 240 kHz
1.3.4.3.1.119 $T_{1a\_max\_up\_ns}$
Scheduled timing advance before time-zero for downlink U-plane egress from DU, per ORAN CUS.

1.3.4.3.1.120 $T_{1a\_max\_cp\_ul\_ns}$
Scheduled timing advance before time-zero for uplink C-plane egress from DU, per ORAN CUS.

1.3.4.3.1.121 $T_{a4\_min\_ns}$
Start of DU reception window after time-zero, per ORAN CUS.

1.3.4.3.1.122 $T_{a4\_max\_ns}$
End of DU reception window after time-zero, per ORAN CUS.

1.3.4.3.1.123 $T_{cp\_adv\_dl\_ns}$
Downlink C-plane timing advance ahead of U-plane, in units of nanoseconds, per ORAN CUS.

1.3.4.3.1.124 $u_l\_u\_plane\_tx\_offset\_ns$
Flag for spectral efficiency feature. Must be set on the RU side YAML to offset UL transmission start from $T_0$.

1.3.4.3.1.125 $pusch\_prb\_stride$
Memory stride, in units of PRBs, for the PUSCH channel. Affects GPU memory layout.

1.3.4.3.1.126 $prach\_prb\_stride$
Memory stride, in units of PRBs, for the PRACH channel. Affects GPU memory layout.

1.3.4.3.1.127 $srs\_prb\_stride$
Memory stride, in units of PRBs, for the SRS. Affects GPU memory layout.
1.3.4.3.1.128 pusch_ldpc_max_num_itr_algo_type

0 - Fixed LDPC iteration count
1 - MCS based LDPC iteration count
Recommend setting pusch_ldpc_max_num_itr_algo_type: 1

1.3.4.3.1.129 pusch_fixed_max_num_ldpc_itrs

Unused currently, reserved to replace pusch_ldpc_n_iterations.

1.3.4.3.1.130 pusch_ldpc_n_iterations

Iteration count is set to pusch_ldpc_n_iterations, when the fixed LDPC iteration count option is selected (pusch_ldpc_max_num_itr_algo_type:0). Because the default value of pusch_ldpc_max_num_itr_algo_type is 1 (iteration count optimized based on MCS), pusch_ldpc_n_iterations is unused.

1.3.4.3.1.131 pusch_ldpc_algo_index

Algorithm index for LDPC decoder: 0 - automatic choice.

1.3.4.3.1.132 pusch_ldpc_flags

pusch_ldpc_flags are flags that configure the LDPC decoder. pusch_ldpc_flags:2 selects an LDPC decoder that optimizes for throughput i.e. processes more than one codeword (for example, 2) instead of latency.

1.3.4.3.1.133 pusch_ldpc_use_half

Indication of input data type of LDPC decoder:
0 - single precision, 1 - half precision

1.3.4.3.1.134 pusch_nMaxPrb

This is for memory allocation of max PRB range of peak cells compared to average cells.
1.3.4.3.1.135 ul_gain_calibration

UL Configured Gain used to convert dBFS to dBm. Default value, if unspecified: 48.68

1.3.4.3.1.136 lower_guard_bw

Lower Guard Bandwidth expressed in kHZ. Used for deriving freqOffset for each Rach Occasion. Default is 845.

1.3.4.3.1.137 tv_pusch

HDF5 file containing static configuration (for example, filter coefficients) for the PUSCH channel.

1.3.4.3.1.138 tv_prach

HDF5 file containing static configuration (for example, filter coefficients) for the PRACH channel.

1.3.4.3.1.139 pusch_ldpc_n_iterations

PUSCH LDPC channel coding iteration count.

1.3.4.3.1.140 pusch_ldpc_early_termination

PUSCH LDPC channel coding early termination.
0 - Disable 1 - Enable

1.3.4.3.2 Startup Configuration (l2_adapter_config)

1.3.4.3.2.1 msg_type

Defines the L2/L1 interface API. Supported options are:
▶ scf_fapi_gnb - Use the small cell forum API.

1.3.4.3.2.2 phy_class

Same as msg_type.
1.3.4.3.2.3 tick_generator_mode

The SLOT.tickication interval generator mode:
0 - poll + sleep. During each tick the threads sleep some time to release the CPU core to avoid hanging
the system, then they poll the system time. 1 - sleep. Sleep to absolute timestamp, no polling. 2 -
timer_fd. Start a timer and call epoll_wait() on the timer_fd.

1.3.4.3.2.4 allowed_fapi_latency

Allowed maximum latency of SLOT FAPI messages, which send from L2 to L1, otherwise the message
is ignored and dropped.
Unit: slot. Default is 0, it means L2 message should be received in current slot.

1.3.4.3.2.5 allowed_tick_error

Allowed tick interval error.
Unit: us
Tick interval error is printed in statistic style. If observed tick error > allowed, the log is printed as Error
level.

1.3.4.3.2.6 timer_thread_config

Configuration for the timer thread.

1.3.4.3.2.7 name

Name of thread.

1.3.4.3.2.8 cpu_affinity

Id of pinned CPU core used for timer thread.

1.3.4.3.2.9 sched_priority

Scheduling priority of timer thread.
1.3.4.3.2.10  message_thread_config

Configuration container for the L2/L1 message processing thread.

1.3.4.3.2.11  name

Name of thread.

1.3.4.3.2.12  cpu_affinity

Id of pinned CPU core used for timer thread.

1.3.4.3.2.13  sched_priority

Scheduling priority of message thread.

1.3.4.3.2.14  ptp

ptp configs for GPS_ALPHA, GPS_BETA.

1.3.4.3.2.15  gps_alpha

GPS Alpha value for ORAN WG4 CUS section 9.7.2. Default value = 0, if undefined.

1.3.4.3.2.16  gps_beta

GPS Beta value for ORAN WG4 CUS section 9.7.2. Default value = 0, if undefined.

1.3.4.3.2.17  mu_highest

Highest supported mu, used for scheduling TTI tick rate.

1.3.4.3.2.18  slot_advance

Timing advance ahead of time-zero, in units of slots, for L1 to notify L2 of a slot request.
1.3.4.3.2.19  enableTickDynamicSfnSlot

Enable dynamic slot/sfn.

1.3.4.3.2.20  staticPucchSlotNum

Debugging param for testing against RU Emulator to send set static PUCCH slot number.

1.3.4.3.2.21  staticPuschSlotNum

Debugging param for testing against RU Emulator to send set static PUSCH slot number.

1.3.4.3.2.22  staticPdschSlotNum

Debugging param for testing against RU Emulator to send set static PDSCH slot number.

1.3.4.3.2.23  staticPdcchSlotNum

Debugging param for testing against RU Emulator to send set static PDCCH slot number.

1.3.4.3.2.24  staticCsiRsSlotNum

Debugging param for testing against RU Emulator to send set static CSI-RS slot number.

1.3.4.3.2.25  staticSsbSlotNum

Override the incoming slot number with the YAML configured SlotNumber for SS/PBCH.  
Example  
staticSsbSlotNum:10

1.3.4.3.2.26  staticSsbPcid

Debugging param for testing against RU Emulator to send set static SSB phycellId.
1.3.4.3.27 staticSsbSFN

Debugging param for testing against RU Emulator to send set static SSB SFN.

1.3.4.3.28 pucch_dtx_thresholds

Array of scale factors for DTX Thresholds of each PUCCH format.
Default value, if not present, is 1.0, which means the thresholds are not scaled.
For PUCCH format 0 and 1, -100.0 is replaced with 1.0.
Example:
pucch_dtx_thresholds: [-100.0, -100.0, 1.0, 1.0, -100.0]

1.3.4.3.29 pusch_dtx_thresholds

Scale factor for DTX Thresholds of UCI on PUSCH.
Default value, if not present, is 1.0, which means the threshold is not scaled.
Example:
pusch_dtx_thresholds: 1.0

1.3.4.3.30 enable_precoding

Enable/Disable Precoding PDUs to be parsed in L2Adapter.
Default value is 0 enable_precoding: 0/1

1.3.4.3.31 prepone_h2d_copy

Enable/Disable preponing of H2D copy in L2Adapter.
Default value is 1 prepone_h2d_copy: 0/1

1.3.4.3.32 enable_beam_forming

Enables/Disables BeamIds to parsed in L2Adapter.
Default value : 0 enable_beam_forming: 1
1.3.4.3.2.33 dl_tb_loc

Transport block location in inside nvipc buffer.
Default value is 1 dl_tb_loc: 0 # TB is located in inline with nvipc’s msg buffer. dl_tb_loc: 1 # TB is located in nvipc’s CPU data buffer. dl_tb_loc: 2 # TB is located in nvipc’s GPU buffer.

1.3.4.3.2.34 instances

Container for cell instances.

1.3.4.3.2.35 name

Name of the instance.

1.3.4.3.2.36 nvipc_config_file

Config dedicated YAML file for nvipc. Example: nvipc_multi_instances.yaml

1.3.4.3.2.37 transport

Configuration container for L2/L1 message transport parameters.

1.3.4.3.2.38 type

Transport type. One of shm, dpdk, or udp.

1.3.4.3.2.39 udp_config

Configuration container for the udp transport type.

1.3.4.3.2.40 local_port

UDP port used by L1.

1.3.4.3.2.41 remote_port

UDP port used by L2.
1.3.4.3.2.42 shm_config

Configuration container for the shared memory transport type.

1.3.4.3.2.43 primary

Indicates process is primary for shared memory access.

1.3.4.3.2.44 prefix

Prefix used in creating shared memory filename.

1.3.4.3.2.45 cuda_device_id

Set this parameter to a valid GPU device ID to enable CPU data memory pool allocation in host pinned memory. Set to -1 to disable this feature.

1.3.4.3.2.46 ring_len

Length, in bytes, of the ring used for shared memory transport.

1.3.4.3.2.47 mempool_size

Configuration container for the memory pools used in shared memory transport.

1.3.4.3.2.48 cpu_msg

Configuration container for the shared memory transport for CPU messages (that is, L2/L1 FAPI messages).

1.3.4.3.2.49 buf_size

Buffer size in bytes.

1.3.4.3.2.50 pool_len

Pool length in buffers.
1.3.4.3.2.51 cpu_data

Configuration container for the shared memory transport for CPU data elements (that is, downlink and uplink transport blocks).

1.3.4.3.2.52 buf_size

Buffer size in bytes.

1.3.4.3.2.53 pool_len

Pool length in buffers.

1.3.4.3.2.54 cuda_data

Configuration container for the shared memory transport for GPU data elements.

1.3.4.3.2.55 buf_size

Buffer size in bytes.

1.3.4.3.2.56 pool_len

Pool length in buffers.

1.3.4.3.2.57 dpdk_config

Configurations for the DPDK over NIC transport type.

1.3.4.3.2.58 primary

Indicates process is primary for shared memory access.

1.3.4.3.2.59 prefix

The name used in creating shared memory files and searching DPDK memory pools.
1.3.4.3.2.60  local_nic_pci

The NIC address or name used in IPC.

1.3.4.3.2.61  peer_nic_mac

The peer NIC MAC address, only need to be set in secondary process (L2/MAC).

1.3.4.3.2.62  cuda_device_id

Set this parameter to a valid GPU device ID to enable CPU data memory pool allocation in host pinned memory. Set to -1 to disable this feature.

1.3.4.3.2.63  need_eal_init

Whether nvipc needs to call rte_eal_init() to initiate the DPDK context. 1 - initiate by nvipc; 0 - initiate by other module in the same process.

1.3.4.3.2.64  lcore_id

The logic core number for nvipc_nic_poll thread.

1.3.4.3.2.65  mempool_size

Configuration container for the memory pools used in shared memory. transport.

1.3.4.3.2.66  cpu_msg

Configuration container for the shared memory transport for CPU messages (that is, L2/L1 FAPI messages).

1.3.4.3.2.67  buf_size

Buffer size in bytes.
1.3.4.3.2.68 pool_len

Pool length in buffers.

1.3.4.3.2.69 cpu_data

Configuration container for the shared memory transport for CPU data elements (that is, downlink and uplink transport blocks).

1.3.4.3.2.70 buf_size

Buffer size in bytes.

1.3.4.3.2.71 pool_len

Pool length in buffers.

1.3.4.3.2.72 cuda_data

Configuration container for the shared memory transport for GPU data elements.

1.3.4.3.2.73 buf_size

Buffer size in bytes.

1.3.4.3.2.74 pool_len

Pool length in buffers.

1.3.4.3.2.75 app_config

Configurations for all transport types, mostly used for debug.

1.3.4.3.2.76 grpc_forward

Whether to enable forwarding nvipc messages and how many messages to be forwarded automatically from initialization. Here count = 0 means forwarding every message forever.

0: disabled; 1: enabled but doesn't start forwarding at initial; -1: enabled and start forwarding at initial with count = 0; Other positive number: enabled and start forwarding at initial with count = grpc_forward.
1.3.4.3.2.77 debug_timing

For debug only.
Whether to record timestamp of allocating, sending, receiving, releasing of all nvipc messages.

1.3.4.3.2.78 pcap_enable

For debug only.
Whether to capture nvipc messages to pcap file.

1.3.4.3.2.79 pcap_cpu_core

CPU core of background pcap log save thread.

1.3.4.3.2.80 pcap_cache_size_bits

Size of /dev/shm/${prefix}_pcap. If set to 29, size is $2^{29} = 512$MB.

1.3.4.3.2.81 pcap_file_size_bits

Max size of /dev/shm/${prefix}_pcap. If set to 31, size is $2^{31} = 2$GB.

1.3.4.3.2.82 pcap_max_data_size

Max DL/UL FAPI data size to capture reduce pcap size.

1.3.4.3.3 Startup Configuration (ru-emulator)

The application binary name for the combined O-RU + UE emulator is ru-emulator. When ru-emulator starts, it reads static configuration from a configuration YAML file. This section describes the fields in the YAML file.

1.3.4.3.3.1 core_list

List of CPU cores that RU Emulator could use.
1.3.4.3.3.2 \texttt{nic\_interface}

PCIe address of NIC to use that is, b5:00.1.

1.3.4.3.3 \texttt{peerethaddr}

MAC address of cuPHYController port.

1.3.4.3.4 \texttt{nvlog\_name}

The nvlog instance name for ru-emulator. Detailed nvlog configurations are in \texttt{nvlog\_config.yaml}.

1.3.4.3.5 \texttt{cell\_configs}

Cell configs agreed upon with DU.

1.3.4.3.6 \texttt{name}

Cell string name (largely unused).

1.3.4.3.7 \texttt{eth}

Cell MAC address.

1.3.4.3.8 \texttt{dl\_iq\_data\_fmt:comp\_meth}

DL U-plane compression method: 0: Fixed point 1: BFP

1.3.4.3.9 \texttt{dl\_iq\_data\_fmt:bit\_width}

Number of bits used for each RE on DL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

1.3.4.3.10 \texttt{ul\_iq\_data\_fmt:comp\_meth}

UL U-plane compression method: 0: Fixed point 1: BFP
1.3.4.3.3.11 ul_iq_data_fmt:bit_width

Number of bits used for each RE on UL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

1.3.4.3.3.12 flow_list

eAxC list

1.3.4.3.3.13 eAxC_prach_list

eAxC prach list

1.3.4.3.3.14 vlan

vlan to use for RX and TX

1.3.4.3.3.15 nic

Index of the nic to use in the nics list.

1.3.4.3.3.16 tti

Slot indication interval.

1.3.4.3.3.17 validate_dl_timing

Validate DL timing (need to be PTP synchronized).

1.3.4.3.3.18 timing_histogram

generate histogram

1.3.4.3.3.19 timing_histogram_bin_size

histogram bin size
1.3.4.3.3.20 oran_timing_info

1.3.4.3.3.21 dl_c_plane_timing_delay

t1a_max_up from ORAN

1.3.4.3.3.22 dl_c_plane_window_size

DL C Plane RX ontime window size.

1.3.4.3.3.23 ul_c_plane_timing_delay

T1a_max_cp_ul from ORAN.

1.3.4.3.3.24 ul_c_plane_window_size

UL C Plane RX ontime window size.

1.3.4.3.3.25 dl_u_plane_timing_delay

T2a_max_up from ORAN.

1.3.4.3.3.26 dl_u_plane_window_size

DL U Plane RX ontime window size.

1.3.4.3.3.27 ul_u_plane_tx_offset

Ta4_min_up from ORAN.

1.3.4.3.4 Run-time Configuration/Status

During run-time, Aerial components can be re-configured or queried for status through gRPC remote procedure calls (RPCs). The RPCs are defined in "protocol buffers" syntax, allowing support for clients written in any of the languages supported by gRPC and protocol buffers.

More information about gRPC may be found at: https://grpc.io/docs/what-is-grpc/core-concepts/

More information about protocol buffers may be found at: https://developers.google.com/protocol-buffers
1.3.4.3.4.1 Simple Request/Reply Flow

Aerial applications support a request/reply flow using the gRPC framework with protobufs messages. At run-time, certain configuration items may be updated and certain status information may be queried. An external OAM client interfaces with the Aerial application acting as the gRPC server.

1.3.4.3.4.2 Streaming Request/Replies

Aerial applications support the gRPC streaming feature for sending periodic status between client and server.
1.3.4.3.3 Asynchronous Interthread Communication

Certain request/reply scenarios require interaction with the high-priority CPU-pinned threads orchestrating GPU work. These interactions occur through Aerial-internal asynchronous queues, and requests are processed on a best effort basis that prioritizes the orchestration of GPU kernel launches and other L1 tasks.

1.3.4.3.4 Aerial Common Service Definition

```proto
syntax = "proto3";
package aerial;

service Common {
  rpc GetSFN (GenericRequest) returns (SFNReply) {}  
  rpc GetCpuUtilization (GenericRequest) returns (CpuUtilizationReply) {}  
  rpc SetPuschH5DumpNextCrc (GenericRequest) returns (DummyReply) {}  
}
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

case FAPIStream (FAPIStreamRequest) returns (stream FAPIStreamReply)
{}

}
message GenericRequest {
string name = 1;
}
message SFNReply {
int32 sfn = 1;
int32 slot = 2;
}
message DummyReply {
}
message CpuUtilizationPerCore {
int32 core_id = 1;
int32 utilization_x1000 = 2;
}
message CpuUtilizationReply {
repeated CpuUtilizationPerCore core = 1;
}
message FAPIStreamRequest {
int32 client_id = 1;
int32 total_msgs_requested = 2;
}
message FAPIStreamReply {
int32 client_id = 1;
bytes msg_buf = 2;
bytes data_buf = 3;
}
1.3.4.3.4.5 rpc GetCpuUtilization

The GetCpuUtilization RPC returns a variable-length array of CPU utilization per-high-priority-core. CPU utilization is available through the Prometheus node exporter, however the design approach used by Aerial high-priority threads results in a false 100% CPU core utilization per thread. This RPC allows retrieval of the actual CPU utilization of high-priority threads. High-priority threads are pinned to specific CPU cores.

1.3.4.3.4.6 rpc GetFAPIStream

This RPC requests snooping of one or more (up to infinite number) of SCF FAPI messages. The snooped messages are delivered from the Aerial gRPC server to a third party client. See cuPHY-CP/cuphyoam/examples/aerial_get_l2msgs.py for an example client.

1.3.4.3.4.7 rpc TerminateCuphycontroller

This RPC message terminates cuPHYController with immediate effect.

1.3.4.3.4.8 rpc CellParamUpdateRequest

This RPC message updates cell configuration without stopping the cell. Message specification:

```cpp
message CellParamUpdateRequest {
  int32 cell_id = 1;
  string dst_mac_addr = 2;
  int32 vlan_tci = 3;
}
```

`dst_mac_addr` must be in ‘XX:XX:XX:XX:XX:XX’ format.

`vlan_tci` must include the 16-bit TCI value of 802.1Q tag.

1.3.4.3.4.9 List of Parameters Supported by Dynamic OAM via gRPC and CONFIG.request (M-plane)

The Configuration unit is across all cells/per cell config. The Cell outage is either in-service or out-of-service.
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Configuration unit</th>
<th>Cell outage</th>
<th>OAM command</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ru_type</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -ru_type $RU_TYPE</td>
<td>$RU_TYPE : 1 for FXN_RU, 2 for FJT_RU, 3 for OTHER_RU (including multiattrs_update.py)</td>
</tr>
<tr>
<td>nic</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -nic $NIC</td>
<td>nic PCIe address. It has to be one of the nic ports configured in cuphycontroller YAML file multiattrs_update.py</td>
</tr>
<tr>
<td>dst_mac_addr</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -dst_mac_addr $DST_MAC_ADDR -vlan_id $VLAN_ID -pcp $PCP</td>
<td>dst_mac_addr, vlan id and pcp have to be updated together multiattrs_update.py</td>
</tr>
<tr>
<td>vlan_id</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -dst_mac_addr $DST_MAC_ADDR -vlan_id $VLAN_ID -pcp $PCP</td>
<td>dst_mac_addr, vlan id and pcp have to be updated together multiattrs_update.py</td>
</tr>
<tr>
<td>pcp</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -dst_mac_addr $DST_MAC_ADDR -vlan_id $VLAN_ID -pcp $PCP</td>
<td>dst_mac_addr, vlan id and pcp have to be updated together multiattrs_update.py</td>
</tr>
<tr>
<td>dl iq_data_fmt</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -dl_comp_meth $COMP_METH -dl_bit_width $BIT_WIDTH</td>
<td>multiattrs_update.py</td>
</tr>
<tr>
<td>ul iq_data_fmt</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>cd $cuBB_SDK/build/cuPHY-CP/cuphyoam &amp;&amp; python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell -server_ip $SERVER_IP -cell_id $CELL_ID -ul_comp_meth $COMP_METH -ul_bit_width $BIT_WIDTH</td>
<td>multiattrs_update.py</td>
</tr>
</tbody>
</table>

1.3. Aerial cuPHY
1.3.4.3.5 M-Plane Hybrid Mode ORAN YANG Model Provisioning

Aerial supports M-plane hybrid mode, which allows NMS/SMO, using ORAN YANG data models to pass RU capabilities, C/U–plane transport config, and U-plane config to L1.

Here is the high level sequence diagram:

Data Model Procedures-Yang data tree write procedure

Data Model Procedures-Yang data tree read procedure
1.3.4.3.5.1 Data Model Transfer APIs (gRPC ProtoBuf contract)

```protobuf
syntax = "proto3";
package p9_messages.v1;

service P9Messages {
  rpc HandleMsg (Msg) returns (Msg) {}  
}

message Msg {
  Header header = 1;
  Body body = 2;
}

message Header {
  string msg_id = 1;  // Message identifier to
                      // 1) Identify requests and notifications
                      // 2) Correlate requests and responses
  optional string oru_name = 2;  // The name (identifier) of the O-RU, if present.
  int32 vf_id = 3;  // The identifier for the FAPI VF ID
  int32 phy_id = 4;  // The identifier for the FAPI PHY ID
  optional int32 trp_id = 5;  // The identifier PHY's TRP, if any
}

message Body {
  oneof msg_body {
    Request request = 1;
    Response response = 2;
  }
}

message Request {
  oneof req_type {
    Get get = 1;
    EditConfig edit_config = 2;
  }
}

message Response  
```

(continues on next page)
{  
oneof resp_type  
  
  GetResp get_resp = 1;  
  EditConfigResp edit_config_resp = 2;  
}

message Get {  
  repeated bytes filter = 1;  
}

message GetResp  
{  
  Status status_resp = 1;  
  bytes data = 2;  
}

message EditConfig  
{  
  bytes delta_config = 1;  // List of Node changes with the associated operation to apply to the node  
}

message EditConfigResp {  
  Status status_resp = 1;  
}

message Error  
{  
  string error_type = 1;  // Error type defined in RFC 6241, Appendix B  
  string error_tag = 2;  // Error tag defined in RFC 6241, Appendix B  
  string error_severity = 3;  // Error severity defined in RFC 6241, Appendix B  
  string error_app_tag = 4;  // Error app tag defined in RFC 6241, Appendix B  
  string error_path = 5;  // Error path defined in RFC 6241, Appendix B  
  string error_message = 6;  // Error message defined in RFC 6241, Appendix B  
}

message Status  
{  
  enum StatusCode  
  {  
    OK = 0;  
    ERROR_GENERAL = 1;  
  }  
  StatusCode status_code = 1;  
  repeated Error error = 2;  // Optional: Error information  
}
1.3.4.3.5.2 List of Parameters Supported by YANG Model

The Configuration unit is across all cells/per cell config. The Cell outage is either in-service or out-of-service.
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Configuration unit</th>
<th>Cell outage</th>
<th>Description</th>
<th>YANG Model</th>
<th>xpath</th>
</tr>
</thead>
<tbody>
<tr>
<td>o-du-mac-address</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>DU side mac address, it is translated to the corresponding ‘nic’ internally</td>
<td>o-ran-uplane-config.yang o-ran-processing-element.yang ietf-interfaces.yang</td>
<td>/processing-elements/ru-elements/transport-flow/eth-flow/o-du-mac-address</td>
</tr>
<tr>
<td>ru-mac-address</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>mac address of the corresponding RU</td>
<td>o-ran-uplane-config.yang o-ran-processing-element.yang ietf-interfaces.yang</td>
<td>/processing-elements/ru-elements/transport-flow/eth-flow/ru-mac-address</td>
</tr>
<tr>
<td>vlan-id</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>vlan id</td>
<td>ietf-interfaces.yang o-ran-interfaces.yang o-ran-processing-element.yang</td>
<td>/processing-elements/ru-elements/transport-flow/eth-flow/ru-mac-address</td>
</tr>
<tr>
<td>pcp</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>vlan priority level</td>
<td>ietf-interfaces.yang o-ran-interfaces.yang o-ran-processing-element.yang</td>
<td>/processing-elements/ru-elements/transport-flow/eth-flow/ru-mac-address</td>
</tr>
<tr>
<td>ul_iq_data_fmt: bit_width</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>Indicate the bit length after compression. BFP values: 9 and 14 for , 16 for no compression Fixed point values: currently only support 16</td>
<td>o-ran-uplane-config.yang</td>
<td>/user-plane-configuration/low-level-tx-endpoints/compression/iq-bitwidth</td>
</tr>
<tr>
<td>dl_iq_data_fmt: bit_width</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>Indicate the bit length after compression. BFP values: 9 and 14 for , 16 for no compression Fixed point values: currently only support 16</td>
<td>o-ran-uplane-config.yang</td>
<td>/user-plane-configuration/low-level-rx-endpoints/compression/iq-bitwidth</td>
</tr>
<tr>
<td>dl_iq_data_fmt: comp_meth</td>
<td>per cell config</td>
<td>out-of-service</td>
<td>Indicate the dl compression method BFP values: BLOCK_FLOATING_POINT Fixed point values: NO_COMPRESSION</td>
<td>o-ran-uplane-config.yang</td>
<td>/user-plane-configuration/low-level-rx-endpoints/compression/compression-method</td>
</tr>
</tbody>
</table>

1.3. Aerial cuPHY
1.3.4.3.5.3 Reference Examples

Here is a client side reference implementation:

```
$cuBB_SDK/cuPHY-CP/cuphyoam/examples/p9_msg_client_grpc_test.cpp
```

Below are a few examples for update and retrieval of related params.

### 1.3.4.3.5.4 Update ru-mac-address, vlan-id, and pcp

**#step 1:** Edit `$cuBB_SDK∕cuPHY-CP∕cuphyoam∕examples∕mac_vlan_pcp.xml` and update `ru_mac`, `vlan_id` and `pcp` accordingly

**#step 2:** Run below cmd to do the provisioning

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd edit_config --xml_file $cuBB_SDK∕cuPHY-CP∕cuphyoam/examples/mac_vlan_pcp.xml
```

**#step 3:** Run below cmds to retrieve the config

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd get --xpath /o-ran-processing-element:processing-elements
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd get --xpath /ietf-interfaces:interfaces
```

### 1.3.4.3.5.5 Update o-du-mac-address(du nic port)

**#step 1:** Edit `$cuBB_SDK∕cuPHY-CP∕cuphyoam∕examples∕nic_du_mac.xml` and update `du_mac`, which is translated to the corresponding nic port internally

**#step 2:** Run below cmd to do the provisioning

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd edit_config --xml_file $cuBB_SDK∕cuPHY-CP∕cuphyoam/examples/nic_du_mac.xml
```

**#step 3:** Run below cmd to retrieve the config

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd get --xpath /o-ran-processing-element:processing-elements
```

### 1.3.4.3.5.6 Update DL/UL IQ data format

**#step 1:** Edit `$cuBB_SDK∕cuPHY-CP∕cuphyoam∕examples∕iq_data_fmt.xml` and update DL/UL IQ data format accordingly

- (compression-method: BLOCK_FLOATING_POINT for BFP or NO_COMPRESSION for fixed point)
- (iq-bitwidth: 9, 14, 16 for BFP or 16 for fixed point)

**#step 2:** Run below cmd to do the provisioning

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd edit_config --xml_file $cuBB_SDK∕cuPHY-CP∕cuphyoam/examples/iq_data_fmt.xml
```

**#step 3:** Run below cmd to retrieve the config

```
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd get --xpath /o-ran-uplane-conf:user-plane-configuration
```
1.3.4.3.7 Update dl and ul Exponent

#step 1: Edit $cuBB_SDK∕cuPHY-CP∕cuphyoam∕examples∕dl_ul_exponent.xml and dl and ul
→ exponent accordingly
#step 2: Run below cmd to do the provisioning
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
→ edit_config --xml_file $cuBB_SDK/cuPHY-CP/cuphyoam/examples/dl_ul_exponent.xml
#step 3: Run below cmd to retrieve the config
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
→ get --xpath /o-ran-uplane-conf:user-plane-configuration

1.3.4.3.6 Logging

1.3.4.3.6.1 Log Levels

Nvlog supports the following log levels: Fatal, Error, Console, Warning, Info, Debug, and Verbose.
A Fatal log message results in process termination. For other log levels, the process continues execution. A typical deployment sends Fatal, Error, and Console levels to stdout. Console level is for printing something that is neither a warning nor an error, but you want to print to stdout.

1.3.4.3.6.2 nvlog

This YAML container contains parameters related to nvlog configuration, see nvlog_config.yaml.

1.3.4.3.6.3 name

Used to create the shared memory log file. Shared memory handle is /dev/shm/${name}.log and temp logfile is named /tmp/${name}.log.

1.3.4.3.6.4 primary

In all processes logging to the same file, set the first starting process to be primary, set others to be secondary.

1.3.4.3.6.5 shm_log_level

Sets the log level threshold for the high performance shared memory logger. Log messages with a level at or below this threshold are sent to the shared memory logger.
Log levels: 0 - NONE, 1 - FATAL, 2 - ERROR, 3 - CONSOLE, 4 - WARNING, 5 - INFO, 6 - DEBUG, 7 - VERBOSE
Setting the log level to LOG_NONE means no logs are sent to the shared memory logger.
1.3.4.3.6.6 console_log_level

Sets the log level threshold for printing to the console. Log messages with a level at or below this threshold are printed to stdout.

1.3.4.3.6.7 max_file_size_bits

Define the rotating log file /var/log/aerial/\{name\}.log size. Size = \(2^\text{bits}\).

1.3.4.3.6.8 shm_cache_size_bits

Define the SHM cache file /dev/shm/\{name\}.log size. Size = \(2^\text{bits}\).

1.3.4.3.6.9 log_buf_size

Max log string length of one time call of the nvlog API.

1.3.4.3.6.10 max_threads

The maximum number of threads that are using nvlog all together.

1.3.4.3.6.11 save_to_file

Whether to copy and save the SHM cache log to a rotating log file under /var/log/aerial/ folder.

1.3.4.3.6.12 cpu_core_id

CPU core ID for the background log saving thread. -1 means the core is not pinned.

1.3.4.3.6.13 prefix_opts

bit5 - thread_id bit4 - sequence number bit3 - log level bit2 - module type bit1 - date bit0 - time stamp
Refer to nvlog.h for more details.

1.3.4.3.7 Metrics

The OAM Metrics API is used internally by cuPHY-CP components to report metrics (counters, gauges, and histograms). The metrics are exposed via a Prometheus Aerial exporter.
1.3.4.3.7.1 Host Metrics

Host metrics are provided via the Prometheus node exporter. The node exporter provides many thousands of metrics about the host hardware and OS, such as but not limited to:
- CPU statistics
- Disk statistics
- Filesystem statistics
- Memory statistics
- Network statistics

See https://github.com/prometheus/node_exporter and https://prometheus.io/docs/guides/node-exporter/ for detailed documentation on the node exporter.

1.3.4.3.7.2 GPU Metrics

GPU hardware metrics are provided through the GPU Operator via the Prometheus DCGM-Exporter. The DCGM-Exporter provides many thousands of metrics about the GPU and PCIe bus connection, such as but not limited to:
- GPU hardware clock rates
- GPU hardware temperatures
- GPU hardware power consumption
- GPU memory utilization
- GPU hardware errors including ECC
- PCIe throughput

See https://github.com/NVIDIA/gpu-operator for details on the GPU operator.
See https://github.com/NVIDIA/gpu-monitoring-tools for detailed documentation on the DCGM-Exporter.
An example Grafana dashboard is available at https://grafana.com/grafana/dashboards/12239.

1.3.4.3.7.3 Aerial Metric Naming Conventions

In addition to metrics available through the node exporter and DCGM-Exporter, Aerial exposes several application metrics.

Metric names are per https://prometheus.io/docs/practices/naming/ and follows the format aerial_<component>_<sub-component>_<metricdescription>_<units>.
Metric types are per https://prometheus.io/docs/concepts/metric_types/.

The component and sub-component definitions are in the table below. For each metric, the description, metric type, and metric tags are provided. Tags are a way of providing granularity to metrics without creating new metrics.
<table>
<thead>
<tr>
<th>Component</th>
<th>Sub-Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuphy</td>
<td></td>
<td>cuPHY Control Plane application</td>
</tr>
<tr>
<td></td>
<td>fapi</td>
<td>L2/L1 interface metrics</td>
</tr>
<tr>
<td></td>
<td>cplane</td>
<td>Fronthaul C-plane metrics</td>
</tr>
<tr>
<td></td>
<td>uplane</td>
<td>Fronthaul U-plane metrics</td>
</tr>
<tr>
<td></td>
<td>net</td>
<td>Generic network interface metrics</td>
</tr>
<tr>
<td>cuphy</td>
<td></td>
<td>cuPHY L1 library</td>
</tr>
<tr>
<td></td>
<td>pbch</td>
<td>Physical Broadcast Channel metrics</td>
</tr>
<tr>
<td></td>
<td>pdsch</td>
<td>Physical Downlink Shared Channel metrics</td>
</tr>
<tr>
<td></td>
<td>pdcch</td>
<td>Physical Downlink Common Channel metrics</td>
</tr>
<tr>
<td></td>
<td>pusch</td>
<td>Physical Uplink Shared Channel metrics</td>
</tr>
<tr>
<td></td>
<td>pucch</td>
<td>Physical Uplink Common Channel metrics</td>
</tr>
<tr>
<td></td>
<td>prach</td>
<td>Physical Random Access Channel metrics</td>
</tr>
</tbody>
</table>

1.3.4.3.7.4 Metrics Exporter Port

Aerial metrics are exported on port 8081. Configurable in cuphycontroller YAML file via 'aerial_metrics_backend_address'.

1.3.4.3.7.5 L2/L1 Interface Metrics

1.3.4.3.7.6 aerial_cuphykp_slots_total

Counts the total number of processed slots.
Metric type: counter
Metric tags:
  ▶ type: “UL” or “DL”
  ▶ cell: “cell number”

1.3.4.3.7.7 aerial_cuphykp_fapi_rx_packets

Counts the total number of messages L1 receives from L2.
Metric type: counter
Metric tags:
  ▶ msg_type: “type of PDU”
  ▶ cell: “cell number”
1.3.4.3.7.8 aerial_cuphycp_fapi_tx_packets
Counts the total number of messages L1 transmits to L2.
Metric type: counter
Metric tags:
   ▶ msg_type: “type of PDU”
   ▶ cell: “cell number”

1.3.4.3.7.9 Fronthaul Interface Metrics

1.3.4.3.7.10 aerial_cuphycp_cplane_tx_packets_total
Counts the total number of C-plane packets transmitted by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
   ▶ cell: “cell number”

1.3.4.3.7.11 aerial_cuphycp_cplane_tx_bytes_total
Counts the total number of C-plane bytes transmitted by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
   ▶ cell: “cell number”

1.3.4.3.7.12 aerial_cuphycp_uplane_rx_packets_total
Counts the total number of U-plane packets received by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
   ▶ cell: “cell number”

1.3.4.3.7.13 aerial_cuphycp_uplane_rx_bytes_total
Counts the total number of U-plane bytes received by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
   ▶ cell: “cell number”
1.3.4.3.7.14 aerial_cuphycp_uplane_tx_packets_total
Counts the total number of U-plane packets transmitted by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
▶ cell: "cell number"

1.3.4.3.7.15 aerial_cuphycp_uplane_tx_bytes_total
Counts the total number of U-plane bytes transmitted by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
▶ cell: "cell number"

1.3.4.3.7.16 aerial_cuphycp_uplane_lost_prbs_total
Counts the total number of PRBs expected but not received by L1 over ORAN Fronthaul interface.
Metric type: counter
Metric tags:
▶ cell: "cell number"
▶ channel: One of “prach” or “pusch”

1.3.4.3.7.17 NIC Metrics

1.3.4.3.7.18 aerial_cuphycp_net_rx_failed_packets_total
Counts the total number of erroneous packets received.
Metric type: counter
Metric tags:
▶ nic: "nic port BDF address"

1.3.4.3.7.19 aerial_cuphycp_net_rx_nombuf_packets_total
Counts the total number of receive packets dropped due to the lack of free mbufs.
Metric type: Counter
Metric tags:
▶ nic: “nic port BDF address”
1.3.4.3.7.20 aerial_cuphycp_net_rx_dropped_packets_total
Counts the total number of receive packets dropped by the NIC hardware.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.21 aerial_cuphycp_net_tx_failed_packets_total
Counts the total number of instances a packet failed to transmit.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.22 aerial_cuphycp_net_tx_accu_sched_missed_interrupt_errors_total
Counts the total number of instances accurate send scheduling missed an interrupt.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.23 aerial_cuphycp_net_tx_accu_sched_rearm_queue_errors_total
Counts the total number of accurate send scheduling rearm queue errors.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.24 aerial_cuphycp_net_tx_accu_sched_clock_queue_errors_total
Counts the total number accurate send scheduling clock queue errors.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"
1.3.4.3.7.25 aerial_cuphycp_net_tx_accu_sched_timestamp_past_errors_total

Counts the total number of accurate send scheduling timestamp in the past errors.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.26 aerial_cuphycp_net_tx_accu_sched_timestamp_future_errors_total

Counts the total number of accurate send scheduling timestamp in the future errors.
Metric type: Counter
Metric tags:
  ▶ nic: "nic port BDF address"

1.3.4.3.7.27 aerial_cuphycp_net_tx_accu_sched_clock_queue_jitter_ns

Current measurement of accurate send scheduling clock queue jitter, in units of nanoseconds.
Metric type: Gauge
Metric tags:
  ▶ nic: "nic port BDF address"

Details:
This gauge shows the TX scheduling timestamp jitter, that is, how far each individual Clock Queue (CQ) completion is from UTC time.

If you set CQ completion frequency to 2MHz (tx_pp=500), you might see the following completions:
cqe 0 at 0 ns
cqe 1 at 505 ns
cqe 2 at 996 ns
cqe 3 at 1514 ns
...

tax_pp_jitter is the time difference between two consecutive CQ completions.
1.3.4.3.7.28 aerial_cuphycp_net_tx_accu_sched_clock_queue_wander_ns

Current measurement of the divergence of Clock Queue (CQ) completions from UTC time over a longer
time period (~8s).
Metric type: Gauge
Metric tags:
▷ nic: “nic port BDF address”

1.3.4.3.7.29 Application Performance Metrics

1.3.4.3.7.30 aerial_cuphycp_slot_processing_duration_us

Counts the total number of slots with GPU processing duration in each 250us-wide histogram bin.
Metric type: Histogram
Metric tags:
▷ cell: “cell number”
▷ channel: one of “pbch”, “pdcch”, “pdsch”, “prach”, or “pusch”
▷ le: histogram less-than-or-equal-to 250us-wide histogram bins, for 250, 500, ..., 2000, +inf bins.

1.3.4.3.7.31 aerial_cuphycp_slot_pusch_processing_duration_us

Counts the total number of PUSCH slots with GPU processing duration in each 250us-wide histogram bin.
Metric type: Histogram
Metric tags:
▷ cell: “cell number”
▷ le: histogram less-than-or-equal-to 250us-wide histogram bins, range 0 to 2000us.

1.3.4.3.7.32 aerial_cuphycp_pusch_rx_tb_bytes_total

Counts the total number of transport block bytes received in the PUSCH channel.
Metric type: Counter
Metric tags:
▷ cell: “cell number”
1.3.4.3.7.33 aerial_cuphycp_pusch_rx_tb_total

Counts the total number of transport blocks received in the PUSCH channel.
Metric type: Counter
Metric tags:
   ▶ cell: "cell number"

1.3.4.3.7.34 aerial_cuphycp_pusch_rx_tb_crc_error_total

Counts the total number of transport blocks received with CRC errors in the PUSCH channel.
Metric type: Counter
Metric tags:
   ▶ cell: "cell number"

1.3.4.3.7.35 aerial_cuphycp_pusch_nrofuesperslot

Counts the total number of UEs processed in each slot per histogram bin PUSCH channel.
Metric type: Histogram
Metric tags:
   ▶ cell: "cell number"
   ▶ le: Histogram bin less-than-or-equal-to for 2, 4, ..., 24, +inf bins.

1.3.4.3.7.36 PRACH Metrics

1.3.4.3.7.37 aerial_cuphy_prach_rx_preambles_total

Counts the total number of detected preambles in PRACH channel.
Metric type: Counter
Metric tags:
   ▶ cell: "cell number"

1.3.4.3.7.38 PDSCH Metrics

1.3.4.3.7.39 aerial_cuphycp_slot_pdsch_processing_duration_us

Counts the total number of PDSCH slots with GPU processing duration in each 250us-wide histogram bin.
Metric type: Histogram
Metric tags:
   ▶ cell: "cell number"
Aerial CUDA-Accelerated RAN, Release 24-2

1.3.4.3.7.40 aerial_cuphy_pdsch_tx_tb_bytes_total
Counts the total number of transport block bytes transmitted in the PDSCH channel.
Metric type: Counter
Metric tags:
  ▶ cell: "cell number"

1.3.4.3.7.41 aerial_cuphy_pdsch_tx_tb_total
Counts the total number of transport blocks transmitted in the PDSCH channel.
Metric type: Counter
Metric tags:
  ▶ cell: "cell number"

1.3.4.3.7.42 aerial_cuphycp_pdsch_nrofuesperslot
Counts the total number of UEs processed in each slot per histogram bin PDSCH channel.
Metric type: Histogram
Metric tags:
  ▶ cell: "cell number"
  ▶ le: Histogram bin less-than-or-equal-to for 2, 4, ..., 24, +inf bins.

1.3.5. cuPHY Release Notes

1.3.5.1 cuPHY Software Manifest

Release Version: 24-2
### 1.3.5.1.1 Aerial CUDA-Accelerated RAN Software Manifest

<table>
<thead>
<tr>
<th>Description</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host OS</strong></td>
<td>▶ x86 platform: Ubuntu 22.04 with 5.15.0-1042-nvidia-lowlatency kernel</td>
</tr>
<tr>
<td></td>
<td>▶ Grace Hopper platform: Ubuntu 22.04 with 6.5.0-1019-nvidia-64k kernel</td>
</tr>
<tr>
<td><strong>GH200</strong></td>
<td>▶ CUDA Toolkit: 12.5.0</td>
</tr>
<tr>
<td></td>
<td>▶ GPU Driver (OpenRM): 555.42.02</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> If running Aerial L1 in MIG mode, downgrade the GPU driver to 550.54.15.</td>
</tr>
<tr>
<td><strong>BF3 NIC</strong></td>
<td>▶ BFB: bf-bundle-2.7.0-33_24.04_ubuuntu-22.04_prod.bfb</td>
</tr>
<tr>
<td></td>
<td>▶ NIC FW: 32.41.1000</td>
</tr>
<tr>
<td><strong>AX800 (EOL)</strong></td>
<td>▶ CUDA Toolkit: 12.5.0</td>
</tr>
<tr>
<td></td>
<td>▶ GPU Driver (OpenRM): 555.42.02</td>
</tr>
<tr>
<td></td>
<td>▶ BFB: DOCA_2.5.0_BSP_4.5.0_Ubuntu_22.04-1.23-10.prod.bfb</td>
</tr>
<tr>
<td></td>
<td>▶ NIC FW: 32.39.2048</td>
</tr>
<tr>
<td><strong>A100X</strong></td>
<td>▶ CUDA Toolkit: 12.5.0</td>
</tr>
<tr>
<td></td>
<td>▶ GPU Driver (OpenRM): 555.42.02</td>
</tr>
<tr>
<td><strong>A100 (EOL)</strong></td>
<td>▶ CUDA Toolkit: 12.5.0</td>
</tr>
<tr>
<td></td>
<td>▶ GPU Driver (OpenRM): 555.42.02</td>
</tr>
<tr>
<td><strong>CX6-DX NIC (EOL)</strong></td>
<td>NIC FW: 22.39.2048</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the CX6-DX NIC is used to run RU emulator on dual ports, downgrade the NIC FW to 22.35.1012 due to a known issue.</td>
</tr>
<tr>
<td><strong>DOCA OFED</strong></td>
<td>24.04-0.6.6</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> DOCA OFED is only required by Grace Hopper platform. It is not required for x86 platform.</td>
</tr>
<tr>
<td><strong>NVIDIA-peermem</strong></td>
<td><strong>Note:</strong> Aerial has been using kernel DMA-buf instead of nvidia-peermem since 23-4 release. Nvidia-peermem is not required anymore.</td>
</tr>
<tr>
<td><strong>GDRCopy</strong></td>
<td>2.4.1</td>
</tr>
<tr>
<td><strong>DPDK</strong></td>
<td>22.11 (Included in Mellanox DOCA)</td>
</tr>
<tr>
<td><strong>DOCA</strong></td>
<td>2.7</td>
</tr>
<tr>
<td><strong>cuPHY</strong></td>
<td>1.14</td>
</tr>
<tr>
<td><strong>SCF</strong></td>
<td>222.10.02 (partial upgrade to 222.10.04)</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>▶ Supermicro Grace Hopper MGX ARS-111GL-NHR (Config 2)</td>
</tr>
<tr>
<td></td>
<td>▶ Dell PowerEdge R750 with duel Intel(R) Xeon(R) Gold 6336Y CPU @ 2.40GHz</td>
</tr>
<tr>
<td></td>
<td>▶ Gigabyte(E251-U70) with Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz (EOL)</td>
</tr>
</tbody>
</table>
Note:
- Aerial support of AX800, A100, CX6-DX has reached End of Life (EOL) on June 30, 2024.
- Aerial has been using DMA-buf, inbox driver and OpenRM driver since 23-4 release. So MOFED and nvidia-peermem are not needed anymore. On the x86 platform, the 5.15 kernel with DMA-buf and inbox driver are used. On the Grace Hopper platform, the 6.2 kernel with DMA-buf and DOCA OFED are used.

### 1.3.5.1.2 Kubernetes Software Manifest

<table>
<thead>
<tr>
<th>Description</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host OS</td>
<td>Grace Hopper platform: Ubuntu 22.04 with 6.5.0-1019-nvidia-64k kernel</td>
</tr>
<tr>
<td>Container OS</td>
<td>Ubuntu 22.04</td>
</tr>
<tr>
<td>Containerd</td>
<td>1.5.8</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>1.23</td>
</tr>
<tr>
<td>Helm</td>
<td>3.8</td>
</tr>
<tr>
<td>BF3 NIC FW</td>
<td>32.41.1000</td>
</tr>
<tr>
<td>GPU Operator</td>
<td>24.3.0</td>
</tr>
<tr>
<td>CUDA Toolkit</td>
<td>12.5.0</td>
</tr>
<tr>
<td>NVIDIA GPU Driver</td>
<td>550.54.15</td>
</tr>
</tbody>
</table>

### 1.3.5.2 Supported Features and Configurations

This release of the Aerial cuBB supports the following configurations and features. These features are verified with test vectors in a simulated environment using TestMAC and RU emulator.

#### 1.3.5.2.1 PUSCH

- Process multiple cells (homogeneous and heterogeneous configurations) using CUDA streams or CUDA graphs
- Frequency multiplexing
- BW: 100 MHz
- Antenna ports: 4
- SU-MIMO layers: up to 4
- MU-MIMO layers: up to 4
- Modulation and coding rates: MCS 0 – MCS 27
- Optimized LDPC decoder
- UCI on PUSCH (HARQ up to 11 bits + CSI part 1 + CSI part 2 up to 11 bits)
Time-interpolated channel estimation and equalization
SINR reporting to L2
MMSE-IRC receiver
Early HARQ in UCI.indication

1.3.5.2.2 PUCCH

Format 0 + DTX detection
Format 1 + DTX detection
Format 2 (unsegmented payload) + DTX detection
Format 3 (unsegmented payload) + DTX detection
SINR / confidence level reporting to L2

1.3.5.2.3 PRACH

Format 0
Format B4 (multiple per slot in FDM)
Interference level reporting

1.3.5.2.4 PDSCH

Process multiple cells (homogeneous and heterogeneous configurations) using CUDA streams or CUDA graphs.
Frequency multiplexing
BW: 100 MHz
Antenna ports: 4
SU-MIMO layers: up to 4
MU-MIMO layers: up to 4
Modulation and coding rates: MCS 0 – MCS 27
Supports Cat-A O-RAN split and Cat-B O-RAN split. For Cat-A O-RAN split, PDSCH is implemented up to modulation and precoding (identity matrix precoder) For Cat-B O-RAN split, PDSCH is implemented up to the rate matching block.
Precoding (4 layers)
1.3.5.2.5 PDCCH

- Process multiple cells (homogeneous and heterogeneous configurations) using CUDA streams or CUDA graphs.
- Interleaved and non-interleaved mode
- Aggregation level (AL) 1, 2, 4, 8, 16
- 1, 2, 3 symbol CORESET
- Precoding (1 layer)

1.3.5.2.6 SS Block

- Process multiple cells (homogeneous and heterogeneous configurations) using CUDA streams or CUDA graphs.
- PSS, SSS generation
- DMRS and PBCH generation and time-frequency mapping
- Precoding (1 layer)

1.3.5.2.7 CSI-RS

- Process multiple cells (homogeneous and heterogeneous configurations) using CUDA streams or CUDA graphs.
- NZP-CSI-RS
- ZP-CSI-RS
- Precoding (1 layer)

1.3.5.2.8 SRS

- Support SRS reporting for normalized 8 bit I/Q samples for upto 32T32R BB Antenna ports.
- Support SRS reporting according to 5G FAPI 222.10.04 for beamManagemnt, codebook and non-codebook SRS usage.
- Support SRS reporting according to 5G FAPI 222.10.02 for SINR reporting.

1.3.5.2.9 MIMO Features

- Support 32 Transmit and Receive antenna ports
- Support SRS-based channel estimation, buffering and FAPI-compliant reporting to L2
- Support PUSCH and PDSCH beamforming weight (BFW) calculation from SRS channel estimates (regularized zero-forcing)
- Support up to 4 layers multi-user MIMO PUSCH
- Support up to 8 layers multi-user MIMO PDSCH
1.3.5.2.10 LDPC Decoder

- Standalone LDPC decoder

1.3.5.2.11 SHM Logger

- Support for C++ std::format style logging like std::format("{} {}!", "Hello", "world ", "something");
- Support for C (printf) style formatted strings.

1.3.5.3 Multicell Capacity

CPU core usage for multicell benchmark:

On Grace Hopper:
- 1 isolated physical Grace core for core-locked PTP applications (phc2sys+ptp4l)
- 10 additional isolated Grace cores for the other core-locked cuphycontroller threads

On x86 based targets:

Without hyperthreading using "1+6+fractional" x86 cores. The shorthand "1+6+fractional" x86 cores is defined as follows:
- 1 isolated physical x86 core for core-locked PTP applications (phc2sys+ptp4l) and the core-locked cuphycontroller L2A H2D prepone thread
- 6 additional isolated physical x86 cores for the other core-locked cuphycontroller threads
- A fraction of a shared floating x86 core for non-core-locked cuphycontroller threads

Additionally, the tested L2 timeline is as follows:
- FAPI SLOT.indication for Slot N is sent from L1 to L2 at the wall-clock time for Slot N-3 (i.e. 3 slot advance).
- For the DDDSUUDDDD TDD pattern with 0-based slot numbering, L2 has up-to-500us, for slot%10 in {2,3,4,5,6}, from SLOT.indication to deliver all FAPI PDUs for Slot N.
- For the DDDSUUDDDD TDD pattern with 0-based slot numbering, L2 has up-to-250us, for slot%10 in {0,1,7,8,9}, from SLOT.indication to deliver all FAPI PDUs for Slot N.

As of 24-2:

Supports 500us L2 processing budget and 7 beam peak and average patterns as defined below using 100MHz:

On Grace Hopper:
- BFP9: 20 4T4R Peak cells / 20 4T4R average cells

while respecting the following configuration for 7 beam traffic patterns:
Table 6: TDD 4T4R - 80 Slot Traffic Models

<table>
<thead>
<tr>
<th>7-beam config</th>
<th>Configuration (4 UL streams RU-&gt;DU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Peak</td>
</tr>
<tr>
<td>Compression</td>
<td>BFP9 and BFP14</td>
</tr>
<tr>
<td>Max PxSCH PRB</td>
<td>270</td>
</tr>
<tr>
<td>DL Throughput/cell</td>
<td>1469.14 Mbps</td>
</tr>
<tr>
<td>UL Throughput/cell</td>
<td>212.64 Mbps</td>
</tr>
<tr>
<td>Peak DL Fronthaul Bandwidth / cell</td>
<td>11.06 Gbps BFP14</td>
</tr>
<tr>
<td></td>
<td>7.14 Gbps BFP9</td>
</tr>
<tr>
<td>Peak UL Fronthaul Bandwidth / cell</td>
<td>11.88 Gbps BFP14</td>
</tr>
<tr>
<td></td>
<td>8.03 Gbps BFP9</td>
</tr>
<tr>
<td>SSB slots</td>
<td>Frame 0 &amp; 2: 0,1,2,3</td>
</tr>
<tr>
<td>#SSB per slot</td>
<td>Frame 0 &amp; 2: 2,2,2,2,2</td>
</tr>
<tr>
<td>TRS slots</td>
<td>Frame 0-3: 6,7,8,9,10,11</td>
</tr>
<tr>
<td></td>
<td>Frame 0 &amp; 2: 16,17</td>
</tr>
<tr>
<td>TRS Symbols</td>
<td>Even cells: 6,10</td>
</tr>
<tr>
<td></td>
<td>Odd cells: 5,9</td>
</tr>
<tr>
<td>CSI-RS slots</td>
<td>Frame 0: 8,10,16</td>
</tr>
<tr>
<td></td>
<td>Frame 1: 6,8,10</td>
</tr>
<tr>
<td></td>
<td>Frame 2: 6</td>
</tr>
<tr>
<td>CSI-RS Symbols</td>
<td>Even cells: 12</td>
</tr>
<tr>
<td></td>
<td>Odd cells: 13</td>
</tr>
<tr>
<td>PDSCH #DCI</td>
<td>12 (6 DL + 6 UL per slot)</td>
</tr>
<tr>
<td>UE/TTI/Cell</td>
<td>6 per DL slot, 6 per UL slot</td>
</tr>
<tr>
<td>UCI on PUSCH HARQ+CSIP1+CSIP2 (bits)</td>
<td>4+37+5</td>
</tr>
<tr>
<td>PUCCH format</td>
<td>1</td>
</tr>
<tr>
<td>PUCCH payload (bits)</td>
<td>18</td>
</tr>
<tr>
<td>PRACH format</td>
<td>B4</td>
</tr>
<tr>
<td>PRACH slots</td>
<td>Frame 0-3: 5, 15</td>
</tr>
<tr>
<td>PRACH occasions</td>
<td>Slot 5: 4, Slot 15: 3</td>
</tr>
</tbody>
</table>

Notes:

- Stated performance achievement and CPU core count usage is for L1 workload only (additional non-L1 workloads in E2E setting may have an impact on the achieved performance and/or CPU core count usage)
- Performance achievement is measured by running L1 in steady-state traffic mode (e.g. impact of workloads such as cell reconfiguration on other cells is not captured)
1.3.5.4 Supported Test Vector Configurations

This release of Aerial cuBB currently supports the following test-vector configurations.

1.3.5.4.1 PUSCH

<table>
<thead>
<tr>
<th>TC start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>7201</td>
<td>7201</td>
<td>base</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7202</td>
<td>7203</td>
<td>mcsTable</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7204</td>
<td>7204</td>
<td>mcs</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7205</td>
<td>7207</td>
<td>num of layers</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7208</td>
<td>7208</td>
<td>rb0, Nrb</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7209</td>
<td>7210</td>
<td>sym0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7211</td>
<td>7211</td>
<td>dmsr0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7212</td>
<td>7213</td>
<td>Nsym</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7214</td>
<td>7214</td>
<td>SCID</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7215</td>
<td>7215</td>
<td>BWPO, nBWP</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7216</td>
<td>7216</td>
<td>RNTI</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7217</td>
<td>7219</td>
<td>addPos</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7220</td>
<td>7220</td>
<td>dataScId</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7221</td>
<td>7222</td>
<td>maxLen</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7223</td>
<td>7223</td>
<td>dmrsScId</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7224</td>
<td>7224</td>
<td>nCdm</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7225</td>
<td>7225</td>
<td>port0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7227</td>
<td>7227</td>
<td>nAnt=2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7228</td>
<td>7228</td>
<td>nAnt=16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7229</td>
<td>7229</td>
<td>slotIdx</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7230</td>
<td>7232</td>
<td>rvIdx</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7233</td>
<td>7235</td>
<td>FDM</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7236</td>
<td>7241</td>
<td>CDM</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7242</td>
<td>7244</td>
<td>rvIdx&gt;0/BGN=1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7245</td>
<td>7245</td>
<td>ulGridSize=106</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7246</td>
<td>7247</td>
<td>dmrs_par per Ueg</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7248</td>
<td>7250</td>
<td>additional FDM</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7251</td>
<td>7257</td>
<td>precoding</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

continues
<table>
<thead>
<tr>
<th>TC start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>7258</td>
<td>7260</td>
<td>mapping type B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7261</td>
<td>7272</td>
<td>Flexible DMRS ports</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7273</td>
<td>7273</td>
<td>MCS &gt; 28</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7274</td>
<td>7279</td>
<td>additional nCDM=1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7280</td>
<td>7283</td>
<td>Flexible SLIV</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7301</td>
<td>7320</td>
<td>multi-params</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>7321</td>
<td>7323</td>
<td>LBRM</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7324</td>
<td>7326</td>
<td>HARQ-rx</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7327</td>
<td>7330</td>
<td>8/16 UEs</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7331</td>
<td>7338</td>
<td>multiple layers</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7340</td>
<td>7340</td>
<td>Multi-layers with nAnt=16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7401</td>
<td>7403</td>
<td>CFO</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7404</td>
<td>7406</td>
<td>TO</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7407</td>
<td>7407</td>
<td>RSSI</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7408</td>
<td>7408</td>
<td>CFO w/ SDM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7409</td>
<td>7409</td>
<td>TO w/ SDM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7410</td>
<td>7411</td>
<td>CEE-TDI</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7412</td>
<td>7413</td>
<td>rx power</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7414</td>
<td>7414</td>
<td>TDI maxLen = 2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7415</td>
<td>7417</td>
<td>small/big/zero rx</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7418</td>
<td>7419</td>
<td>additional TDI</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7420</td>
<td>7426</td>
<td>IRC=0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7427</td>
<td>7432</td>
<td>SINR meas</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7501</td>
<td>7516</td>
<td>UCI on PUSCH (w/o data)</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7517</td>
<td>7530</td>
<td>UCI on PUSCH (w/ data)</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>7531</td>
<td>7531</td>
<td>UciOnPusch DTX</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7532</td>
<td>7532</td>
<td>UciOnPusch CRC fail</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7533</td>
<td>7534</td>
<td>UciOnPusch addPos</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7551</td>
<td>7570</td>
<td>UciOnPusch (multi-params)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>7571</td>
<td>7575</td>
<td>UCI w/ and w/o data</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7601</td>
<td>7613</td>
<td>FR1 BW mu = 1</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>7614</td>
<td>7621</td>
<td>FR1 BW mu = 0</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

continues
### 1.3.5.4.2 PUCCH

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>6001</td>
<td>6003</td>
<td>bases for format 0</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6004</td>
<td>6010</td>
<td>vary single parameter for format 0</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>6011</td>
<td>6040</td>
<td>vary multiple parameters for format 0</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>6041</td>
<td>6046</td>
<td>vary slotIdx (single-UCI) for format 0</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6047</td>
<td>6056</td>
<td>multi-UCI tests for format 0</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6057</td>
<td>6061</td>
<td>vary slotIdx (multi-UCI) for format 0</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6101</td>
<td>6103</td>
<td>bases for format 1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6104</td>
<td>6116</td>
<td>vary single parameter for format 1</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>6117</td>
<td>6146</td>
<td>vary multiple parameters for format 1</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>6147</td>
<td>6155</td>
<td>vary slotIdx (single-UCI) for format 1</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>6156</td>
<td>6173</td>
<td>multi-UCI tests for format 1</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>6175</td>
<td>6192</td>
<td>TA estimation for format 1</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>6193</td>
<td>6194</td>
<td>192 UCI groups for format 1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6201</td>
<td>6203</td>
<td>bases for format 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6204</td>
<td>6219</td>
<td>test Nf for format 2</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>6220</td>
<td>6235</td>
<td>test Nt and freq hopping for format 2</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>6236</td>
<td>6236</td>
<td>11 info bits and 2 PRBS for format 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6239</td>
<td>6245</td>
<td>different payload sizes for format 2</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>6301</td>
<td>6310</td>
<td>bases for format 3</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6311</td>
<td>6313</td>
<td>multi-UCI tests for format 3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6314</td>
<td>6324</td>
<td>tests with freqHop enabled for format 3</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>6325</td>
<td>6335</td>
<td>tests with freqHop disabled for format 3</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>6336</td>
<td>6346</td>
<td>tests with add'l DMRS position, freqHop enabled for format 3</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>6347</td>
<td>6357</td>
<td>tests with add'l DMRS position, freqHop disabled for format 3</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

continues on next page

1.3. Aerial cuPHY 271
<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>6358</td>
<td>6364</td>
<td>different payload sizes for format 3</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>6365</td>
<td>6373</td>
<td>24-UCI tests for format 3</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>6501</td>
<td>6513</td>
<td>sweep different bandwidth for format 0, mu = 1</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>6514</td>
<td>6526</td>
<td>sweep different bandwidth for format 1, mu = 1</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>6527</td>
<td>6539</td>
<td>sweep different bandwidth for format 2, mu = 1</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>6540</td>
<td>6552</td>
<td>sweep different bandwidth for format 3, mu = 1</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>6553</td>
<td>6560</td>
<td>sweep different bandwidth for format 0, mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>6561</td>
<td>6568</td>
<td>sweep different bandwidth for format 1, mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>6569</td>
<td>6576</td>
<td>sweep different bandwidth for format 2, mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>6577</td>
<td>6584</td>
<td>sweep different bandwidth for format 3, mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>6585</td>
<td>6586</td>
<td>rx power for format 0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6587</td>
<td>6588</td>
<td>rx power for format 1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6589</td>
<td>6590</td>
<td>rx power for format 2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6591</td>
<td>6592</td>
<td>rx power for format 3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6593</td>
<td>6595</td>
<td>very small/very big/forcRxZero rx power for format 0</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6596</td>
<td>6598</td>
<td>very small/very big/forcRxZero rx power for format 1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6599</td>
<td>6601</td>
<td>very small/very big/forcRxZero rx power for format 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6602</td>
<td>6605</td>
<td>very small/very big/forcRxZero rx power for format 3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6801</td>
<td>6802</td>
<td>perf TV F08</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6803</td>
<td>6804</td>
<td>perf TV F14</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
### 1.3.5.4.3 PRACH

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>5001</td>
<td>base</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5002</td>
<td>5002</td>
<td>format 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5003</td>
<td>5003</td>
<td>rootIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5004</td>
<td>5004</td>
<td>zoneIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5005</td>
<td>5005</td>
<td>prmbIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5006</td>
<td>5006</td>
<td>Nant</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5007</td>
<td>5007</td>
<td>N_nc</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5008</td>
<td>5008</td>
<td>delay</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5009</td>
<td>5009</td>
<td>SNR</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5010</td>
<td>5010</td>
<td>CFO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5011</td>
<td>5011</td>
<td>2-UE</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5012</td>
<td>5012</td>
<td>4-UE</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5013</td>
<td>5013</td>
<td>4FDM/16UE</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5014</td>
<td>5018</td>
<td>rx power</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5101</td>
<td>5101</td>
<td>FDD,mu=0,B4,nAnt=2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5102</td>
<td>5102</td>
<td>FDD,mu=1,B4,nAnt=4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5103</td>
<td>5103</td>
<td>TDD,mu=0,B4,nAnt=8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5104</td>
<td>5104</td>
<td>TDD,mu=1,B4,nAnt=16</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5105</td>
<td>5105</td>
<td>FDD,mu=0,F0,nAnt=16</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5106</td>
<td>5106</td>
<td>FDD,mu=1,F0,nAnt=8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5107</td>
<td>5107</td>
<td>TDD,mu=0,F0,nAnt=4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5108</td>
<td>5108</td>
<td>TDD,mu=1,F0,nAnt=2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5201</td>
<td>5213</td>
<td>FR1 BW mu = 1</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>5214</td>
<td>5221</td>
<td>FR1 BW mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>5801</td>
<td>5802</td>
<td>perf TV F08</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5803</td>
<td>5804</td>
<td>perf TV F14</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5901</td>
<td>5901</td>
<td>demo_msg1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5911</td>
<td>5914</td>
<td>comformance TC</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
### 1.3.5.4.4 PDSCH

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>3201</td>
<td>3201</td>
<td>base</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3202</td>
<td>3203</td>
<td>mcsTable</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3204</td>
<td>3204</td>
<td>mcs</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3205</td>
<td>3207</td>
<td>num of layers</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3208</td>
<td>3208</td>
<td>rb0, Nrb</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3209</td>
<td>3210</td>
<td>sym0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3211</td>
<td>3211</td>
<td>dmrs0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3212</td>
<td>3213</td>
<td>Nsym</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3214</td>
<td>3214</td>
<td>SCID</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3215</td>
<td>3215</td>
<td>BWPO, nBWP</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3216</td>
<td>3216</td>
<td>RNTI</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3217</td>
<td>3219</td>
<td>addPos</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3220</td>
<td>3220</td>
<td>dataScld</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3221</td>
<td>3222</td>
<td>maxLen</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3223</td>
<td>3223</td>
<td>dmrsScld</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3224</td>
<td>3224</td>
<td>nCdm</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3225</td>
<td>3225</td>
<td>port0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3226</td>
<td>3228</td>
<td>nAnt</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3229</td>
<td>3229</td>
<td>slotIdx</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3230</td>
<td>3232</td>
<td>rvldx</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3233</td>
<td>3235</td>
<td>FDM</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3236</td>
<td>3241</td>
<td>SDM/SCID</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3242</td>
<td>3244</td>
<td>rvldx&gt;0/BGN=1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3245</td>
<td>3245</td>
<td>dlGridSize=106</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3246</td>
<td>3247</td>
<td>dmrs_par per Ueg</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3248</td>
<td>3254</td>
<td>precoding</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3255</td>
<td>3257</td>
<td>mapping type B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3258</td>
<td>3260</td>
<td>mixed precoding</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3261</td>
<td>3261</td>
<td>refPoint</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3262</td>
<td>3262</td>
<td>TxPower</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3263</td>
<td>3263</td>
<td>modComp</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 9 – continued from previous page

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>3264</td>
<td>3264</td>
<td>precoding (mixed nPorts)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3265</td>
<td>3265</td>
<td>TxPower with 2 UEs</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3266</td>
<td>3267</td>
<td>different rv</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3268</td>
<td>3269</td>
<td>multi-layer</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3271</td>
<td>3276</td>
<td>nCDM = 1</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>3321</td>
<td>3322</td>
<td>LBRM</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3323</td>
<td>3333</td>
<td>RE map from CSI-RS</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>3334</td>
<td>3336</td>
<td>8/16 UEs (SU-MIMO)</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3337</td>
<td>3337</td>
<td>16 UEs (MU-MIMO)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3401</td>
<td>3413</td>
<td>FR1 BW mu = 1</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>3414</td>
<td>3421</td>
<td>FR1 BW mu = 0</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3901</td>
<td>3901</td>
<td>demo_coreset0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3902</td>
<td>3902</td>
<td>demo_msg2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3903</td>
<td>3903</td>
<td>demo_msg4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3904</td>
<td>3904</td>
<td>demo_traffic_dl</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3001</td>
<td>3015</td>
<td>multi-params</td>
<td>15</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3016</td>
<td>3154</td>
<td>sweep Zc/mcs (3016,3017,3024,3025,3032,3039,3045,3057 are skipped)</td>
<td>131</td>
<td>131</td>
<td></td>
</tr>
</tbody>
</table>

1.3.5.4.5 PDCCH

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>2001</td>
<td>base</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2002</td>
<td>2002</td>
<td>slotIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2003</td>
<td>2003</td>
<td>nBWP</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2004</td>
<td>2004</td>
<td>BPW0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2005</td>
<td>2005</td>
<td>sym0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2006</td>
<td>2007</td>
<td>Nsym</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2008</td>
<td>2009</td>
<td>crstIdx</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2010</td>
<td>2010</td>
<td>intl</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2011</td>
<td>2012</td>
<td>nBndl</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2013</td>
<td>2014</td>
<td>nintl</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2015</td>
<td>2015</td>
<td>nShift</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>2016</td>
<td>isCSS</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2017</td>
<td>2017</td>
<td>rnti</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2018</td>
<td>2018</td>
<td>scrblld</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2019</td>
<td>2019</td>
<td>scrbRnti</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2020</td>
<td>2022</td>
<td>aggrL</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2023</td>
<td>2023</td>
<td>dBQam</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2024</td>
<td>2024</td>
<td>dBDMrs</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2025</td>
<td>2025</td>
<td>Npayload</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2026</td>
<td>2027</td>
<td>crstMap</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2028</td>
<td>2028</td>
<td>nDCI</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2029</td>
<td>2029</td>
<td>Npayload</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2030</td>
<td>2030</td>
<td>aggrL</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2031</td>
<td>2031</td>
<td>precoding</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2032</td>
<td>2032</td>
<td>modComp</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2033</td>
<td>2033</td>
<td>multi-PDCCH</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2101</td>
<td>2112</td>
<td>multi-params</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>2201</td>
<td>2213</td>
<td>FR1 BW mu = 1</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>2214</td>
<td>2221</td>
<td>FR1 BW mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>2801</td>
<td>2802</td>
<td>perf TV F14</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2803</td>
<td>2804</td>
<td>perf TV F08</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2805</td>
<td>2806</td>
<td>perf TV F09</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2901</td>
<td>2901</td>
<td>demo_msg2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2902</td>
<td>2902</td>
<td>demo_msg4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2903</td>
<td>2903</td>
<td>demo_coreset0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2904</td>
<td>2904</td>
<td>demo_traffic_dl</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2905</td>
<td>2905</td>
<td>demo_msg5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### 1.3.5.4.6 SS Block

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1001</td>
<td>base</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>1002</td>
<td>μ = 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1003</td>
<td>1003</td>
<td>N_CELL_ID</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1004</td>
<td>1004</td>
<td>n_hf = 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1005</td>
<td>1005</td>
<td>L_max = 4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1006</td>
<td>1006</td>
<td>k_SSB</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1007</td>
<td>1007</td>
<td>offsetPointA</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1008</td>
<td>1008</td>
<td>SFN</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1009</td>
<td>1009</td>
<td>blockIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1010</td>
<td>1010</td>
<td>precoding</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1011</td>
<td>1011</td>
<td>betaPss</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1101</td>
<td>1101</td>
<td>μ=0, 1SSB</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1102</td>
<td>1102</td>
<td>μ=1, 1SSB</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1103</td>
<td>1103</td>
<td>μ=1, 2SSB</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1104</td>
<td>1104</td>
<td>μ=1, 2SSB</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1202</td>
<td>1213</td>
<td>FR1 BW, μ = 1</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>1214</td>
<td>1221</td>
<td>FR1 BW, μ = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>1801</td>
<td>1801</td>
<td>Perf TV</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1901</td>
<td>1901</td>
<td>demo_ssb</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1902</td>
<td>1902</td>
<td>for CP pipeline</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### 1.3.5.4.7 CSI-RS

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>4001</td>
<td>4004</td>
<td>nPorts = 1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4005</td>
<td>4007</td>
<td>nPorts = 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4008</td>
<td>4009</td>
<td>nPorts = 4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4010</td>
<td>4012</td>
<td>nPorts = 8</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4013</td>
<td>4038</td>
<td>nPorts &gt; 8, row &gt; 8</td>
<td>26</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>4039</td>
<td>4039</td>
<td>RB0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4040</td>
<td>4040</td>
<td>nRB</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4041</td>
<td>4041</td>
<td>sym0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4042</td>
<td>4042</td>
<td>sym1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4043</td>
<td>4043</td>
<td>nID</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4044</td>
<td>4044</td>
<td>power control</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4045</td>
<td>4050</td>
<td>freqDomainAllocation</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4051</td>
<td>4051</td>
<td>idxSlot</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4052</td>
<td>4054</td>
<td>batching</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4055</td>
<td>4055</td>
<td>small gird size</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4056</td>
<td>4056</td>
<td>TRS</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4057</td>
<td>4057</td>
<td>precoding</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4058</td>
<td>4058</td>
<td>modComp</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4059</td>
<td>4060</td>
<td>16/32 CSIRS PDUs</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4101</td>
<td>4103</td>
<td>multiple parameters</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4201</td>
<td>4213</td>
<td>FR1 BW mu = 1</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>4214</td>
<td>4221</td>
<td>FR1 BW mu = 0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>4801</td>
<td>4801</td>
<td>perf TV F08</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4802</td>
<td>4802</td>
<td>perf TV F09</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4803</td>
<td>4803</td>
<td>perf TV F14</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### 1.3.5.4.8 SRS

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>8001</td>
<td>8001</td>
<td>base</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8002</td>
<td>8002</td>
<td>rnti</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8003</td>
<td>8003</td>
<td>Nap=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8004</td>
<td>8004</td>
<td>Nap=4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8005</td>
<td>8005</td>
<td>nSym=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8006</td>
<td>8006</td>
<td>nSym=4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8007</td>
<td>8007</td>
<td>Nrep=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8008</td>
<td>8008</td>
<td>Nrep=4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8009</td>
<td>8009</td>
<td>sym0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8010</td>
<td>8010</td>
<td>cfgIdx</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8011</td>
<td>8011</td>
<td>seqId</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8012</td>
<td>8012</td>
<td>bwIdx=1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8013</td>
<td>8013</td>
<td>bwIdx=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8014</td>
<td>8014</td>
<td>bwIdx=3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8015</td>
<td>8015</td>
<td>cmbSize</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8016</td>
<td>8016</td>
<td>cmbOffset</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8017</td>
<td>8017</td>
<td>cyclic shift</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8018</td>
<td>8018</td>
<td>freqPosition</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8019</td>
<td>8019</td>
<td>freqShift</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8020</td>
<td>8020</td>
<td>freqHopping=1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8021</td>
<td>8021</td>
<td>freqHopping=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8022</td>
<td>8022</td>
<td>freqHopping=3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8023</td>
<td>8023</td>
<td>grpSeqHopping=1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8024</td>
<td>8024</td>
<td>grpSeqHopping=2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8025</td>
<td>8025</td>
<td>rsrcType,Tsrs,Toffset</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8026</td>
<td>8026</td>
<td>idxSlot</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8027</td>
<td>8033</td>
<td>multi-SRS</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8034</td>
<td>8034</td>
<td>rsrcType,Tsrs,Toffset</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8035</td>
<td>8035</td>
<td>16 users wideband</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8051</td>
<td>8057</td>
<td>multiple parameters</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8101</td>
<td>8164</td>
<td>sweep cfgIdx</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

*continues on next page*
### Table 11 – continued from previous page

<table>
<thead>
<tr>
<th>TC Start</th>
<th>TC End</th>
<th>Description</th>
<th>TV Generated</th>
<th>cuPHY Pass</th>
<th>cuBB Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>8201</td>
<td>8213</td>
<td>FR1 BW mu=1</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>8214</td>
<td>8221</td>
<td>FR1 BW mu=0</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>8222</td>
<td>8226</td>
<td>rx power</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8227</td>
<td>8227</td>
<td>additional BW</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8301</td>
<td>8302</td>
<td>SRS integration</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8401</td>
<td>8415</td>
<td>32 nAnt</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>8420</td>
<td>8421</td>
<td>32 nAnt</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8501</td>
<td>8524</td>
<td>64 nAnt</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>8801</td>
<td>8801</td>
<td>F09 perf TV</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8802</td>
<td>8802</td>
<td>20M perf TV</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90001</td>
<td>90007</td>
<td>single channel</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>90011</td>
<td>90012</td>
<td>dlmix/ulmix</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>90013</td>
<td>90015</td>
<td>s-slot</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>90016</td>
<td>90018</td>
<td>multi-cell base case</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>90019</td>
<td>90019</td>
<td>prcd+noPrcd</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90020</td>
<td>90020</td>
<td>BFP14+BFP9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90021</td>
<td>90022</td>
<td>HARQ</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>90023</td>
<td>90023</td>
<td>empty slot</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90032</td>
<td>90037</td>
<td>multi-slot combo TC</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>90041</td>
<td>90046</td>
<td>SRS + UL + DL</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>90051</td>
<td>90056</td>
<td>mixed cells</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>90057</td>
<td>90058</td>
<td>adaptive re-tx</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>90060</td>
<td>90060</td>
<td>SRS even/odd frames</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90501</td>
<td>90505</td>
<td>bug TCs</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>90601</td>
<td>90603</td>
<td>multi-channel TCs</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

### 1.3.5.4.9 mSlot_mCell
The `ldpc_perf_collect.py` Python script from the cuPHY repository can be used to perform error rate tests for the cuPHY LDPC decoder. There are test input files defined for $Z = [64, 128, 256, 384]$, $BG = [1, 2]$. The tests check whether the block error rate (BLER, also sometimes referred to as Frame Error Rate or FER) is less than 0.1.

From the build directory, the following commands run the tests:

```bash
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG1_Z64\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG1_Z128\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG1_Z256\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG1_Z384\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG2_Z64\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG2_Z128\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG2_Z256\_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i
./util/ldpc/test/ldpc_decode_BG2_Z384\_BLER0.1.txt -f -w 800 -P
```

Each test input file contains multiple tests for different code rates, as specified by the number of parity nodes. The format of the input files has the following form:

```
# BG Z num_parity num_iter SNR max_BER max_BLER
#--------------------------------------------------------------
1 384 4 10 6.87 1.000000e+00 4.833980e-04 1.000000e-01 8.
1 384 5 10 6.15 1.000000e+00 4.833980e-04 1.000000e-01 8.
1 384 6 10 5.64 1.000000e+00 4.833980e-04 1.000000e-01 8.
1 384 7 10 5.17 1.000000e+00 4.833980e-04 1.000000e-01 8.
1 384 8 10 4.79 1.000000e+00 4.833980e-04 1.000000e-01 8.
```

After running each of the test cases, the `ldpc_perf_collect.py` script displays an output table:

```
+-------------------------------------------------------------------------------------
<table>
<thead>
<tr>
<th># BG</th>
<th>Z</th>
<th>num_parity</th>
<th>num_iter</th>
<th>SNR</th>
<th>max_BER</th>
<th>max_BLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>384</td>
<td>4</td>
<td>10</td>
<td>6.87</td>
<td>1.000000e+00</td>
<td>4.833980e-04</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>5</td>
<td>10</td>
<td>6.15</td>
<td>1.000000e+00</td>
<td>4.833980e-04</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>6</td>
<td>10</td>
<td>5.64</td>
<td>1.000000e+00</td>
<td>4.833980e-04</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>7</td>
<td>10</td>
<td>5.17</td>
<td>1.000000e+00</td>
<td>4.833980e-04</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>8</td>
<td>10</td>
<td>4.79</td>
<td>1.000000e+00</td>
<td>4.833980e-04</td>
</tr>
</tbody>
</table>
```

(continues on next page)
<table>
<thead>
<tr>
<th>1</th>
<th>384</th>
<th>5</th>
<th>10</th>
<th>6.150 1.000000e+00</th>
<th>1.481120e-04</th>
<th>1.000000e-01</th>
<th>7.250000e-02</th>
<th>PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>384</td>
<td>6</td>
<td>10</td>
<td>5.640 1.000000e+00</td>
<td>5.652230e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>7</td>
<td>10</td>
<td>5.170 1.000000e+00</td>
<td>7.886480e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>8</td>
<td>10</td>
<td>4.790 1.000000e+00</td>
<td>1.673470e-04</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>9</td>
<td>10</td>
<td>4.480 1.000000e+00</td>
<td>1.185190e-04</td>
<td>1.000000e-01</td>
<td>7.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>10</td>
<td>10</td>
<td>4.200 1.000000e+00</td>
<td>8.552320e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>11</td>
<td>10</td>
<td>3.920 1.000000e+00</td>
<td>5.385890e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>12</td>
<td>10</td>
<td>3.660 1.000000e+00</td>
<td>1.234020e-04</td>
<td>1.000000e-01</td>
<td>9.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>13</td>
<td>10</td>
<td>3.450 1.000000e+00</td>
<td>7.013490e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>14</td>
<td>10</td>
<td>3.220 1.000000e+00</td>
<td>7.620150e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>15</td>
<td>10</td>
<td>3.020 1.000000e+00</td>
<td>5.800190e-05</td>
<td>1.000000e-01</td>
<td>7.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>16</td>
<td>10</td>
<td>2.830 1.000000e+00</td>
<td>8.774270e-05</td>
<td>1.000000e-01</td>
<td>8.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>17</td>
<td>10</td>
<td>2.640 1.000000e+00</td>
<td>4.838420e-05</td>
<td>1.000000e-01</td>
<td>7.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>384</td>
<td>18</td>
<td>10</td>
<td>2.500 1.000000e+00</td>
<td>3.950640e-05</td>
<td>1.000000e-01</td>
<td>7.000000e-02</td>
<td>PASS</td>
</tr>
<tr>
<td>1 384</td>
<td>19</td>
<td>10</td>
<td>2.310 1.000000e+00 3.551140e-05 1.000000e-01 8.375000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>20</td>
<td>10</td>
<td>2.150 1.000000e+00 2.500590e-05 1.000000e-01 8.500000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>21</td>
<td>10</td>
<td>1.980 1.000000e+00 3.181230e-05 1.000000e-01 7.625000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>22</td>
<td>10</td>
<td>1.810 1.000000e+00 3.299600e-05 1.000000e-01 8.000000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>23</td>
<td>10</td>
<td>1.670 1.000000e+00 2.618960e-05 1.000000e-01 9.125000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>24</td>
<td>10</td>
<td>1.530 1.000000e+00 3.136840e-05 1.000000e-01 7.875000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>25</td>
<td>10</td>
<td>1.400 1.000000e+00 2.663350e-05 1.000000e-01 8.375000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>26</td>
<td>10</td>
<td>1.270 1.000000e+00 3.255210e-05 1.000000e-01 8.625000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>27</td>
<td>10</td>
<td>1.140 1.000000e+00 2.692950e-05 1.000000e-01 7.500000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>28</td>
<td>10</td>
<td>0.999 1.000000e+00 5.149150e-05 1.000000e-01 9.250000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>29</td>
<td>10</td>
<td>0.889 1.000000e+00 3.225620e-05 1.000000e-01 8.750000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>30</td>
<td>10</td>
<td>0.772 1.000000e+00 3.536340e-05 1.000000e-01 9.375000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>31</td>
<td>10</td>
<td>0.650 1.000000e+00 4.113400e-05 1.000000e-01 9.125000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 384</td>
<td>32</td>
<td>10</td>
<td>0.547 1.000000e+00 3.965440e-05 1.000000e-01 8.750000e-02 PASS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(continues on next page)
| 1 384 33 | 10 | 0.428 1.000000e+00 5.489460e-05 1.000000e-01 | 9.625000e-02 PASS |
| 1 384 34 | 10 | 0.333 1.000000e+00 5.030780e-05 1.000000e-01 | 8.875000e-02 PASS |
| 1 384 35 | 10 | 0.220 1.000000e+00 3.906250e-05 1.000000e-01 | 8.875000e-02 PASS |
| 1 384 36 | 10 | 0.127 1.000000e+00 2.929690e-05 1.000000e-01 | 8.250000e-02 PASS |
| 1 384 37 | 10 | 0.034 1.000000e+00 3.225620e-05 1.000000e-01 | 9.000000e-02 PASS |
| 1 384 38 | 10 | -0.066 1.000000e+00 2.929690e-05 1.000000e-01 | 8.375000e-02 PASS |
| 1 384 39 | 10 | -0.170 1.000000e+00 2.737330e-05 1.000000e-01 | 8.500000e-02 PASS |
| 1 384 40 | 10 | -0.253 1.000000e+00 3.521540e-05 1.000000e-01 | 7.500000e-02 PASS |
| 1 384 41 | 10 | -0.344 1.000000e+00 5.563450e-05 1.000000e-01 | 9.375000e-02 PASS |
| 1 384 42 | 10 | -0.424 1.000000e+00 2.559780e-05 1.000000e-01 | 8.750000e-02 PASS |
| 1 384 43 | 10 | -0.515 1.000000e+00 4.690460e-05 1.000000e-01 | 9.000000e-02 PASS |
| 1 384 44 | 10 | -0.605 1.000000e+00 5.755800e-05 1.000000e-01 | 9.000000e-02 PASS |
| 1 384 45 | 10 | -0.693 1.000000e+00 3.980230e-05 1.000000e-01 | 8.125000e-02 PASS |
| 1 384 46 | 10 | -0.766 1.000000e+00 5.208330e-05 1.000000e-01 | 9.000000e-02 PASS |
Plots of current SNR values used for BLER testing are shown below:

![Plots of current SNR values](image)

1.3.5.5 SCF FAPI Support

1.3.5.5.1 Overview

Aerial cuBB supports the 5G FAPI 222.10.02 defined by the Small Cell Forum. This release supports most of the control interface (P5) and data path interface (P7) SCF messages.

1.3.5.5.2 SCF FAPI Messages Supported

The table below summarizes the status of the SCF FAPI messages supported.

<table>
<thead>
<tr>
<th>SCF Messages</th>
<th>PDU Types</th>
<th>SCF L2 Adapter</th>
<th>SCF TestMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL_TTI.request</td>
<td>PDCCH</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PDSCH[7]</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CSI-RS</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>SSB</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>UL_TTI.request</td>
<td>PRACH</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PUSCH[7]</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PUCCH</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Table 12 – continued from previous page

<table>
<thead>
<tr>
<th>SCF Messages</th>
<th>PDU Types</th>
<th>SCF L2 Adapter</th>
<th>SCF TestMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>UL_DCI.request</td>
<td>PDCCH</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SLOT errors</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>TX_Data.request[1]</td>
<td>PDSCH</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rx_Data.indication[1]</td>
<td>PUSCH (also contains RNTI, HARQ Id, UL_CQI, Timing adv, RSSI)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>CRC.indication</td>
<td>CRC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>UCI.indication</td>
<td>PUSCH[8]</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PUCCH format 0,1</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>PUCCH format 2,3,4</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>SR for format 0,1</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>SR for format 2,3,4</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>HARQ for format 0,1</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>HARQ for format 2,3,4</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CSI part 1</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>CSI part 2</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>RSSI and UL SINR metrics</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SRS.indication[5][6]</td>
<td>SRS</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>RACH.indication</td>
<td>PRACH</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Config.request[2]</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Config.response</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Start.request</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Stop.request</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Stop.indication</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Error.indication</td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Param.request</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Param.response</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

**Note[1]:** The SCF implementation is based on SCF_222.10.02, but with the following exceptions:

- PDU Length of TX_DATA.request and RX_DATA.indication are changed to 32-bits. This is defined in SCF_222.10.03.
- The implementation supports multiple UE per TTI when the TLV tag is 2 in each PDU. However, the offset value in the TLV is ignored and L1 assumes all TBs in that slot placed in a flat buffer one after the other.
- The RX_DATA.indication FAPI message contains the MAC PDU (TB data) in the data_buf of the NVIPC message.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_DATA.request</td>
<td>uint16_t</td>
<td>The total length (in bytes) of the PDU description and PDU data, without the padding bytes. Value: 0 65535. Change type to uint32_t, value range is: 0 2^32 -1. [NVIDIA change]: Use it as the PDU data (TB data) size without the PDU description.</td>
</tr>
<tr>
<td>PDU Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX_DATA.Indication</td>
<td>uint16_t</td>
<td>The length of PDU in bytes. A length of 0 indicates a CRC or decoding error. Value: 0 65535. Change type to uint32_t, value range is: 0 2^32 -1.</td>
</tr>
<tr>
<td>PDU Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RX_DATA.Indication</td>
<td>Variable</td>
<td>The contents of PDU. This will be a MAC PDU. [NVIDIA workaround]: Removed this field, do not parse it in wireshark dissector. For SCF_222.10.04, although the tag value is set to 1, the MAC PDU is still delivered in a separate NVIPC buffer.</td>
</tr>
<tr>
<td>PDU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UL_TTI.request</td>
<td>uint8_t</td>
<td>Indicates SR. Only valid for format 0 and 1. [NVIDIA workaround]: Enhance to use it as BitLenSr for format 2, 3, 4.</td>
</tr>
<tr>
<td>SRFlag</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note[2]:** Precoding Matrix (Table 3-33) with vendor tag 0xA011 is supported. Digital beam table (Table 3-32) is not supported.

**Note[3]:** For NZP CSI-RS, only 4 antennas and single CSI-RS PDU.

**Note[4]:** The current implementation supports multi-bit SR over PUCCH format 2, 3, and 1. Because SCF FAPI 10.02 doesn’t provide any field explicitly suggesting the bit length of the SR in the PUCCH_PDU of UL_TTI.request, use the SRFlag field to provide the SR bit length. For example, if the desired SR bit length is 3, set SRFlag = 3.

**Note[5]:** SRS.indication and SRS PDU in UL_TTI.request are supported according to SCF FAPI 222.10.02. SRS can be enabled when flag enable_srs is set in the cuphycontroller_.xxx.yaml file i.e. enable_srs: 1.
Note[6]: SRS.indication and SRS PDU in UL_TTI.request are also supported according to SCF FAPI 222.10.04, which needs to be enabled with the "-DSCF_FAPI_10_04=ON" build option and flag enable_srs is set in the cuphycontroller_xxx.yaml file i.e. enable_srs: 1, as described in Running cuBB End-to-End.

- The format of the SRS.indication message is given in SCF FAPI 222.10.04 Table 3-129; the report TLV is defined in Table 3-130.
- The supported report type is Normalized Channel I/Q Matrix defined in Table 3.132 for codebook or nonCodebook SRS usage.
- The SRS Report TLV tag is 1 (customized value), the length is the actual report size in bytes without padding, the value field has the offset (in bytes) into the data_buf portion of NVIPC message for each SRS PDU. The report data is placed in the data_buf portion of the NVIPC message for all SRS PDUs.
- In case of wideband SRS, it is possible that the data_buf portion of NVIPC message carrying SRS.indication does not have enough space to accomodate SRS channel vectors for all the SRS PDUs. In this case, Aerial supports splitting of SRS.indication into multiple message. This feature can be enabled using CONFIG TLV 0x102B / indicationInstancesPerSlot as defined in 5G FAPI 222.10.04 specification table 3-36 for PHY configuration. If this TLV is not enabled by L2 and SRS.indication cannot accomodate all the SRS channel vectors, the SRS.indication will carry partial SRS information. On processing such a SRS PDU, an error indication with error code 0x35 is sent to L2 indicating partial SRS indication.
- Additionally Table 3.131 FAPIv3 Beamforming report, with PRG-level resolution for beamManagement SRS usage is also supported. The SRS Report TLV tag is 2 (customized value), is defined for encoding the SINR reports in the msg_buf at an offset of 32 bit from the value field, the length is the actual report size in bytes without padding. Also, currently PRG size of 2 is only supported.

Note[7]: If flag mMIMO_enable is set in the cuphycontroller_xxx.yaml file i.e. mMIMO_enable: 1 to enable Dynamic Beamforming, indicates that the L2 shall encode the TX Precoding and Beamforming PDU & RX Beamforming PDU to include fields for numPRGs, prgSize and digBFInterface but L2 shall not encode the beamIdx because when Dynamic Beamforming is used, L2 does not have information available for beamIds but L2 needs to provide the remaining information in the PDU to L1.

Note[8]: To get HARQ values in UCI.indication for UCI on PUSCH, before complete PUSCH slot processing, L2 should include PHY configurationTLV 0x102B (indicationInstancesPerSlot) with UCI.indication set to 2, according to Table 3–36 in SCF FAPI 222.10.04. If UCI.indication set to 2 in config.request for any cell the early HARQ feature will get activated for all cells.

1.3.5.5.3 Vendor Specific Message

A new vendor specific message SLOT.response was added after the 22-4 release. Before the 22-4 release, L2 has to set an event using the nvIPC notify function to inform L1 about “EOM” after sending the last FAPI message. This works well for single cell and when all FAPI messages are on time. L1 also uses the nvIPC notify function to set an event after sending each message.

The new SLOT.response FAPI message is used by L2 as the last FAPI message for each cell in each slot. It has the following advantages:

- It works as “EOM” for each cell in each slot.
- Each cell sends a SLOT.response as the last FAPI message of each slot.
- L2 should send SLOT.response even in empty slots (i.e. slots that have no scheduling).
- A “Dummy” or empty DL/UL TTI are optional/not-required.
The notify event from L2 is optional/not-required.

The SLOT.response message format is shown below:

```c
typedef struct
{
    scf_fapi_body_header_t msg_hdr;
    uint16_t sfn;
    uint16_t slot;
} __attribute__((__packed__)) scf_fapi_slot_rsp_t;
```

Message-id 0x8F is used for this message

```c
{ ...
    SCF_FAPI_RX_PRACH_INTERFERENCE_INDICATION = 0x8E,
    SCF_FAPI_SLOT_RESPONSE = 0x8F,
    SCF_FAPI_RESV_2_END = 0xFF,
} scf_fapi_message_id_e;
```

L1 continues to send a notify event after all FAPI messages to L2 to minimize impact on L2.

1.3.5.5.3.1 Message Sequence

An example message sequence is shown below:
Note: On receiving the first SLOT.indication, L2 is unable to send SLOT.response for 2-3 slots because it has a slot advance of 3.
1.3.5.5.3.2 Impact of Late Messages

- All messages are late for a cell (DL_TTI+TX_DATA+UL_DCI or UL_TTI)
  - All messages are dropped for the said cell. No impact on other cells.
- DL_TTI arrived on time but TX_DATA.request is late for a cell
  - This is considered as a partial slot. Due to cell grouping, PDSCH & DL-PDCCH is dropped for all cells.
- UL_TTI is late for a cell
  - ULSCH is not processed for the said cell. No impact on other cells.
- UL_DCI is late for a cell
  - UL-PDCCH is not processed for the said cell. No impact on other cells.
- SLOT.response is late for a cell
  - All FAPI messages received in time will be processed for the cell.

1.3.5.5.3.3 How to Enable or Disable SLOT.response

This feature is enabled by default in L1 after the 23-1 release. When integrating with L2, L2 is required to send this vendor-specific message in the manner described above.

Option ENABLE_L2_SLT_RSP should be configured with the same value in L1, L2 and libnvipc.so standalone build for L2. Refer to cuBB Quickstart Guide for details.

If L2 doesn't support the SLOT.response message, disable this feature by setting the “-ENABLE_L2_SLT_RSP=OFF” flag in the cmake command:

```
cmake <existing flags> -DENABLE_L2_SLT_RSP=OFF
```

Once the feature is enabled, the following is true:

- L2 has to send a vendor-specific SLOT.response message as the last FAPI message for each cell.
  - L2 to send this message even in empty slot (where nothing is scheduled).
- allowed_fapi_latency is deprecated and presumed to be 0.
  - L2 to complete sending all FAPI messages within the 500 us time-budget marked by SLOT.indication from L1.
  - Late FAPI messages will be dropped.
- A “Dummy” DL/UL TTI messages in empty slots is optional.
- A notify event after sending all FAPI messages is optional.
  - ipc_sync_mode in the L2 Adapter config file is deprecated.
- L1 will continue to send a Notify event after all FAPI messages to minimize impact on L2.
1.3.5.5.4 Dynamic Beamforming for 32T32R

To enable this feature in Aerial software, flag mMIMO_enable should be set/introduced in the cuphycontroller_xxx.yaml file i.e. `mMIMO_enable: 1`.

Two additional TLVs are required in CONFIG.req:

- **TLV 0xA016 denoting NUM_TX_PORT (uint8_t)**
  
  This field specifies the number of DL BB ports for PHY. 5G FAPI 222.10.04 described the field numTxAnt and numRxAnt in Table 3-37 as - 'numTxAnt cannot exceed the number of DL BB ports for the PHY'. Hence the fields in table 3-37 represent the logical antenna ports.

- **TLV 0xA017 denoting NUM_RX_PORT (uint8_t)**
  
  This field specifies the number of UL BB ports for PHY. 5G FAPI 222.10.04 described the field numTxAnt and numRxAnt in Table 3-37 as - 'numRxAnt cannot exceed the number of UL BB ports for the PHY'. Hence the fields in table 3-37 represent the logical antenna ports.

5G FAPI 223 describes baseband ports as a mapping between layers to RU TX/RX ports. PHY needs to know the BB ports from L2 (see Fig 3-3 in SCF-223.2.0.4).

This field will be used by PHY to read the number of DL BB ports.

If the TLV is not received from L2 and flag mMIMO_enable is set in the cuphycontroller_xxx.yaml file i.e. `mMIMO_enable: 1`, the default value for number of DL BB ports is set to 8.

- **TLV 0xA017 denoting NUM_RX_PORT (uint8_t)**

  This field specifies the number of UL BB ports for PHY. 5G FAPI 222.10.04 described the field numTxAnt and numRxAnt in Table 3-37 as - 'numRxAnt cannot exceed the number of UL BB ports for the PHY'. Hence the fields in table 3-37 represent the logical antenna ports.

5G FAPI 223 describes baseband ports as a mapping between layers to RU TX/RX ports. PHY needs to know the BB ports from L2 (see Fig 3-3 in SCF-223.2.0.4).

This field will be used by PHY to read the number of UL BB ports.

If the TLV is not received from L2 and flag mMIMO_enable is set in the cuphycontroller_xxx.yaml file i.e. `mMIMO_enable: 1`, the default value for number of UL BB ports is set to 4.

DL & UL TTI have an additional field added for TRP scheme. See Note-6 in SCF FAPI Messages supported section.

Dynamic Beamforming is supported for PDSCH (RA type-1 and without CSI-RS) and PUSCH only.

A UE that is scheduled for SRS on S-slot should not be scheduled for dynamic beamforming of PDSCH and PUSCH in subsequent D & U slots until SRS indication for the UE is received. This prevents a race condition between L1 and L2 where the SRS channel vectors have been updated in the GPU hosted memory, but the latest SRS channel vectors are yet to be sent to L2. In this case, L2 might make a scheduling decision based on stale SRS channel vectors and the BFW calculation might happen with refreshed SRS channel vectors.

Two new FAPI messages have been defined from L2 to L1 to implement beamforming weight calculation in L1 as follows:

- **SCF_FAPI_DL_BFW_CVI_REQUEST = 0x90**
- **SCF_FAPI_UL_BFW_CVI_REQUEST = 0x91**

Structure of the FAPI message from L2 to L1 for beamforming weight calculation are as below. The same message structure is used for DL(PDSCH) and UL(PUSCH). When used for DL(PDSCH), it is referred to as DLBFW_CVI.request and when used for UL(PUSCH), it is referred to as ULBFW_CVI.request.
Table 3-1001 DLBFW_CVI.request message body

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFN</td>
<td>uint16_t</td>
<td>SFN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value: 0 -&gt; 1023</td>
</tr>
<tr>
<td>Slot</td>
<td>uint16_t</td>
<td>Slot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value: 0 -&gt; 159</td>
</tr>
<tr>
<td>nPDUs</td>
<td>uint8_t</td>
<td>Number of PDUs that are included in this message. All PDUs in the message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are numbered in order. Each PDU is corresponding to a UE Group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value: 0 -&gt; 255</td>
</tr>
</tbody>
</table>

For Number of PDUs:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDUSize</td>
<td>uint16_t</td>
<td>Size of the PDU control information (in bytes). This length value includes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the 4 bytes required for the PDU type and PDU size parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value 0 -&gt; 65535</td>
</tr>
<tr>
<td>DLBFW CVI Configuration</td>
<td>structure</td>
<td>See Table 3-1002 DLBFW CVI PDU</td>
</tr>
</tbody>
</table>

Table 3-1002 DLBFW CVI PDU
Timeline for receiving DLBFW_CVI.request and ULBFW_CVI.request is as shown below:

Downlink timeline for slot N (32T32R)
1.3.5.5 Additional Aerial Specific Error Codes Reported in ERROR.indication from L1 to L2

Additional Aerial specific error codes have been added, starting from value 0x33, and L2 may receive these error codes in ERROR.indication message from L1 to L2. For example:

- `SCF_ERROR_CODE_FAPI_END = 0x32`, //Vendor specific error codes — begin
- `SCF_ERROR_CODE_L1_PROC_OBJ_UNAVAILABLE_ERR = 0x33`,
- `SCF_ERROR_CODE_MSG_LATE_SLOT_ERR = 0x34`, //Indicates that L1’s timer thread did not wake up on the slot boundary and slot indication for the indicated SFN,slot is late and will not be sent from L1 to L2
- `SCF_ERROR_CODE_PARTIAL_SRS_IND_ERR = 0x35`, //Indicates partial SRS indication
- `SCF_ERROR_CODE_L1_DL_CPLANE_TX_ERROR = 0x36`, //Indicates a DL C-plane transmission error (Timing/Functional)
- `SCF_ERROR_CODE_L1_UL_CPLANE_TX_ERROR = 0x37`, //Indicates a UL C-plane transmission error (Timing/Functional)
- `SCF_ERROR_CODE_L1_DL_GPU_ERROR = 0x38`, //Indicates a DL GPU pipeline processing error
- `SCF_ERROR_CODE_L1_DL_CPU_TASK_ERROR = 0x39`, //Indicates a DL CPU Task incompletion error
- `SCF_ERROR_CODE_L1_UL_CPU_TASK_ERROR = 0x3A`, //Indicates a UL CPU Task incompletion error
- `SCF_ERROR_CODE_L1_P1_EXIT_ERROR = 0x3B`, //Indicates Part 1 of the error indication during L1 app exit process
SCF_ERROR_CODE_L1_P2_EXIT_ERROR = 0x3C, //Indicates Part 2 of the error indication during L1 app exit process post cudaDeviceSynchronize if CUDA coredump env variables are set

SCF_ERROR_CODE_L1_DL_CH_ERROR = 0x3D, //Indicates DL channel run (CPU/GPU) error
SCF_ERROR_CODE_L1_UL_CH_ERROR = 0x3E //Indicates UL channel run (CPU/GPU) error

1.3.5.6 Limitations

1.3.5.6.1 Known Limitations

- The cuPHY library and binaries are intended for the Linux environment on the qualified platforms only.
- The supported configurations are limited to those listed above. Other configurations are not supported and may not perform well.
- Only homogeneous configurations supported for multiple cells.
- The configurable YAML parameters enable_h2d_copy_thread, h2d_copy_thread_cpu_affinity, and h2d_copy_thread_sched_priority are optional in the cuphycontroller YAML file. If these parameters are not present, the code uses the default values and throws the exception “YAML invalid key:” on the cuphycontroller console. This exception message has no impact on the functionality and can be disregarded.
- GPU Initiated Comms for DL (gpu_init_comms_dl flag in the cuphycontroller config yaml) is required to be enabled by default from 22-2.4 release onwards. The flag enables the feature within Aerial L1 to engage GPU kernels to prepare and send U-Plane packets on the DL as opposed to CPU Initiated Comms (gpu_init_comms_dl=0) which exercises CPU code/consumes CPU cycles to prepare/send U-plane packets on the DL.
- No simultaneous DL and UL scheduling in S-slot. However, DL-only s-slot is supported in E2E test with O-RU.
- When the FAPI messages for a given cell are sent via nvipc, L1 expects an explicit notify (once per cell) via nvipc. In the case of multiple cells, multiple explicit notify APIs be called from L2. When a cell doesn’t have any messages for a given slot, L1 expects dummy DL_TTI and/or UL_TTI.request, that is (nPDU = 0), to be sent “per cell”. If the Slot Response feature is enabled by compiling Aerial with -DENABLE_L2_SLT_RSP=ON, this step is optional.
- For multi cells operation, L2 can signal the L2Adapter in 2 ways:
  - Single event per slot: which contains SCF FAPI messages for all cells. The single event is raised by calling nvipc notify(1) once per slot after the messages for all the cells are sent.
  - Single event per cell: which is signaled by L2 after all FAPI messages for a given cell are sent. It is expected that multiple nvipc notify(1) are called for multiple cells. The number of times that notify is being called must be the same as the number of active cells. A cell is marked active after START.req is received from L2. In this case, L1 expects dummy DL_TTI and UL_TTI described above. This is the default behavior.

To select the operation mode, set the ipc_sync_mode in yaml:

```
# Option 1: Sync per slot
ipc_sync_mode: 0
# Option 2: Sync per active cell
ipc_sync_mode: 1
```
If Slot Response feature is enabled by compiling Aerial with \-DENABLE_L2_SLT_RSP=ON, this setting is a no-op as L1 does not expect any event from L2.

- Cell life cycle management:
  - All cells have to be configured before any cell start.
  - No In-service configuration update.
  - CONFIG.request received in CONFIGURED (Out-of-Service) state can be used to change PCI and the supported PRACH parameters specified in dynamic PRACH section in cuBB quickstart guide only. PHY ignores any other TLVs received in CONFIG.request. If CONFIG.response indicates success, then only PCI and supported PRACH parameters are changed. All other parameters remain as in the initial CONFIG.request received for the cell.
  - PHY reconfiguration of a cell in CONFIGURED (Out-of-Service) state can take up to 40 ms to complete (details below). Another CONFIG.request for any cell during this time (around 20 ms) that occurs before receiving a CONFIG.response returns a CONFIG.response with the error code “MSG_INVALID_STATE”. The ERROR.indication will NOT be sent for this error. L2 needs to wait to receive a CONFIG.response before sending a CONFIG.request for another cell in CONFIGURED state.
    - If Aerial is configured for 4 cells and 3 cells are In-service with data running, reconfiguration of 1 cell (Out-of-Service) can take around 40 ms to complete
    - If Aerial is configured for 4 cells and 3 cells are In-service with no data running, reconfiguration of 1 cell (O-RU) can take around 20 ms to complete
    - If CONFIG.response is received with error code “MSG_INVALID_CONFIG”, then reconfiguration was unsuccessful and the cell is still with the configuration received in initial CONFIG.request.
    - No UE attach allowed in all cells during the reconfiguration time.

- Dynamic M-plane parameters:
  - When OAM sends gRPC message to change MAC address in M-plane, it must be a valid O-RU MAC address.
  - Aerial supports only single section per packet in the UL.
  - The nvlog_observer and nvlog_collect are deprecated in 23-1.
  - F13 test cases are deprecated in 23-2.

- Early HARQ in UCI.indication:
  - This feature is supported only for the first UL slot (x4 slots) and when all the early-HARQ bits are resident in symbols 0-3.
  - UCI.Indication with early HARQ will not have any measurement values.
  - If only HARQ is scheduled on PUSCH then with this feature enabled, no UCI.indication will be sent to L2 after full slot processing of PUSCH. Consequently no measurements for that slot will be reported to L2.
  - If CSI reports are also scheduled on PUSCH along with HARQ, then UCI.indication with early HARQ will not have any measurement values. But the UCI.indication sent after full slot processing of PUSCH will have the measurements.
  - A constraint to enable early-HARQ is that these HARQ bits should be fully resident in OFDM symbols 0-3. So HARQ bits resident in OFDM symbols 0-3 will be in the 1st UCI.indication (that is, early-HARQ indication) and all other HARQ bits in the subsequent UCI.indication (that is, after full slot PUSCH processing completes).
Multiple cell operation without issuing dummy config.req:
- L2 should wait for at least 40msec between two CONFIG.request even at the initial stage, so that CONFIG.response is received by L2.
- L2 can retry the failed CONFIG.request for a given cell after 1 sec.

Multi-L2 with single cuphycontroller per GPU:
- The total cell number of all L2 instances cannot exceed the cell_group_num configured in cuphycontroller.yaml.
- nvIPC only supports static cell allocation defined in the nvipc_multi_instances.yaml for multiple L2 instances. The number of cells and the cell mapping in each L2 instance cannot change after L1 is configured.
- Support dynamic cell start/stop in each L2 instance. Do not support dynamic L2 restart. L2 instance needs to hold the nvipc instance after connecting to L1.

32T32R TDD single cell:
- DL SU/MU-MIMO Number of Layers combinations
  - 1 UE in a Group: 1 or 2 or 4 layers
  - 2 UEs in a Group: 1+1 layers or 2+2 layers
  - 4 UEs in a Group: 1+1+1+1 layers
- UL SU/MU-MIMO Number of Layers combinations
  - 1 UE in a Group: 1 or 2 layers
  - 2 UEs in a Group: 1+1 or 2+2 layers
- PDSCH Resource Allocation Type 0 (RAT0) is not supported.
- Partial PRB allocation is not supported for the UE’s in one group. Only full bandwidth can be allocated with PRG size 2.
- Currently PRG size of 2 is only supported for SRS & BFW.
- SRS channel is only supported in the special slot along with no other UL channels.
- SRS is not supported in the UL Slot, due to the presence of other UL channels.
- SRS reports related to antennaSwitching (FAPI 222.10.04, Table 3-133 - Channel SVD Representation) is not supported.
- The following test cases are not verified:
  - NZP-CSI-RS & TRS in combination with PDSCH.
  - Multiple UE Groups with UE’s in each group containing same set of PRB’s.
  - Multi-slot SRS + BFW + PDSCH/PUSCH + all other UL/DL channels.

64T64R TDD single cell:
- PDSCH Resource Allocation Type 0 (RAT0) is not supported.
- Partial PRB allocation is not supported for the UE’s in one group. Only full bandwidth can be allocated with PRG size 2.
- Currently PRG size of 2 is only supported for SRS & BFW.
- SRS channel is only supported in the special slot along with no other UL channels.
- SRS is not supported in the UL Slot, due to the presence of other UL channels.
SRS reports related to antennaSwitching (FAPI 222.10.04, Table 3-133 - Channel SVD Representation) is not supported.

- The following test cases are not verified:
  - NZP-CSI-RS & TRS in combination with PDSCH.
  - Multiple UE Groups with UE’s in each group containing same set of PRB’s.
  - Multi-slot SRS + BFW + PDSCH/PUSCH + all other UL/DL channels.

### 1.3.5.6.2 Known Issues

- There is a known issue to run Aerial L1 in MIG mode while using GPU driver 555.42.02. The workaround is to downgrade the GPU driver to 550.54.15.
- cuBB test case 7600 reports CRC error when debug synch check is enabled. Issue only appears when certain versions of synch debug tool are enabled.
- The support for CPU Initiated Comms (gpu_init_comms_dl=0) mode is no longer available after the 22-2.4 release and it is recommended that this mode not be enabled for testing purposes.
- Support up to 8 DMRS ports, if the allocations are contiguous in PDSCH and PUSCH.
- Some DOCA error messages are not real errors. For example, the following messages contain debug info:
  ```
  E [FH,QUEUE] Doca RxQ created! ...
  [DOCA][ERR][DOCA_GPU::m1x5:1229] ...
  ```
- Changing shm_log_level to 6 or 7 in nvlog_config.yaml causes a crash in the msg_processing thread.
- SCHED_FIFO + 100% CPU poll thread causes the system to hang on the 5.4.0-65-lowlatency kernel. The solution is one of the following:
  - Configure the kernel option CONFIG_RCU_NOCB_CPU=y, recompile, and install the kernel.
  - Upgrade the host system to 5.15.0-71-lowlatency or later.
- 32T32R TDD single cell:
  - SRS + UL_BFW + PUSCH is failing due to Order Kernel Timeout.
  - GPU/CPU Stats (DRV.MAP_DL & DRV.MAP_UL) not coming in phy logs for Aggr DL_BFW & Aggr UL_BFW.
  - Partial bandwidth allocation for 100Mhz is failing i.e. launch_pattern 91104 and 91105.
- CUDA application on Grace Hopper:
  - CUDA applications on the Grace Hopper platform require ATS support. Currently, ATS is not enabled on the arm64 platform when IOMMU passthrough is enabled.
- NIC string conversion issue on Grace Hopper:
  - While working on dynamic CPU core assignments in K8s pod, we need to parse and dump the cuphycontroller config yaml file. On the Grace Hopper, the `nic: 0000:01:00.0` will be converted to `nic: 60.0`. This is because the PCIe address might be interpreted as a 60 based integer according to `https://yaml.org/type/int.html`. The fix is to explicitly tell yaml parser to interpret the PCIe address as a string by putting single quotation marks around or `!!str` before the pcie address, e.g., `nic: ’0000:01:00.0’` or `nic: !!str 0000:01:00.0`. 

1.3. Aerial cuPHY
From
```
sed -i \"s∕nic:.*∕nic: 0000:01:00.0∕\" \${cuBB_SDK}/cuPHY-CP/cuphycontroller/
   \→ config/cuphycontroller_P5G_FXN.yaml
```
to
```
sed -i \"s∕nic:.*∕nic: \'0000:01:00.0\'/\" \${cuBB_SDK}/cuPHY-CP/cuphycontroller/
   \→ config/cuphycontroller_P5G_FXN.yaml
```
(or sed -i \"s∕nic:.*∕nic: !\![\str 0000:01:00.0∕\" \${cuBB_SDK}/cuPHY-CP/
   \→ cuphycontroller/config/cuphycontroller_P5G_FXN.yaml)

▶ The following test cases are not passing. They could be functionality issues or test framework issues:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Test Cases</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDSCH</td>
<td>3853</td>
<td>32TR</td>
</tr>
<tr>
<td>PUSCH</td>
<td>7374, 7476</td>
<td>RKHS ChEst</td>
</tr>
<tr>
<td></td>
<td>7599, 7600</td>
<td>Multiple CSIP2</td>
</tr>
<tr>
<td>mSlot_mCell</td>
<td>90027-90029, 90064-90067</td>
<td>DSUUU</td>
</tr>
<tr>
<td></td>
<td>90071, 90081, 90083</td>
<td>32TR</td>
</tr>
<tr>
<td></td>
<td>90070, 90073, 90077</td>
<td>32TR (AX100 only)</td>
</tr>
<tr>
<td></td>
<td>900017, 90137, 90138, 90140, 90143</td>
<td>64TR</td>
</tr>
<tr>
<td></td>
<td>90605</td>
<td>Multiple SRS</td>
</tr>
<tr>
<td></td>
<td>90608, 90606, 90607</td>
<td>64TR static+dynamic BF</td>
</tr>
</tbody>
</table>

1.3.5.7 Acknowledgements

1.3.5.7.1 Abseil

Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.
"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

1.3.5.7.2 Backward-cpp

Copyright 2013 Google Inc. All Rights Reserved.

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

1.3.5.7.3 BoringSSL

BoringSSL is a fork of OpenSSL. As such, large parts of it fall under OpenSSL
licensing. Files that are completely new have a Google copyright and an ISC
license. This license is reproduced at the bottom of this file.

Contributors to BoringSSL are required to follow the CLA rules for Chromium:
https://cla.developers.google.com/clas

Files in third_party/ have their own licenses, as described therein. The MIT
license, for third_party/fiat, which, unlike other third_party directories, is
compiled into non-test libraries, is included below.

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the
OpenSSL License and the original SSLeay license apply to the toolkit. See below
for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

The following are Google-internal bug numbers where explicit permission from some authors is recorded for use of their work. (This is purely for our own record keeping.)
   27287199
   27287880
   27287883

OpenSSL License
---------------

/* ====================================================================
 * Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http:∕∕www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http:∕∕www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT `AS IS' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 */

(continues on next page)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* ====================================================================
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*/

Original SSLeay License
-----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 */

* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
* *
* This library is free for commercial and non-commercial use as long as
* the following conditions are adhered to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* Ihash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
* *
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
* *
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the routines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the app directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
* *
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG `AS IS' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
*
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*
ISC license used for completely new code in BoringSSL:

/* Copyright (c) 2015, Google Inc.
 *
* Permission to use, copy, modify, and/or distribute this software for any
* purpose with or without fee is hereby granted, provided that the above
* copyright notice and this permission notice appear in all copies.
* *
* THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
* WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
* MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
* SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
* WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION
* OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
* CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. */

The code in third_party/fiat carries the MIT license:

Copyright (c) 2015-2016 the fiat-crypto authors (see

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
Aerial CUDA-Accelerated RAN, Release 24-2

Licenses for support code
-------------------------

Parts of the TLS test suite are under the Go license. This code is not included in BoringSSL (i.e. libcrypto and libssl) when compiled, however, so distributing code linked against BoringSSL does not trigger this license:

Copyright (c) 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BoringSSL uses the Chromium test infrastructure to run a continuous build, trybots etc. The scripts which manage this, and the script for generating build metadata, are under the Chromium license. Distributing code linked against BoringSSL does not trigger this license.

Copyright 2015 The Chromium Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

(continues on next page)
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.3.5.7.4 Benchmark

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work

(continues on next page)
"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the

(continues on next page)
appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same “printed page” as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the “License”);
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to

(continues on next page)
communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same “printed page” as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.3.5.7.6 c-ares

# c-ares license

Copyright (c) 2007 - 2018, Daniel Stenberg with many contributors, see AUTHORS file.

Copyright 1998 by the Massachusetts Institute of Technology.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
1.3.5.7.7 CivetWeb

Copyright (c) 2013-2021 The CivetWeb developers (CREDITS.md)
Copyright (c) 2004-2013 Sergey Lyubka
Copyright (c) 2013 No Face Press, LLC (Thomas Davis)
Copyright (c) 2013 F-Secure Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the
Software without restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so, subject to the
following conditions:
The above copyright notice and this permission notice shall be included in all copies
or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.3.5.7.8 Data plane API

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each (continues on next page)
Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner].

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
1.3.5.7.9 DPDK

The DPDK uses the Open Source BSD-3-Clause license for the core libraries and drivers. The kernel components are naturally GPL-2.0 licensed.

Including big blocks of License headers in all files blows up the source code with mostly redundant information. An additional problem is that even the same licenses are referred to by a number of slightly varying text blocks (full, abbreviated, different indentation, line wrapping and/or white space, with obsolete address information, ...) which makes validation and automatic processing a nightmare.

To make this easier, DPDK uses a single line reference to Unique License Identifiers in source files as defined by the Linux Foundation's SPDX project (https://spdx.org/).

Adding license information in this fashion, rather than adding full license text, can be more efficient for developers; decreases errors; and improves automated detection of licenses. The current set of valid, predefined SPDX identifiers is set forth on the SPDX License List at https://spdx.org/licenses/.

DPDK uses first line of the file to be SPDX tag. In case of *#!* scripts, SPDX tag can be placed in 2nd line of the file.

For example, to label a file as subject to the BSD-3-Clause license, the following text would be used:

SPDX-License-Identifier: BSD-3-Clause

To label a file as GPL-2.0 (e.g., for code that runs in the kernel), the following text would be used:

SPDX-License-Identifier: GPL-2.0

To label a file as dual-licensed with BSD-3-Clause and GPL-2.0 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

SPDX-License-Identifier: (BSD-3-Clause OR GPL-2.0)

To label a file as dual-licensed with BSD-3-Clause and LGPL-2.1 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

SPDX-License-Identifier: (BSD-3-Clause OR LGPL-2.1)

Any new file contributions in DPDK shall adhere to the above scheme. It is also being recommended to replace the existing license text in the code with SPDX-License-Identifiers.

Any exception to the DPDK IP policies shall be approved by DPDK Tech Board and DPDK Governing Board. Steps for any exception approval:

1. Mention the appropriate license identifier form SPDX. If the license is not listed in SPDX Licenses. It is the submitters responsibility to get it first listed.
2. Get the required approval from the DPDK Technical Board. Technical Board may advise the author to check alternate means first. If no other alternative

(continues on next page)
are found and the merit of the contributions are important for DPDK's mission, it may decide on such exception with two-thirds vote of the members.

3. Technical Board then approach Governing Board for such limited approval for the given contribution only.

Any approvals shall be documented in "Licenses/exceptions.txt" with record dates.

DPDK project supported licenses are:

1. BSD 3-clause "New" or "Revised" License
   SPDX-License-Identifier: BSD-3-Clause
   URL: http://spdx.org/licenses/BSD-3-Clause#licenseText
   DPDK License text: licenses/bsd-3-clause.txt
2. GNU General Public License v2.0 only
   SPDX-License-Identifier: GPL-2.0
   URL: http://spdx.org/licenses/GPL-2.0.html#licenseText
   DPDK License text: licenses/gpl-2.0.txt
3. GNU Lesser General Public License v2.1
   SPDX-License-Identifier: LGPL-2.1
   URL: http://spdx.org/licenses/LGPL-2.1.html#licenseText
   DPDK License text: licenses/lgpl-2.1.txt

1.3.5.7.10 Eigen

Mozilla Public License
Version 2.0

1. Definitions
1.1. "Contributor" means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. "Contributor Version" means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution" means Covered Software of a particular Contributor.

1.4. "Covered Software" means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. "Incompatible With Secondary Licenses" means
   a. that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or
   b. that the Covered Software was made available under the terms of version 1.1 or earlier of the License, but not also under the terms of a Secondary License.

1.6. "Executable Form"
means any form of the work other than Source Code Form.

1.7. “Larger Work”
means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. “License”
means this document.

1.9. “Licensable”
means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. “Modifications”
means any of the following:
   a. any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or
   b. any new file in Source Code Form that contains any Covered Software.

1.11. “Patent Claims” of a Contributor
means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. “Secondary License”
means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. “Source Code Form”
means the form of the work preferred for making modifications.

1.14. “You” (or “Your”)
means an individual or a legal entity exercising rights under this License. For legal entities, “You” includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, “control” means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions
2.1. Grants
Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:
   a. under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
   b. under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

(continues on next page)
2.2. Effective Date
The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope
The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

a. for any code that a Contributor has removed from Covered Software; or

b. for infringements caused by: (i) Your and any other third party’s modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

c. under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses
No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3).

2.5. Representation
Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use
This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions
Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities
3.1. Distribution of Source Form
All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form
If You distribute Covered Software in Executable Form then:

a. such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
b. You may distribute such Executable Form under the terms of this License, or
sublicense it under different terms, provided that the license for the Executable
Form does not attempt to limit or alter the recipients’ rights in the Source Code
under this License.

3.3. Distribution of a Larger Work
You may create and distribute a Larger Work under terms of Your choice, provided that
You also comply with the requirements of this License for the Covered Software. If
the Larger Work is a combination of Covered Software with a work governed by one or
more Secondary Licenses, and the Covered Software is not Incompatible With
Secondary Licenses, this License permits You to additionally distribute such
Covered Software under the terms of such Secondary License(s), so that the
recipient of the Larger Work may, at their option, further distribute the Covered
Software under the terms of either this License or such Secondary License(s).

3.4. Notices
You may not remove or alter the substance of any license notices (including copyright
notices, patent notices, disclaimers of warranty, or limitations of liability)
contained within the Source Code Form of the Covered Software, except that You may
alter any license notices to the extent required to remedy known factual
inaccuracies.

3.5. Application of Additional Terms
You may choose to offer, and to charge a fee for, warranty, support, indemnity or
liability obligations to one or more recipients of Covered Software. However, You
may do so only on Your own behalf, and not on behalf of any Contributor. You must
make it absolutely clear that any such warranty, support, indemnity, or liability
obligation is offered by You alone, and You hereby agree to indemnify every
Contributor for any liability incurred by such Contributor as a result of warranty,
support, indemnity or liability terms You offer. You may include additional
disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation
If it is impossible for You to comply with any of the terms of this License with
respect to some or all of the Covered Software due to statute, judicial order, or
regulation then You must: (a) comply with the terms of this License to the maximum
extent possible; and (b) describe the limitations and the code they affect. Such
description must be placed in a text file included with all distributions of the
Covered Software under this License. Except to the extent prohibited by statute or
regulation, such description must be sufficiently detailed for a recipient of
ordinary skill to be able to understand it.

5. Termination
5.1. The rights granted under this License will terminate automatically if You fail
to comply with any of its terms. However, if You become compliant, then the rights
granted under this License from a particular Contributor are reinstated (a)
provisionally, unless and until such Contributor explicitly and finally terminates
Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You
of the non-compliance by some reasonable means prior to 60 days after You have come
back into compliance. Moreover, Your grants from a particular Contributor are
reinstated on an ongoing basis if such Contributor notifies You of the non-
compliance by some reasonable means, this is the first time You have received
notice of non-compliance with this License from such Contributor, and You become
compliant prior to 30 days after Your receipt of the notice.
5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty
Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability
Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party’s negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation
Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party’s ability to bring cross-claims or counter-claims.

9. Miscellaneous
This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License
10.1. New Versions
Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.
10.2. Effect of New Versions
You may distribute the Covered Software under the terms of the version of the License you originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions
If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses
If you choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice
This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - “Incompatible With Secondary Licenses” Notice
This Source Code Form is “Incompatible With Secondary Licenses”, as defined by the Mozilla Public License, v. 2.0.

1.3.5.7.11 Fluent Helm Charts

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

“License” shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

“Licensor” shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

“Legal Entity” shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, “control” means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the

(continues on next page)
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made,
Aerial CUDA-Accelerated RAN, Release 24-2

use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NONINFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

1.3.5.7.12 Fmtlog

MIT License
Copyright (c) 2021 Meng Rao <raomeng1@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell (continues on next page)
copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.3.5.7.13 GDRCopy

Copyright (c) 2014-2021, NVIDIA CORPORATION. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.3.5.7.14 Google APIs

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

(continues on next page)
"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the...
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don’t include (continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

The brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.3.5.7.15 GoogleTest

Copyright 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
1.3.5.7.16 gRPC

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed with the Derivative Works; or
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
you make the Work available under this License (as defined below),
you promise and guarantee to the licensor and every other user to
these terms with every use and convey the right to do so. In terms
of these contributions, You promise a worldwide, royalty-free, and
non-exclusive, sub-licensable license to grant a royalty-free
license in your contributions to third-parties under the terms
and conditions specified in the License.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

-----------------------------------------------

BSD 3-Clause License

Copyright 2016, Google Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mozilla Public License, v. 2.0

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at https://mozilla.org/MPL/2.0/.

1.3.5.7.17 libuv

Copyright (c) 2015-present libuv project contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This license applies to parts of libuv originating from the https://github.com/joyent/libuv repository:

Copyright Joyent, Inc. and other Node contributors. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

====

This license applies to all parts of libuv that are not externally maintained libraries.

The externally maintained libraries used by libuv are:

- tree.h (from FreeBSD), copyright Niels Provos. Two clause BSD license.
- inet_ppton and inet_ntop implementations, contained in src/inet.c, are copyright the Internet Systems Consortium, Inc., and licensed under the ISC license.
- stdint-msvc2008.h (from msinttypes), copyright Alexander Chemeris. Three clause BSD license.
- pthread-fixes.c, copyright Google Inc. and Sony Mobile Communications AB. Three clause BSD license.

1.3.5.7.18 LibYAML

Copyright (c) 2017-2020 Ingy döt Net
Copyright (c) 2006-2016 Kirill Simonov

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.3.5.7.19 Libyang

Copyright (c) 2015-2021, CESNET
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of CESNET nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.3.5.7.20 Mimalloc

MIT License

Copyright (c) 2018-2021 Microsoft Corporation, Daan Leijen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

1.3.5.7.21 Prometheus Client Library for Modern C++

MIT License

Copyright (c) 2016-2019 Jupp Mueller
Copyright (c) 2017-2019 Gregor Jasny

And many contributors, see
https://github.com/jupp0r/prometheus-cpp/graphs/contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the “Software”), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

1.3.5.7.22 Protocol Buffers

Copyright 2008 Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following disclaimer
  in the documentation and/or other materials provided with the
  distribution.
* Neither the name of Google Inc. nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

(continues on next page)
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

1.3.5.7.23 protoc-gen-validate (PGV)

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or (continues on next page)
Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You 

(continues on next page)
meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions (continues on next page)
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
1.3.5.7.24 RE2

// Copyright (c) 2009 The RE2 Authors. All rights reserved.
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
// * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
// * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
/
/
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.3.5.7.25 UDPA API

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
This section describes the supported configurations, test-vector configurations, and limitations for this release of Aerial cuPHY.

**Important Terms**

This section defines common acronyms, abbreviations, and terms that are used in this Aerial cuBB documentation.

<table>
<thead>
<tr>
<th>Term or Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>Software suite that accelerates 5G RAN functions with the GPU</td>
</tr>
<tr>
<td>cuBB</td>
<td>CUDA GPU software libraries/tools that accelerate 5G RAN compute-intensive processing</td>
</tr>
<tr>
<td>cuPHY</td>
<td>CUDA 5G PHY layer software library of the cuBB</td>
</tr>
<tr>
<td>cuPHY-CP</td>
<td>cuPHY control-plane software</td>
</tr>
<tr>
<td>HDF5</td>
<td>A data file format used for storing test vectors. The HDF5 software library provides functions</td>
</tr>
<tr>
<td>CMake</td>
<td>CMake is a software tool for configuring the makefiles for building the CUDA examples</td>
</tr>
<tr>
<td>Term or Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
</tr>
<tr>
<td>DPDK</td>
<td>Data Plane Development Kit</td>
</tr>
<tr>
<td>CX6-DX</td>
<td>Mellanox ConnectX6-DX NIC</td>
</tr>
<tr>
<td>CDM/FDM/TDM</td>
<td>Code-division multiplexing, Frequency Division Multiplexing, Time-Division Multiplexing</td>
</tr>
<tr>
<td>MU-MIMO</td>
<td>Multi-User Multiple Input - Multiple Output</td>
</tr>
<tr>
<td>SU-MIMO</td>
<td>Single-User Multiple Input - Multiple Output</td>
</tr>
<tr>
<td>RB</td>
<td>Resource Block</td>
</tr>
<tr>
<td>PRB</td>
<td>Physical Resource Block</td>
</tr>
<tr>
<td>RE</td>
<td>Resource Element</td>
</tr>
<tr>
<td>REG</td>
<td>Resource Element Group</td>
</tr>
<tr>
<td>CORESET</td>
<td>Control Resource Set</td>
</tr>
<tr>
<td>DCI</td>
<td>Downlink Control Information</td>
</tr>
<tr>
<td>DMRS</td>
<td>Demodulation Reference Signal</td>
</tr>
<tr>
<td>eCPRI</td>
<td>Enhanced Common Public Radio Interface</td>
</tr>
<tr>
<td>MIB</td>
<td>Master Information Block</td>
</tr>
<tr>
<td>O-RAN</td>
<td>Open Radio Access Network</td>
</tr>
<tr>
<td>SIB/SIB1</td>
<td>System Information Block</td>
</tr>
<tr>
<td>TTI</td>
<td>Transmission Time Interval</td>
</tr>
<tr>
<td>LDPC</td>
<td>Low-Density Parity Check Code</td>
</tr>
<tr>
<td>PDCCH</td>
<td>Physical Downlink Control Channel</td>
</tr>
<tr>
<td>PDSCH</td>
<td>Physical Downlink Shared Channel</td>
</tr>
<tr>
<td>PUCCH</td>
<td>Physical Uplink Control Channel</td>
</tr>
<tr>
<td>PUSCH</td>
<td>Physical Uplink Shared Channel</td>
</tr>
<tr>
<td>PRACH</td>
<td>Physical Random Access Channel</td>
</tr>
<tr>
<td>UCI</td>
<td>Uplink Control Information</td>
</tr>
<tr>
<td>UE-EM</td>
<td>UE Emulator Test Equipment</td>
</tr>
</tbody>
</table>

1.4. Aerial cuPHY Developer Guide

1.4.1. cuPHY Software Architecture Overview

The cuPHY library software stack is shown in the figure below. It consists of L2 adapter, cuPHY driver, cuPHY CUDA kernels that process PHY channels and cuPHY controller.

The interface between the L2 and L1 goes through nvipc interface, which is provided as a separate
library. L2 and L1 communicate using FAPI protocol [6]. L2 adapter takes in slot commands from the L2 and translates them into L1 tasks, which are then consumed by cuPHY driver. Similarly, L1 task results are sent from cuPHY driver to L2 adapter, which are then communicated to L2.

The user transport block (TB) data in both DL and UL directions go through the same nvipc interface. The data exchange directly happens between cuPHY and L2 with the control of cuPHY driver.

cuPHY driver controls execution of cuPHY L1 kernels and manages the movement of data in and out of these kernels. The interface between the cuPHY L1 kernels and the NIC is also managed by the cuPHY driver by using the FH driver, that is provided as a library.

cuPHY controller is the main application that initializes the cell configurations, FH buffers and configures all threads that are used by L1 control tasks.

The functionality of each of these components is explained in more detail in the Components section.

![Fig. 1: cuPHY Software Stack](image)
1.4.2. Aerial cuPHY Components

1.4.2.1 L2 Adapter

The L2 Adapter is the interface between the L1 and the L2, which translates SCF FAPI commands to slot commands. The slot commands are received by cuPHY driver to initiate cuPHY tasks. It makes use of nvipc library to transport messages and data between L1 and L2. It is also responsible for sending slot indications to drive the timing of the L1-L2 interface. L2 Adapter keeps track of the slot timing and it can drop messages received from L2 if they are received late.

1.4.2.2 cuPHY Driver

The cuPHY driver is responsible for orchestrating the work on the GPU and the FH by using cuPHY and FH libraries. It processes L2 slot commands generated by L2 adapter to launch tasks and communicates cuPHY outputs (e.g. CRC indication, UCI indication, measurement reports, etc.) back to L2. It uses L2 adapter FAPI message handler library to communicate with L2.

cuPHY driver configures and initiates DL and UL cuPHY tasks, which in turn launch CUDA kernels on the GPU. These processes are managed at the slot level. The cuPHY driver also controls CUDA kernels responsible for transmission and reception of user plane (U-plane) packets to and from the NIC interface. The CUDA kernels launched by the driver take care of re-ordering and decompression of UL packets and compression of DL packets. The DL packets are transmitted by GPU initiated communications after the compression.

cuPHY driver interacts with the FH interface using ORAN compliant FH library to coordinate transmission of FH control plane (C-plane) packets. The transmission of C-plane packets is done via DPDK library calls (CPU initiated communication). The U-plane packets are communicated through transmit and receive queues created by the cuphycontroller.

1.4.2.3 FH Driver Library

The FH library ensures timely transmission and reception of FH packets between the O-DU and O-RU. It uses accurate send scheduling functions of the NIC to comply with the timing requirements of the O-RAN FH specification.

The FH driver maintains the context and connection per eAxCid. It is responsible of encoding and decoding of FH commands for U-plane and C-plane messages.

The FAPI commands received from the L2 trigger processing of DL or UL slots. C-plane messages are for both DL and UL generated on the CPU and communicated to the O-RU through the NIC interface with DPDK. The payload of DL U-plane packets are prepared on the GPU and sent to the NIC interface from the memory pool on the GPU with the DOCA GPU NetIO library. The flow of DL C-plane and U-plane packets is illustrated in the below figure.

As shown in the above figure, UL U-plane packets received from the O-RU are directly copied to GPU memory from the NIC interface with the DOCA GPU NetIO library. The UL data is decompressed and processed by GPU kernels. After the UL kernels are completed, the decoded UL data transport blocks are sent to the L2.
Fig. 2: User and Control Plane Data Flow through cuPHY driver and cuPHY tasks
Fig. 3: Flow of packets on the FH
1.4.2.4 cuPHY Controller

The cuPHY controller is the main application that initializes the system with the desired configuration. During the start-up process, cuPHY controller creates a new context (memory resources, tasks) for each new connection with an O-RU, identified by MAC address, VLAN ID and set of eAxCids. It starts cuphydriver DL/UL worker threads and assigns them to CPU cores as configured in the yaml file. It also prepares GPU resources and initiates FH driver and NIC class objects.

cuPHY controller prepares L1 according to the desired gNB configuration. It can also bring a carrier in and out of service with the cell lifecycle management functionality.

1.4.2.5 cuPHY

cuPHY is a CUDA implementation of 5G PHY layer signal processing functions. The cuPHY library supports all 5G NR PHY channels in compliance with 3GPP Release 15 specification. As shown in the below figure, cuPHY library corresponds to upper PHY stack according to O-RAN 7.2x split option [8].

![cuPHY library within 5G NR software stack](image)

Fig. 4: cuPHY library within 5G NR software stack

cuPHY is optimized to take advantage of the massive parallel processing capability of the GPU architecture by running the workloads in parallel when possible. cuPHY driver orchestrates signal processing tasks running on the GPU. These tasks are organized according to the PHY layer channel type, e.g. PDSCH, PUSCH, SSB, etc. A task related to a given channel is termed as pipeline. For example, PDSCH channel is processed in PDSCH pipeline and the PUSCH channel is processed in PUSCH pipeline. Each pipeline includes a series of functions related to the specific pipeline and consists of multiple CUDA kernels. Each pipeline is capable of running signal processing workloads for multiple cells. The pipelines are dynamically managed for each slot by cuPHY driver with channel aggregate objects. The group of cuPHY channel pipelines that is executed in a given time slot depends on what is scheduled by the L2 in that time slot.

The cuPHY library exposes a set of APIs per PHY channel to create, destroy, setup, configure and run each pipeline as shown in the following figure. L2 adapter translates SCF FAPI messages and other system configurations and cuPHY driver invokes associated cuPHY APIs for each slot. The API’s shown as grey such as (Re)-Config, StateUpdate are not currently supported.

The following are descriptions of the APIs in the above figure:

---

Chapter 1. Aerial cuBB
1.4.2.5.1 PDSCH Pipeline

The PDSCH pipeline receives configuration parameters for each cell and the UE and the corresponding DL transport blocks (TBs). After completing the encoding of the PDSCH channel, the pipeline outputs IQ samples mapped to the resource elements (REs) allocated to the PDSCH. The PDSCH pipeline consists of multiple CUDA kernels, which are launched with CUDA graph functionality to reduce the kernel launch overhead. The diagram of the CUDA graph used by PDSCH pipeline is shown in the following figure. The green boxes represent CUDA kernels and the orange boxes represent input and output buffers.

The PDSCH pipeline contains the following components:

- CRC calculation of the TBs and code-blocks (CBs)
- LDPC encoding
- Fused Rate Matching and Modulation Mapper
- DMRS generation

The CRC calculation component performs the code block segmentation and the CRC calculation. The CRC is calculated first for each TB and then for each CB. The fused rate matching and modulation com-
ponent performs rate-matching, scrambling, layer-mapping, pre-coding and modulation. This compo-
nent is also aware of which resource elements it should skip if CSI-RS is configured.

The PDSCH pipeline involves the following kernels:
- prepare_crc_buffers
- crcDownlinkPdschTransportBlockKernel
- crcDownlinkPdschCodeBlocksKernel
- ldpc_encode_in_bit_kernel
- fused_dl_rm_and_modulation
- fused_dmrs

Kernels exercised only if CSI-RS parameters are present are as follows:
- zero_memset_kernel
- genCsirsReMap
- postProcessCsirsReMap

The cuPHY PDSCH transmit pipeline populates parts of a 3D tensor buffer of I/Q samples in GPU memory, where each sample is a complex number using fp16, i.e. each sample is a __half2 using x for the real part and y for the imaginary part. The output 3D tensor buffer is allocated by the cuPHY driver when the application is first launched and it is reset for every slot (i.e., between successive PDSCH launches) by the cuPHY driver. Here, re-setting the buffer means, it is initialized to all zero values.

The output tensor contains 14 symbols on time domain (x-axis), 273 PRBs (Physical Resource Blocks) on frequency domain (y-axis), and up to 16 layers on spatial domain (z-axis). For the y-axis, each PRB contains 12 REs, and each RE is a __half2 data. Contiguous PRBs for the same OFDM symbol and spatial layer are allocated next to each other on memory. The resources are mapped in memory in the following order: frequency domain, time domain and then the spatial domain (or layer domain). This is the maximum size of the output buffer needed for a cell per slot.

The PDSCH only fills in parts of that buffer, i.e., its allocated PRBs, based on various configuration parameters it receives that vary over time. Parts of the slot can be filled by other down-link control channels. From a PDSCH standpoint, only the two fused_* kernels listed above, fused_dl_rm_and_modulation and fused_dmrs write to the output buffer. The fused rate-matching and modulation kernel writes data part of the I/Q samples, while the DMRS kernel only writes the DMRS...
symbols, i.e., only 1 or 2 contiguous symbols in the x-dimension. Note that, unlike other components, DMRS is not dependent on any of the previous pipeline stages.

The PDSCH pipeline expects pre-populated structs cuphyPdschStatPrms_t (cuPHY PDSCH static parameters) and cuphyPdschDynPrms_t (cuPHY PDSCH dynamic parameters) that include the input data and the necessary configuration parameters.

The TB data input can exist either in CPU or GPU memory depending on the cuphyPdschDataIn_t pBufferType. If this is GPU_BUFFER, then the host to device (H2D) memory copies for that data can happen before PDSCH setup is executed for each cell. This is called prepone H2D copy and it can be configured by setting the prepone_h2d_copy flag in the l2_adapter_config_.yaml file. If prepone H2D copy is not enabled, the copy operations happen as part of PDSCH setup. It is highly recommended that the prepone H2D copy should be enabled to achieve high capacity in a multiple cell scenario.

The way LDPC kernels are initiated can change when multiple TBs are configured on PDSCH. If the LDPC configuration parameters are identical across TBs, PDSCH launches a single LDPC kernel for all TBs (as it is the case for the other PDSCH components). If the LDPC configuration parameters vary across the TBs, then multiple LDPC kernels are launched, one for each unique configuration parameters set. Each LDPC kernel is launched on a separate CUDA stream.

The PDSCH CUDA graph contains only kernel nodes and has the layout shown in the PDSCH graph diagram shown above. As it is not possible to dynamically change the graph geometry at runtime, PDSCH_MAX_HET_LDPC_CONFIGS_SUPPORTED potential LDPC kernel nodes are created. Depending on the LDPC configuration parameters and the number of TBs, only a subset of these kernels perform LDPC encoding. The remaining nodes are disabled at runtime if needed per PDSCH. The DMRS kernel node is not dependent on any of the other PDSCH kernels. Therefore, it can be placed anywhere in the graph. The three kernels preceding the DMRS in the graph are only exercised if CSI-RS parameters are present (or CSI-RS is configured). These kernels compute information needed by the fused rate matching and modulation kernel about the REs that need to be skipped.

1.4.2.5.2 PDCCH Pipeline

The cuPHY PDCCH channel processing involves the following kernels:

- encodeRateMatchMultipleDCIsKernel
- genScramblingSeqKernel
- genPdcchTfSignalKernel

When running in graphs mode, the CUDA graph launched on every slot contains only kernel nodes and its current layout is as depicted in the below figure.

![Fig. 7: cuPHY PDCCH graph layout](image)

PDCCH kernel takes static and dynamic parameters as in PDSCH.

Notes on PDCCH configuration and dataset conventions:

- The PdcchParams dataset contains the coreset parameters for a given cell. Dataset DciParams_coreset_0_dci_0 contains the DCI parameters for the first DCI of coreset 0.
There is a separate dataset for every DCI in a cell with the naming convention: DciParams_coreset_<i>_dci_<j>, where i has values from 0 up to (number of coresets – 1), while j starts from 0 for every coreset i and goes up to (PdcchParams[i].numDlDci – 1) for that coreset.

- Dataset DciPayload_coreset_0_dci_0 contains the DCI payload, in bytes, for the first DCI of coreset 0. It follows the naming convention mentioned above DciParams_coreset_0_dci_0.
- Dataset(s) DciPmW_coreset_i_dci_j hold the precoding matrix for a given DCI, coreset pair, if it has precoding enabled.
- X_tf_fp16 is the 3D output tensor for that cell and is used for reference checks in the various PDCCH examples.
- X_tf_cSamples_bfp* datasets that contain compressed data are not used in cuPHY, since compression happens in cuphydriver after all cuPHY processing for all downlink channels scheduled in a slot has completed.

1.4.2.5.3 SSB Pipeline

The cuPHY SS Block channel processing involves the following kernels:

- encodeRateMatchMultipleSSBsKernel
- ssbModTFSigKernel

When running in graphs mode, the CUDA graph launched on every slot contains only these two kernel nodes connected in sequence.

Notes on SSB configuration and dataset conventions:

- The SSTxParams dataset contains all the nSsb, SSB parameters for a given cell.
- SSB bursts cannot be multiplexed in frequency domain, they can only be multiplexed in time domain.
- nSsb datasets contains the number of SSBs in a cell, this is also the size of the SSTxParams dataset.
- x_mib contains the Master Information Block (MIB) for each SSB in the cell as an uint32_t element; only the least significant 24-bits of each element are valid.
- Dataset(s) Ssb_PM_W* contain the precoding matrices if precoding is enabled for a given SSB.
- X_tf_fp16 is the 3D output tensor for that cell and is used for reference checks in the various SSB examples. Every I/Q sample there is stored as __half2c.
  X_tf is similar to X_tf_fp16 but every I/Q sample there is stored as float2 instead of __half2; not currently used in cuPHY.
- X_tf_cSamples_bfp* datasets hold the output compressed and are not used in cuPHY as compression is applied as part of the cuphydriver.
1.4.2.5.4 CSI-RS Pipeline

The cuPHY CSI-RS channel processing involves the following kernels:

▶ genScramblingKernel
▶ genCsirsTfSignalKernel

When running in graphs mode, the CUDA graph launched on every slot contains only these two kernel nodes connected in sequence.

Notes on CSI-RS configuration and dataset conventions:

▶ CsirsParamsList contains configuration parameters which are used for non-zero power signal generation (e.g., NZP, TRS).
▶ Please note that CsirsParamsList dataset can have multiple elements. All elements in the dataset can be processed with single setup/run call.
▶ X_tf_fp16 is the 3D reference output tensor for that cell and is used for reference checks in the various CSI-RS examples. Every I/Q sample there is stored as __half2c.
▶ X_tf is similar to X_tf_fp16 but every I/Q sample there is stored as float2 instead of __half2; not currently used in cuPHY.
▶ X_tf_cSamples_bfp* datasets hold the output compressed and are not used in cuPHY as compression is applied as part of cuphydriver.
▶ X_tf_remap is reference output for RE Map, this is not used currently as current implementation only generates NZP signal.
▶ Dataset(s) Csirs_PM_W* contain precoding matrices and are used if precoding is enabled.

1.4.2.5.5 PUSCH Pipeline

The PUSCH pipeline includes the following components (which are illustrated in the PUSCH Pipeline Front End and PUSCH and CSI Part 1 Decoding figures):

▶ Least squares (LS) channel estimation
▶ Minimum Mean Square Error (MMSE) channel estimation
▶ Noise and interference covariance estimation
▶ Shrinkage and whitening
▶ Channel Equalization
▶ Carrier frequency offset (CFO) estimation and CFO averaging
▶ Timing offset (TO) estimation and averaging.
▶ Received signal strength indicator (RSSI) estimation and averaging
▶ Noise variance estimation
▶ Received signal received power (RSRP) estimation and averaging
▶ SNR estimation
▶ De-rate matching
▶ LDPC backend
If CSI part 2 is configured, the following components are also used (these components are illustrated in the *PUSCH and CSI Part 1 Decoding* and *PUSCH and CSI Part 2 Decoding* figures):

- Simplex decoder or RM decoder or Polar decoder (for CSI decoding of CSI part 1 depending on the UCI payload size)
- CSI part 2 de-scrambling and de-rate matching
- Simplex decoder or RM decoder or Polar decoder (for CSI decoding of CSI part 2 depending on the UCI payload size)

The PUSCH pipeline receives IQ samples, which are provided by order and decompression kernels. The received IQ data is stored in the address `cuphyPuschDataIn_t PhyPuschAggr::DataIn.pTDataRx` as the `cuphyTensorPrm_t` type. The IQ samples are represented by half precision (16-bits) real and imaginary values. The size of the input buffer is multiplication of number of maximum PRBs (273), number of subcarriers per PRB (12), number of OFDM symbols per slot (14) and number of maximum antenna ports per cell (16). This buffer is created for each cell.
1.4.2.5.5.1 Channel Estimation

<table>
<thead>
<tr>
<th>First Stage (LS CE)</th>
<th>PhyPuschAggr::DataIn.pTDataRx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Buffer</td>
<td>CUPHY_C_16_F: tensor vector of IQ samples</td>
</tr>
<tr>
<td>Data type</td>
<td>[(ORAN_MAX_PRB<em>CUPHY_N_TONES_PER_PRB), OFDM_SYMBOLS_PER_SLOT, MAX_AP_PER_SLOT]: [(273</em>12),14,16]</td>
</tr>
<tr>
<td>Description</td>
<td>IQ samples of the input data received from the FH for an UL slot. The I/Q data are represented in half precision float.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx::m_tRefDmrsLSEstVec[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>the index i refers to a PRB range (or UE group)</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_C_32_F: float complex IQ samples</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(CUPHY_N_TONES_PER_PRB*(number_of PRBs)/2), NUM_LAYERS, NUM_ANTENNAS, NH]: [(12*(number of PRBs)/2), (number of layers), (number of RX antennas), (number of DMRS symbols)]</td>
</tr>
<tr>
<td>Description</td>
<td>IQ samples of the initial channel estimates on DMRS symbols. The I/Q data are represented in half precision float.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx::m_tRefDmrsAccumVec[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>the index i refers to a PRB range (or UE group)</td>
</tr>
<tr>
<td>Data Type</td>
<td>CUPHY_C_32_F: float complex IQ samples</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1,2]: Two dimensions for one active and one non-active buffer</td>
</tr>
<tr>
<td>Description</td>
<td>Holds summation of conj(H_ls[k])*H_ls[k+1] in a given PRB range, which is then used to calculate mean delay in the next stage. The index k refers to the subcarrier index in a given PRB range. conj() represents the conjugation function.</td>
</tr>
</tbody>
</table>

Channel estimation (CE) consists of two stages: least-squares (LS) CE and minimum-mean-square (MMSE) CE.

In the LS CE stage, DMRS symbols are used to obtain initial channel estimate on DMRS REs and to calculate mean delay of the channel impulse response (CIR). The mean delay and the initial estimates are then used to obtain channel estimates in data REs on the second stage with MMSE filtering operation.

The second stage invokes a dispatch kernel chEstFilterNoDftSOfdmDispatchKernel() to support different configurations. The dispatch kernel first calculates mean channel delay by using the stored value tInfoDmrsAccum from the first stage. It then chooses an appropriate kernel depending on number of PRBs in the given PUSCH allocation and number of consecutive DMRS symbols (drvdUe-GrpPrms.dmrsMaxLen). The MMSE filtering operation is done by the kernel windowedChEstFilterNoDftSOfdmKernel().
<table>
<thead>
<tr>
<th>Second Stage (MMSE CE)</th>
<th>Input Buffer</th>
<th>Receives outputs of the second stage as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input Buffer</td>
<td>statDescr.tPr  mFreqInterpCoefsSmall, statDescr.tPrmFreqInterpCoefs, or statDescr.tPrmFreqInterpCoefs4</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Interpolation filter coefficients depending on the number of PRBs</td>
</tr>
<tr>
<td></td>
<td>Output Buffer</td>
<td>PuschRx::m_tRefHEstVec[i]</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td>the index i refers to a PRB range (or UE group)</td>
</tr>
<tr>
<td></td>
<td>Data type</td>
<td>CUPHY_C_32_F: float complex IQ samples</td>
</tr>
<tr>
<td></td>
<td>Dimensions</td>
<td>[NUM_ANTENNAS, NUM_LAYERS, NF, NH]: [(number of RX antennas), (number of layers), (12*(number of PRBs)), (number of DMRS symbols)]</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Estimates of the received channel on the DMRS symbols.</td>
</tr>
</tbody>
</table>

### 1.4.2.5.5.2 Noise and Interference Covariance Estimation

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>Receives outputs of channel estimation kernel as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_tRefNoiseVarPreEq</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F: float real values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, NUM_UUE_GROUPS]</td>
</tr>
<tr>
<td>Description</td>
<td>Estimates of the noise variance pre-equalization per UE group (or PRB range).</td>
</tr>
</tbody>
</table>

| Output Buffer | PuschRx:: m_tRefLwInvVec[i] |
| Note: the index i refers to a PRB range (or UE group) |
| Data Type     | CUPHY_C_32_F: float complex IQ samples |
| Dimensions    | [NUM_ANTENNAS, NUM_ANTENNAS, numPRB]: [(number of RX antennas), (number of RX antennas),(number of PRBs)] |
| Description   | Inverse Cholesky factor of noise-interference tensor information. |
### 1.4.2.5.5.3 Carrier Frequency and Timing Offset Estimation

<table>
<thead>
<tr>
<th>Input Buffers</th>
<th>PuschRx::m_tRefHEstVec[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This buffer is received from Channel Estimation kernel.</td>
</tr>
<tr>
<td></td>
<td>Note: the index i refers to a PRB range (or UE group).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_tRefCfoEstVec[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: the index i refers to a PRB range (or UE group)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>CUPHY_R_32_F: float real values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>[MAX_ND_SUPPORTED, (number of UEs)]: [14, (number of UEs)]</td>
</tr>
<tr>
<td>Description</td>
<td>CFO estimate vector.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_tRefCfoHz</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>CUPHY_R_32_F: float real values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>[1, (number of UEs)]</td>
</tr>
<tr>
<td>Description</td>
<td>CFO estimate values in Hz.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_tRefTaEst</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>CUPHY_R_32_F: float real values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>[1, (number of UEs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Timing offset estimates.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_tRefCfoPhaseRot</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>CUPHY_C_32_F: float complex values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>[CUPHY_PUSCH_RX_MAX_N_TIME_CH_EST, CUPHY_PUSCH_RX_MAX_N_LAYERS_PER_UE_GROUP, MAX_N_USER_GROUPS_SUPPORTED] : [(max number of channel estimates in time, =4), (max layers per UE group, =8), (max UE groups, =128)]</td>
</tr>
<tr>
<td>Description</td>
<td>Carrier offset phase rotation values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_tRefTaPhaseRot</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>CUPHY_C_32_F: float complex values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>[1, CUPHY_PUSCH_RX_MAX_N_LAYERS_PER_UE_GROUP] : [1, (max layers per UE group, =8)]</td>
</tr>
<tr>
<td>Description</td>
<td>Carrier offset phase rotation values</td>
</tr>
</tbody>
</table>
1.4.2.5.4 Soft De-mapper

<table>
<thead>
<tr>
<th><strong>Channel Equalization Coefficients Computation Kernel</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Buffers</strong></td>
</tr>
<tr>
<td>PuschRx::m_tRefHEstVec[i], PuschRx:: m_tRefLwInvVec[i], PuschRx:: m_tRefCfoEstVec[i]</td>
</tr>
<tr>
<td>These buffers are received from Noise and Interference Co-variance Estimation, Channel Estimation and CFO Estimation kernels.</td>
</tr>
<tr>
<td><em>Note:</em> the index i refers to a PRB range (or UE group).</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
</tr>
<tr>
<td>PuschRx:: m_tRefReeDiagInvVec[i]</td>
</tr>
<tr>
<td><em>Note:</em> the index i refers to a PRB range (or UE group)</td>
</tr>
<tr>
<td><strong>Data Type</strong></td>
</tr>
<tr>
<td>CUPHY_R_32_F: float real values</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
</tr>
<tr>
<td>[CUPHY_N_TONES_PER_PRB, NUM_LAYERS, NUM_PRBS, nTimeChEq]</td>
</tr>
<tr>
<td>[12*(number of PRBs), (number of layers), (number of PRBs), (number of time domain estimates)]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Channel equalizer residual error vector.</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
</tr>
<tr>
<td>PuschRx:: m_tRefCoefVec[i]</td>
</tr>
<tr>
<td><em>Note:</em> the index i refers to a PRB range (or UE group)</td>
</tr>
<tr>
<td><strong>Data Type</strong></td>
</tr>
<tr>
<td>CUPHY_C_32_F: float complex IQ samples</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
</tr>
<tr>
<td>[NUM_ANTENNAS, CUPHY_N_TONES_PER_PRB, NUM_LAYERS, NUM_PRBS, NH]</td>
</tr>
<tr>
<td>[(number of RX antennas), 12*(number of PRBs), (number of layers), (number of PRBs), (number of DMRS positions)]</td>
</tr>
<tr>
<td><strong>Descriptions</strong></td>
</tr>
<tr>
<td>Channel equalizer coefficients.</td>
</tr>
<tr>
<td>Channel Equalization MMSE Soft De-mapping Kernel</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Input Buffers</strong></td>
</tr>
<tr>
<td>PuschRx:: m_tRefCoefVec[i], PuschRx:: m_tRefCfoEstVec[i],</td>
</tr>
<tr>
<td>PuschRx:: m_tRefReeDiagInvVec[i]</td>
</tr>
<tr>
<td>PuschRx:: m_drvdUeGrpPrmsCpu[i].tInfoDataRx</td>
</tr>
<tr>
<td>These buffers are received from Noise and Interference Covariance Estimation, Channel Estimation and CFO Estimation kernels.</td>
</tr>
<tr>
<td><em>Note</em>: the index i refers to a PRB range (or UE group).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Output Buffer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PuschRx:: m_tRefDataEqVec[i]</td>
</tr>
<tr>
<td><em>Note</em>: the index i refers to a PRB range (or UE group)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CUPHY C 16 F : tensor vector of half float IQ samples.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dimensions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>[NUM_LAYERS, NF, NUM_DATA_SYMS]</td>
</tr>
<tr>
<td>([number of layers], 12*(number of PRBs), (number of data OFDM symbols])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Equalized QAM data symbols.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Output Buffer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PuschRx:: m_tRefLLRVec[i]</td>
</tr>
<tr>
<td><em>Note</em>: the index i refers to a PRB range (or UE group)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CUPHY R 16 F : tensor vector of half float real samples.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dimensions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>[CUPHY_QAM_256, NUM_LAYERS, NF, NUM_DATA_SYMBOLS]</td>
</tr>
<tr>
<td>([number of bits for 256QAM = 8], (number of layers), (number of layers), 12*(number of PRBs), (number of data OFDM symbols])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Descriptions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LLRs or softbits. Used if UCI on PUSCH is enabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Output Buffer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PuschRx:: m_tRefLLRCdm1Vec[i]</td>
</tr>
<tr>
<td><em>Note</em>: the refers to a PRB range (or UE group)index i</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CUPHY R 16 F : tensor vector of half float real samples.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dimensions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>[CUPHY_QAM_256, NUM_LAYERS, NF, NUM_DATA_SYMBOLS]</td>
</tr>
<tr>
<td>([number of bits for 256QAM = 8], (number of layers), (number of layers), 12*(number of PRBs), (number of data OFDM symbols])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Descriptions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Output LLRs or softbits. Used if there is no UCI on PUSCH.</td>
</tr>
</tbody>
</table>
1.4.2.5.5.5 De-rate matching and Descrambling

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PuschRx:: m_tRefLLRVec[i] or PuschRx:: m_tRefLLRCdm1Vec[i], PuschRx:: m_pTbPrmsGpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_pHarqBuffers</td>
</tr>
<tr>
<td>Data type</td>
<td>uint8_t</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Function of TB size and number of TBs.</td>
</tr>
<tr>
<td>Description</td>
<td>Rate-matching/descrambling output. It is on a host pinned GPU memory. It is mapped to PhyPuschAggr::DataInOut.pHarqBuffersInOut</td>
</tr>
</tbody>
</table>

1.4.2.5.5.6 RSSI Estimation

The RSSI is calculated from the received signal by first calculating the received signal power on each RE and each receive antenna. The total power is then calculated by summation of powers across the frequency resources and receive antennas and average over OFDM symbols in accordance to the SCF FAPI specification.

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PuschRx:: m_drvdUeGrpPrmsCpu[i].tInfoDataRx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_tRefRssiFull</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float real samples.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[MAX_ND_SUPPORTED, MAX_N_ANTENNAS_SUPPORTED, nUEgroups]:</td>
</tr>
<tr>
<td></td>
<td>[(max number of time domain estimates, =14), (max number of antennas, =64), (number of UE groups)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured RSSI (per symbol, per antenna, per UE group).</td>
</tr>
</tbody>
</table>

| Output Buffer | PuschRx:: m_tRefRssi |
| Data type     | CUPHY_R_32_F : tensor vector of float real samples. |
| Dimensions    | [1, nUEgroups]:[1, (number of UE groups)] |
| Description   | Measured RSSI per UE group. |
### 1.4.2.5.5.7 RSRP and SINR Estimation

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PuschRx::m_tRefHEstVec[i], PuschRx::m_tRefReeDiagInvVec[i], PuschRx::m_tRefNoiseVarPreEq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_tRefRsrp</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float real samples.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, nUEgroups];[1, (number of UE groups)]</td>
</tr>
<tr>
<td>Description</td>
<td>RSRP values across UEs.</td>
</tr>
</tbody>
</table>

| Output Buffer         | PuschRx:: m_tRefNoiseVarPostEq                                                                 |
| Data type             | CUPHY_R_32_F : tensor vector of float real samples.                                             |
| Dimensions            | [1, nUEgroups];[1, (number of UE groups)]                                                       |
| Description           | Post-equalization noise variances across UEs                                                    |

| Output Buffer         | PuschRx:: m_tRefSinrPreEq                                                                      |
| Data type             | CUPHY_R_32_F : tensor vector of float real samples.                                             |
| Dimensions            | [1, nUEgroups];[1, (number of UE groups)]                                                       |
| Description           | Pre-equalization SINR values across UEs.                                                        |

| Output Buffer         | PuschRx:: m_tRefSinrPostEq                                                                     |
| Data type             | CUPHY_R_32_F : tensor vector of float real samples.                                             |
| Dimensions            | [1, nUEgroups];[1, (number of UE groups)]                                                       |
| Description           | Post-equalization SINR values across UEs.                                                       |

### 1.4.2.5.5.8 UCI on PUSCH Decoder

If UCI is configured on PUSCH channel, output of the soft-demapper first goes through de-segmentation to separate HARQ, CSI part 1 and CSI part 2 and SCH softbits (or LLRs). This initial step is done by the kernel uciOnPuschSegLLRs0Kernel().

If CSI-part2 is present, CSI-part2 control kernel is launched as shown in the figure below as a dashed box. This kernel determines the number of CSI-part2 bits and rate-matched bits and selects the correct decoder kernels and initiates their setup functions.

De-segmentation of CSI-part2 payload is done by uciOnPuschSegLLRs2Kernel() kernel, which separates CSI-part2 UCI and SCH softbits.
<table>
<thead>
<tr>
<th><strong>UCI on PUSCH De-segmentation of First Phase</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Buffer</strong></td>
<td>PuschRx:: m_tPrmLLRVec[i]</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
<td>PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[i].d_harqLLRs;</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>__half*</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>HARQ soft bits.</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
<td>PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[ueIdx].d_csi1LLRs;</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>__half*</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>CSI part 1 soft bits.</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
<td>PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[i].d_schAndCsi2LLRs</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>__half*</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Shared channel (SCH) and CSI part 2 soft bits.</td>
</tr>
</tbody>
</table>
Fig. 9: Graph Diagram of the PUSCH and CSI Part 1 Decoding

<table>
<thead>
<tr>
<th><strong>UCI on PUSCH De-segmentation of Second Phase</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Buffer</strong></td>
<td>PuschRx::m_tPrmLLRVec[i]</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
<td>PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[i].d_schAndCsi2LLRs;</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>__half*</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Pointer to SCH softbits</td>
</tr>
<tr>
<td><strong>Output Buffer</strong></td>
<td>PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[i].d_schAndCsi2LLRs + PuschRx::m_pTbPrmsGpu-&gt;pUePrmsGpu[i].G;</td>
</tr>
<tr>
<td><strong>Data type</strong></td>
<td>__half*</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Pointer to CSI part2 softbits</td>
</tr>
</tbody>
</table>
1.4.2.5.5.9 Simplex Decoder

The simplex decoder implements maximum likelihood (ML) decoder. It receives input LLRs and outputs estimated codewords. It also reports HARQ DTX status.

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_LLRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>__half*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td>Description</td>
<td>Pointer to input LLRs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_cbEst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Uint32_t*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td>Description</td>
<td>Decoded UCI payload.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_DTXStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Uint8_t*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Parameter.</td>
</tr>
<tr>
<td>Description</td>
<td>Pointer to HARQ detection status.</td>
</tr>
</tbody>
</table>
1.4.2.5.5.10 Reed Muller (RM) Decoder

The RM decoder implements maximum likelihood (ML) decoder. It receives input LLRs and outputs estimated codewords. It also reports HARQ DTX status.

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_LLRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>__half*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td>Description</td>
<td>Pointer to input LLRs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_cbEst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>uint32_t*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Single dimensional array, the size depending on the payload.</td>
</tr>
<tr>
<td>Description</td>
<td>Decoded UCI payload.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_DTXStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Uint8_t*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Parameter.</td>
</tr>
<tr>
<td>Description</td>
<td>Pointer to HARQ detection status.</td>
</tr>
</tbody>
</table>

1.4.2.5.5.11 Polar Decoder

Polar decoder uses CRC aided list decoder with tree pruning. There are many variants of the decoding algorithm that is used in decoding of Polar codes. Please see [2, 3] for some of the related work. The exact implementation in cuPHY is optimized for the GPU architecture.

The tree-pruning algorithms combine leaf nodes together, which is a better data structure for execute decoding in parallel. Hence it is more suitable for GPU architecture. There are different methods of forming leaf nodes in the tree pruning algorithm. In our implementation we use rate-0 and rate-1 leaf codewords. In rate-0 leaf nodes, multiple bits are always frozen and are zero, whereas there are no frozen bits in rate-1 leaf nodes. In rate-1 codewords, LLRs can be decoded in parallel.

Tree pruning is done by compCwTreeTypesKernel() before the input LLRs are received by the Polar Decoder kernel.

If the list size is equal to 1, polarDecoderKernel(), if the list size is greater than 1, listPolarDecoderKernel() is run.
Input Buffer | PuschRx:: m_cwTreeLLRsAddrVec  
---|---  
Data type | __half*  
Dimensions | Single dimensional array, the size depending on the payload.  
Description | Pointer to codeword tree of LLR addresses.  
Output Buffer | PuschRx:: m_cbEstAddrVec  
---|---  
Data type | uint32_t*  
Dimensions | Single dimensional array, the size depending on the payload.  
Description | Pointer to estimated CB addresses.  

### 1.4.2.5.5.12 LDPC Decoder

LDPC decoder is implemented with normalized layered min-sum algorithm [1] and it uses short float (FP16) data type as log-likelihood ratio (LLR) metrics.

| Input Buffer | PuschRx:: m_LDPCDecodeDescSet.llr_input[m_LDPCDecodeDescSet .num_tbs]  
---|---  
The first address is also mapped to PuschRx::m_pHarqBuffers[ueIdx]  
---|---  
Data type | cuphyTransportBlockLLRDesc_t  
Dimensions | Single dimensional array, the size depending on the number of valid TB descriptors. The max size is 32.  
Description | Input LLR buffers.  
Output Buffer | PuschRx:: m_LDPCDecodeDescSet.tb_output[m_LDPCDecodeDescSet .num_tbs]  
---|---  
The first address is also mapped to PuschRx::d_LDPCOut + offset  
Offset is a function of UE index and number of codewords per UE.  
---|---  
Data type | cuphyTransportBlockDataDesc_t  
Dimensions | Single dimensional array, the size depending on the number of valid TB descriptors.  
Description | Pointer to estimated TB addresses.  

1.4.2.5.13 CRC Decoder

<table>
<thead>
<tr>
<th>Code Block CRC Decoder Kernel</th>
<th>Input Buffer</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PuschRx::d_pLDPCOut, PuschRx:: m_pTbPrmsGpu</td>
<td>LDPC decoder output and TB parameters needed to decode the CRC.</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_outputPrms.pCbCrcsDevice;</td>
<td></td>
</tr>
<tr>
<td>Data type</td>
<td>uint32_t</td>
<td>CRC output.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, total number of CBs (across UEs)]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transport Block CRC Decoder Kernel</th>
<th>Input Buffer</th>
<th>Data Type</th>
<th>Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PuschRx:: m_outputPrms.pTbPayloadsDevice, PuschRx:: m_pTbPrmsGpu</td>
<td>Uint8_t</td>
<td>[1, total number of TB payload bytes]</td>
<td>TB CRC output.</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PuschRx:: m_outputPrms.pTbCrcsDevice</td>
<td>uint32_t</td>
<td>[1, total number of TBs (across UEs)]</td>
<td></td>
</tr>
</tbody>
</table>

1.4.2.5.6 PUCCH Pipeline

The PUCCH pipeline can be divided into logical stages. The first, front-end processing, is unique for each PUCCH format and involves descrambling and demodulation to recover transmitted symbols. For formats 0 and 1, this is the only stage performed as there is no decoding necessary to recover data. For formats 2 and 3, this is followed by decoding. Here, the kernels used are the same as those in PUSCH for the same decoding type. Finally, the decoded data is segmented into HARQ, SR and CSI payloads.

The kernels responsible for front-end processing are as follows:

- pucchF0RxKernel
- pucchF1RxKernel
- pucchF2RxKernel
pucchF3RxKernel

with each corresponding to formats 0 through 3 respectively. For formats 0 and 1, hard decisions are made as part of demodulation to recover 1 or 2 payload bits, depending on specific configuration. For formats 2 and 3, LLRs are recovered from demodulation and used for decoding. Each front-end processing kernel also calculates RSSI, and RSRP and uses DMRS to perform SINR, interference, and timing advance estimation.

For formats 2 and 3, payloads less than 12 bits in length are handled by the Reed Muller decoder kernel detailed in Section 3.6.10. Payloads of 12 bits and larger are handled by a de-rate matching and de-interleaving kernel (polSegDeRmDeItlKernel) and then processed by the polar decoder kernel detailed in Section 3.6.11.

Finally, formats 2 and 3 decoded payloads are segmented by a segmentation kernel (pucchF234UciSegKernel) to recover the corresponding HARQ, SR, and CSI payloads.

---

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PucchRx::m_tPrmDataRxBufCpu[i].tlnfoDataRx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_C_16_F : tensor vector of IQ samples</td>
</tr>
<tr>
<td>Dimensions</td>
<td>([ORAN_MAX_PRB*CUPHY_N_TONES_PER_PRB), OFDM_SYMBOLS_PER_SLOT, MAX_AP_PER_SLOT)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PucchRx::m_outputPrms.pF0UciOutGpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>cuphyPucchF0F1UciOut_t*</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Single dimensional array of length equal to the number of format 0 UCIs</td>
</tr>
<tr>
<td>Description</td>
<td>HARQ values and estimator measurements, including SINR, Interference, RSSI, RSRP (in dB) and timing advance</td>
</tr>
</tbody>
</table>

---

**Fig. 11:** Graph Diagram of the PUCCH Pipeline

---

1.4. Aerial cuPHY Developer Guide 379
<table>
<thead>
<tr>
<th>Data type</th>
<th>cuphyPucchF0F1UciOut_t*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>Single dimensional array of length equal to the number of format 1 UCIs</td>
</tr>
<tr>
<td>Description</td>
<td>HARQ values and estimator measurements, including SINR, Interference, RSSI, RSRP (in dB) and timing advance (in uSec)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tSinr</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured SINR per UCI (in dB)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tRssi</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured RSSI per UCI (in dB)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tRsrp</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured RSRP per UCI (in dB)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tInterf</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured Interference per UCI (in dB)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tNoiseVar</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured Noise Variance per UCI (in dB)</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PucchRx:: m_tTaEst</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>Measured Timing Advance per UCI (in uSec)</td>
</tr>
</tbody>
</table>
### Table 14 – continued from previous page

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PucchRx::m_tUciPayload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_8U : tensor vector of unsigned bytes</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(total number payload bytes for format 2 &amp; 3 UCIs rounded up to 4-byte words for each payload)]</td>
</tr>
<tr>
<td>Description</td>
<td>Format 2 &amp; 3 UCI payloads rounded to 4-byte words. If 1 UCI has HARQ &amp; CSI-P1 of 1 bit each, they each get a 4-byte word for a total of 8 bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PucchRx::m_tHarqDetectionStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_8U : tensor vector of unsigned bytes</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>HARQ detection status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PucchRx::m_tCsiP1DetectionStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_8U : tensor vector of unsigned bytes</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>CSI Part 1 detection status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PucchRx::m_tCsiP2DetectionStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_8U : tensor vector of unsigned bytes</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(number of format 2 &amp; 3 UCIs)]</td>
</tr>
<tr>
<td>Description</td>
<td>CSI Part 2 detection status</td>
</tr>
</tbody>
</table>

#### 1.4.2.5.7 PRACH Pipeline

The PRACH pipeline uses IQ samples segmented for each occasion and performs detection and estimation for configured PRACH signals. This process operates across a number of kernels as follows:

1. The prach_compute_correlation kernel takes input IQ data and performs averaging among repetitions followed by a time-domain correlation (done in frequency domain) against a reference version of the expected PRACH signal. This kernel simultaneously operates on each PRACH occasion.

2. An inverse FFT kernel transforms the frequency domain correlation results to time domain. A separate kernel operates on each occasion.

3. The prach_compute_pdp kernel performs non-coherent combining of correlation results for each preamble zone. It then calculates power and the peak index and value for each preamble zone.

4. The prach_search_pdp kernel computes preamble and noise power estimates and reports the preamble index with peak power. It also does threshold-based detection declaration.

There is also a separate set of kernels as part of the PRACH pipeline for performing RSSI calculations.

1. The memsetRssi kernel clears a device buffer used in computing RSSI.
2. The `prach_compute_rssi` kernel computes RSSI for each PRACH occasion both for each antenna and average power over all antennas.

3. The `memcpyRssi` kernel stores the RSSI results in host-accessible memory.

---

**Fig. 12: Graph Diagram of the PRACH Pipeline**

<table>
<thead>
<tr>
<th>Input Buffer</th>
<th>PrachRx:: h_dynParam[i].dataRx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_C_16_F : tensor for each occasion buffer</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[(Preamble length+5)*Number of repetitions , N_ant]</td>
</tr>
<tr>
<td>Output Buffer</td>
<td>PrachRx:: numDetectedPrmb</td>
</tr>
<tr>
<td>Data type</td>
<td>CUPHY_R_32U : tensor vector of uint32</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Number of detected preambles for each occasion</td>
</tr>
</tbody>
</table>

Output Buffer: PrachRx:: prmbIndexEstimates

Data type: CUPHY_R_32U : tensor vector of uint32
<table>
<thead>
<tr>
<th>Dimensions</th>
<th>[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Detected preamble index for each preamble and occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: prmbDelayEstimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Delay estimate for each preamble and occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: prmbPowerEstimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Power estimate for each preamble and occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: antRssi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[N_ant, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>RSSI for each antenna and occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: rssi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>RSSI for each occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: interference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Interference for each occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: prmbPowerEstimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Power estimate for each preamble and occasion</td>
</tr>
</tbody>
</table>

continues on next page
Table 15 – continued from previous page

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: antRssi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[N_ant, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>RSSI for each antenna and occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: rssi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>RSSI for each occasion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Buffer</th>
<th>PrachRx:: interference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>CUPHY_R_32_F : tensor vector of float values</td>
</tr>
<tr>
<td>Dimensions</td>
<td>[1, PRACH_MAX_OCCASIONS_AGGR]</td>
</tr>
<tr>
<td>Description</td>
<td>Interference for each occasion</td>
</tr>
</tbody>
</table>

1.4.2.5.8 Performance Optimization

The cuPHY library is designed to accelerate PHY layer functionality of commercial grade 5G gNB DU. Software optimizations ensure reduced latency and scalable performance with the increased number of cells. We can categorize them as:

- **Use of CUDA Graphs**: The cuPHY library makes use of CUDA graph feature to reduce kernel launch latency. The CUDA kernels implementing signal processing components within each cuPHY physical layer channel pipeline are represented as nodes in a CUDA graph and the inter-component dependencies as edges between nodes. Since graph creation is expensive, a base graph with the worst case topology is created during initialization of channel pipelines where there are several specializations of component kernels. When the channel is scheduled for a given slot only the necessary subset of graph nodes are updated and enabled.

- **Use of MPS (Multi-Process Service)**: The cuPHY driver creates multiple MPS contexts, each with an upper limit to the maximum number of SMs (Streaming Multiprocessors) that can be used by kernels launched there. MPS contexts for control channels (e.g. PUCCH, PDCCH) usually have significantly lower SM limits compared to MPS contexts for shared channels due to the expected computation load. Each MPS context also has one or more CUDA streams associated with it, with potentially different CUDA stream priorities.

- **Kernel fusion**: the cuPHY implementation may fuse functionality from different processing steps into a single CUDA kernel for improved performance. For example, the rate matching, scrambling and modulation processing steps of the downlink shared channel are all performed in a single kernel. The motivation for these customizations is to reduce memory access latency and therefore improve performance. For example, assume that there are two kernels that are run in sequence. The first kernel makes a computation, writes the output to the global memory and the second kernel needs to read this output from the global memory to continue the computation. In this case, fusing these two kernels can reduce the number of accesses to the global memory, which has higher latency.
Optimization of L1-L2 data flow: Data flow between the L2 and L1, and between the L1 and the FH are important for optimization of the latency. Data TB payloads for PDSCH channel need to be copied from L2 to L1 whenever a PDSCH channel is scheduled by the L2. The size of TBs increases with higher data throughput and the number of TBs also can also increase with the number of cells and the number of UEs scheduled on a given time slot. cuPHY library pipelines the TB H2D (host to device) copy to run in parallel with PDSCH channel setup processing. Such pipelining hides the TB H2D copy latency reducing overall PDSCH completion time.

1.4.2.6 Running cuPHY Examples

cuPHY library comes with example programs that can be used to test cuPHY channel pipelines and components. How to run cuPHY channel pipelines are explained in Aerial Release Guide Document in the section “Running the cuPHY Examples”. Please refer to the release guide on how to run the cuPHY channel pipelines. In running these examples, note that recent cuPHY implementation uses graphs mode to improve performance as explained in Section 3.1 of this document.

cuPHY library also includes examples for its components. Some examples are provided below.

Uplink channel estimation

cuPHY/build/examples/ch_est/cuphy_ex_ch_est -i ~/<tv_name>.h5

Sample test run:

cuPHY/build/examples/ch_est/cuphy_ex_ch_est -i TVnr_7550_PUSCH_gNB_CUPHY_s0p0.h5

UE group 0: ChEst SNR: 138.507 dB
ChEst test vector TVnr_7550_PUSCH_gNB_CUPHY_s0p0.h5 PASSED
22:53:17.726075 datasets.cpp:974 WRN[90935] [CUPHY.PUSCH_RX] LDPC throughput mode disabled
22:53:17.943272 cuphy.hpp:84 WRN[90935] [CUPHY.MEMFOOT]cuphyMemoryFootprint - GPU allocation:
684.864 MiB for cuPHY PUSCH channel object (0x7ffc16f09f90).
22:53:17.943273 pusch_rx.cpp:1188 WRN[90935] [CUPHY.PUSCH_RX] PuschRx:
Running with eqCoeffAlgo 3

Simplex decoder

cuPHY/build/examples/simplex_decoder/cuphy_ex_simplex_decoder -i ~/<tv_name>.h5

Sample test run:

cuPHY/build/examples/simplex_decoder/cuphy_ex_simplex_decoder -i TVnr_61123_SIMPLEX_gNB_CUPHY_s0p0.h5
AERIAL_LOG_PATH unset
Using default log path
Log file set to /tmp/simplex_decoder.log
22:57:33.455795 WRN 90935 0 [CUPHY.PUSCH_RX] Simplex code: found 0 mismatches out of 1 codeblocks
Exiting bg_fmtlog_collector - log queue ever was full: 0

PUSCH de-rate match
1.4.3. Using Test MAC and RU Emulator

TestMAC and RU emulator are the tools that are used by developers to test the system in a controlled environment. TestMAC functions as the L2/L1 interface, which schedules packets according to a predefined launch pattern. RU emulator is a basic implementation of ORAN FH interface. Its functions include verifying the timing of FH packets, checking the integrity of DL IQ samples and scheduling the transmission of UL IQ samples.

Functional blocks of TestMAC are displayed in the following figure. TestMAC is responsible for scheduling DL packets and validating received UL messages. TestMAC uses a predefined launch pattern for scheduling. The launch pattern defines the TDD pattern across multiple frames and the test vectors (TVs) used on each slot. The test vectors contain the L1 configuration for each PHY channel in a given slot. TestMAC obtains the slot timing from L1 via L2 adapter. The timing is indicated by the slot indication message. TestMAC prepares the FAPI message according to the L1 configuration contained in the TV. If a given slot is an UL, TestMAC parses the corresponding TV and compares the received data with the expected values included in the TV.

The RU emulator has the following functions:

- Validation of timing of the transmitted packets by the DU (DL u-plane, DL c-plane, UL c-plane)
- Validation of the transmitted IQ samples or DL u-plane payload data
- Transmission of UL u-plane packets as a response to UL c-plane messages

The logic used by RU emulator to process received packets is displayed in the following figure. If the received packet is a U-plane, RU emulator will continue parsing the packet header to retrieve eAxC id, frame number, subframe number, slot id, startSym index, number of symbols, start PRB index and number of PRBs. It then compares the payload with the corresponding data included in the TV. If the received packet is a C-plane message for an UL packet, they are again parsed to extract the information for the UL data allocation same as for DL packets. RU emulator then transmits the UL u-plane data symbol by symbol and it uses accurate send scheduling function.
Fig. 13: Test MAC functionality
RU emulator needs the cuphycontroller configuration to obtain PCI address of the NIC interface, MAC address of the peer system, cell configurations, VLAN ID and eAxCid values for each cell. It also uses launch pattern file to understand the TDD pattern and the L1 configuration for each slot.

Fig. 14: RU Emulator received packet processing

1.4.4. Using 5G Models for Testing and Validation

Aerial CUDA-Accelerated RAN includes a simulation model called nr_sim that is written in Matlab matching with the CUDA implementation in cuPHY library. It can be found under $cuBB_SDK/5GModel/nr_matlab. It serves as a reference model for Aerial design and verification, which covers from L1/L2 FAPI interface to O-DU/O-RU FH interface.

A high level function block diagram of the nr_sim is shown in the following figure. The core of nrSim is the simulation engine nrSimulator.m, which includes Matlab models for gNB transmitter and receiver, MIMO fading channel and UE transmitter. nrSimulator.m can be called by runSim.m with external configuration mode or by runRegression.m with internal configuration mode.

The simulator provides three major features: waveform compliance test, test vector generation and PHY performance simulation.

1.4.4.1 Waveform compliance test

The purpose of waveform compliance test is to make sure our understanding of 3GPP standards regarding signal waveform generation is correct. It is achieved by checking nrSim generated signal against Matlab 5G Toolbox generated signal.

1.4.4.2 Test Vector Generation

nrSim can generate test vectors for L2/L1 FAPI PDU, cuPHY channel pipeline API parameters, cuPHY channel pipeline output and the compressed samples in a slot.

Two types of test vectors will be generated for each test case configuration.

- FAPI test vector including FAPI PDU for all the channels in this slot and FH compressed samples for this slot. There is only one FAPI TV per slot.
- cuPHY test vector including cuPHY parameters and input/output for a cuPHY channel pipeline call. There can be multiple cuPHY TVs per slot.
Fig. 15: nr_sim functionality
Fig. 16: Waveform compliance test

Fig. 17: Test vector generation
1.4.4.3 PHY Performance Simulation

The purpose of this test is to make sure that Aerial PHY performance can meet 3GPP requirement by checking nrSim performance simulation results with the same channel condition and test configuration specified by the 3GPP standard.

1.4.4.4 nrSim Configuration

The input to the simulation engine nrSimulator.m is a single data structure SysPar, which includes all the 3GPP related configurations and simulation control related configurations. The outputs of nrSimulator include SysPar, UE (array of structures for all UEs) and gNB (structure for gNB).

\[ \text{[SysPar, UE, gNB]} = \text{nrSimulator(SysPar)} \]

Matlab functions listed in the table below generate the default configuration for the parameters in SysPar.

<table>
<thead>
<tr>
<th>Data structure</th>
<th>Field</th>
<th>Description</th>
<th>Matlab function for default configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysPar</td>
<td>testAlloc</td>
<td>Specify DL/UL direction and the number of each type of channels allocated for the slot</td>
<td>initSysPar</td>
</tr>
<tr>
<td></td>
<td>carrier</td>
<td>Specify carrier level configuration</td>
<td>cfgCarrier</td>
</tr>
<tr>
<td></td>
<td>ssb</td>
<td>Specify SSB configuration</td>
<td>cfgSsb</td>
</tr>
<tr>
<td></td>
<td>pdccch</td>
<td>Specify PDCCH channel configuration</td>
<td>cfgPdcch</td>
</tr>
<tr>
<td></td>
<td>pdsch</td>
<td>Specify PDSCH channel configuration</td>
<td>cfgPdsch</td>
</tr>
<tr>
<td></td>
<td>csirs</td>
<td>Specify CSIRS channel configuration</td>
<td>cfgCsirs</td>
</tr>
<tr>
<td></td>
<td>prach</td>
<td>Specify PRACH channel configuration</td>
<td>cfgPrach</td>
</tr>
<tr>
<td></td>
<td>pucch</td>
<td>Specify PUCCH channel configuration</td>
<td>cfgPucch</td>
</tr>
<tr>
<td></td>
<td>pusch</td>
<td>Specify PUSCH channel configuration</td>
<td>cfgPusch</td>
</tr>
<tr>
<td></td>
<td>srs</td>
<td>Specify SRS channel configuration</td>
<td>cfgSrs</td>
</tr>
<tr>
<td></td>
<td>Chan</td>
<td>Specify MIMO fading channel configuration</td>
<td>cfgChan</td>
</tr>
<tr>
<td></td>
<td>Sim-Ctrl</td>
<td>Specify Simulation control parameters</td>
<td>cfgSimCtrl</td>
</tr>
</tbody>
</table>

Configuration options for the testAlloc is summarized in the table below. DL and UL fields indicate if the test is for a DL or an UL slot. The remaining fields hold the number of PHY channel allocations for the test. A given test can include multiple combinations of PHY channels, i.e. 1 SSB allocation, 4 PDCCH allocations, 4 PDSCH allocations, etc.
### Data structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>testAlloc</td>
<td>Enable DL test</td>
</tr>
<tr>
<td>UI</td>
<td>Enable UL test</td>
</tr>
<tr>
<td>Ssb</td>
<td>Enable SSB allocation</td>
</tr>
<tr>
<td>Pdcch</td>
<td>Number of PDCCH channels in a slot</td>
</tr>
<tr>
<td>Pdsch</td>
<td>Number of PDSCH channels in a slot</td>
</tr>
<tr>
<td>Csirs</td>
<td>Number of CSIRS channels in a slot</td>
</tr>
<tr>
<td>Prach</td>
<td>Number of PRACH channels in a slot</td>
</tr>
<tr>
<td>Pucch</td>
<td>Number of PUCCH channels in a slot</td>
</tr>
<tr>
<td>Pusch</td>
<td>Number of PUSCH channels in a slot</td>
</tr>
<tr>
<td>Srs</td>
<td>Number of SRS channels in a slot</td>
</tr>
</tbody>
</table>

SysPar definition for 3GPP carrier and slot configuration with each channel is mostly based on SCF-FAPI specification.

The Chan configuration refers to MIMO fading channel model.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chan</td>
<td>'AWGN', 'TDLx-xx-xxx' (3GPP MIMO fading channel)</td>
</tr>
<tr>
<td>SNR</td>
<td>Channel SNR in dB</td>
</tr>
<tr>
<td>Delay</td>
<td>Channel propagation delay in second</td>
</tr>
<tr>
<td>CFO</td>
<td>Carrier frequency offset in Hz</td>
</tr>
<tr>
<td>Use5Gtoolbox</td>
<td>Reserved</td>
</tr>
<tr>
<td>gain</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The SimCtrl structure includes global configuration settings that are used in the simulation.
<table>
<thead>
<tr>
<th>Data structure</th>
<th>Field</th>
<th>Sub-field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCtrl</td>
<td>N_UE</td>
<td>Number of UEs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N_frame</td>
<td>Number of frames per run</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N_slot_run</td>
<td>Number of slots in a frame to run. (0: run all slots in a frame)</td>
<td></td>
</tr>
<tr>
<td>t ylim</td>
<td>eDo-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mainSim</td>
<td></td>
<td>Enable time domain simulation (required for applying fading channel model, delay and CFO)</td>
<td></td>
</tr>
<tr>
<td>plotFigure</td>
<td>tfGrid</td>
<td>Plot time/freq domain signal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>constella-</td>
<td>Plot constellation before and after equalizer</td>
<td></td>
</tr>
<tr>
<td>genTV</td>
<td>Enable</td>
<td>Enable TV generation at gNB side</td>
<td></td>
</tr>
<tr>
<td>enableUE</td>
<td></td>
<td>Enable TV generation at UE side</td>
<td></td>
</tr>
<tr>
<td>tvDirName</td>
<td>Name for TV directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuPHY</td>
<td></td>
<td>Enable cuPHY TV in h5 format</td>
<td></td>
</tr>
<tr>
<td>FAPI</td>
<td></td>
<td>Enable FAPI TV in h5 format</td>
<td></td>
</tr>
<tr>
<td>FAPYaml</td>
<td></td>
<td>Enable FAPI TV in yaml format</td>
<td></td>
</tr>
<tr>
<td>slotIdx</td>
<td>Indices of slots on which TV will be generated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for ceS-</td>
<td>lotIdxFlag</td>
<td>Force slot index = slotIdx(1) for every slot</td>
<td></td>
</tr>
<tr>
<td>bypass-</td>
<td>Comp</td>
<td>Bypass FH sample compression</td>
<td></td>
</tr>
<tr>
<td>idx</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TVname</td>
<td>Prefix for the name of TVs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fp16AlgoSel</td>
<td>0: Use half function (Matlab fixed point toolbox required) 1: Use vpf16 function (Matlab fixed point toolbox not required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFOflag</td>
<td>Enable CFO correction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>enableRssiM</td>
<td>eas</td>
<td>Enable RSSI measurement</td>
<td></td>
</tr>
<tr>
<td>capSamp</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>result</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.4.4.5 nrSim Usage

For different test and simulation purpose, nrSim provides two modes to change the configurations and run the Matlab model.

▶ External configuration mode (runSim): This mode is to use an external configuration file in yaml format to update the parameters. nrSim will read this yaml configuration file and set the SysPar parameters accordingly. It is recommended that non matlab developer uses this mode to generate test vectors which requires no change to the Matlab code.

▶ Internal configuration mode (runRegression): This mode is to change the SysPar parameters directly in the Matlab code between initSysPar and nrSimulator. Matlab developer can pre-define a set of configuration used by compliance test, test vector generation and PHY performance simulation. Multiple runs can be performed in this mode with different configurations.

1.4.4.6 Matlab Environment Preparation

Matlab version:
▶ R2020a or later

Matlab licenses:
▶ MATLAB
▶ Communications Toolbox
▶ DSP System Toolbox
▶ Signal Processing Toolbox
▶ Fixed-Point Designer (optional)
  ▶ Call half function to accelerate testing/simulation
  ▶ Can be disabled by setting SimCtrl.fp16AlgoSel = 1
▶ Parallel Computing Toolbox (optional)
  ▶ Accelerate testing/simulation automatically
▶ 5G Toolbox (optional)
  ▶ Not required for TV generation
  ▶ Required for waveform compliance test and performance simulation

Preparation:
▶ After download the source code, launch Matlab on the directory of nr_matlab and run startup to add all sub-directories into Matlab search path.
1.4.4.7 External Configuration Mode (runSim)

1) Find the yaml configuration template file cfg_template.yaml under nr_matlab. If it is missing, run genCfgTemplate to generate it.

2) Use a text editor to change the parameters in the yaml file. Basically cfg_template.yaml is a yaml (text) version of SysPar data structure. Please refer to section 3 for the description of SysPar parameters. After change is done, save it to another file name, for example, cfg_test.yaml.

3) Run runSim(cfg_filename, tv_filename), for example, runSim('cfg_test.yaml', 'my_test'). nrSim will read cfg_test.yaml file, update SysPar accordingly, run nrSimulator and generate test vector files with name starting with my_test. The generated TV files are stored under the folder named by SysPar.SimCtrl.tvDirName, for example, GPU_test_input.

4) Another option is to use runSim(cfg_filename, 'test', tv_filename),

Notes:
- This mode only supports test vector generation with SimCtrl.genTV.enable set to 1. It does not support waveform compliance test and PHY performance test.
- If SimCtrl.plotFigure.tfGrid is set to 1, the time/freq signal in a frame or the specified number of slots in a frame (controlled by N_slot_run) can be plotted to provide visualized channel allocations.
- Non Matlab developer can write script in any language to modify the yaml template file and automatically generate a number of different yaml configuration files for different testing purpose.

1.4.4.8 Internal Configuration Mode (runRegression)

Instead of updating configuration through the external yaml configuration file case by case, the internal configuration mode changes SysPar parameters directly inside the Matlab code, which allows Matlab developer to define and execute a batch of test cases more efficiently. The main function for this mode is runRegression, which supports a flexible combination of testSet, channelSet and caseSet as the input arguments.

runRegression(testSet, channelSet, caseSet)

<table>
<thead>
<tr>
<th>Values</th>
<th>Value selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>testSet</td>
<td>'Compliance', 'TestVector', 'Performance', 'allTests'</td>
</tr>
<tr>
<td>channelSet</td>
<td>'ssb', 'pdcch', 'pdsch', 'csirs', 'dlmix', 'allDL', 'prach', 'pucch', 'pusch', 'srs', 'ulmix', 'allUL', 'allChannels'</td>
</tr>
<tr>
<td>caseSet</td>
<td>'full', 'compact', 'selected'</td>
</tr>
</tbody>
</table>

Here are some example commands.

- Full regression test for all channels
  runRegression({'allTests'}, {'allChannels'}, 'full')

- Waveform compliance test and test vector generation for pdcch and pdsch channels with compact set
  runRegression({'Compliance', 'TestVector'}, {'pdcch', 'pdsch'}, 'compact')

- PHY performance simulation for PRACH channel
The test cases for each channel are defined in the Matlab file testCompGenTV_xxxx.m, where xxxx is the channel name. Matlab developer can modify the Matlab file to create and assign test cases for full set, compact set and selected set.

- **full set** includes all the test cases which can be generated by nrSim and pass waveform compliance test against 5G Toolbox.
- **compact set** includes a subset of full set test cases which are supported by cuPHY implementation. TVs from Compact set can be used for nightly CICD regression test.
- **selected set** includes a subset of compact set test cases which are essential for cuPHY verification. TVs from Selected set can be used for merge request (MR) CICD regression test.

Notes:
- testCompGenTV_dlmix and testCompGenTV_ulmix supports multi-channel multi-slot TV generation without waveform compliance check.
- testPerformance_prach, testPerformance_pusch and testPerformance_pucch support PHY performance test for PRACH (format 0/B4), PUSCH (non-UCI) and PUCCH (format 0/1).

Below is an example of full regression test summary with Matlab command

```matlab
runRegression({'allTests'}, {'allChannels'}, 'full')
```

<table>
<thead>
<tr>
<th>Channel</th>
<th>Compliance_Test</th>
<th>Error</th>
<th>Test_Vector</th>
<th>Error</th>
<th>Performance_Test</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSB</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PDCCH</td>
<td>47</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PDSCH</td>
<td>222</td>
<td>0</td>
<td>222</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CSIRS</td>
<td>55</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DLMIX</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PRACH</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>PUCCH</td>
<td>110</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>PUSCH</td>
<td>199</td>
<td>0</td>
<td>199</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>SRS</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ULMIIX</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Total: 670 0 692 0 140 0

Elapsed time is 1221.657852 seconds.

Fig. 18: An example output of a full regression test summary

### 1.4.5. References

[1] 3GPP, "NR; Physical channels and modulation," 3GPP TR 38.211, v15.4.0.
[2] 3GPP, "NR; Multiplexing and channel coding," 3GPP TR 38.212, v15.4.0.
[3] 3GPP, "NR; Physical layer procedures for control," 3GPP TR 38.213, v15.4.0.
[4] 3GPP, "NR; Physical layer procedures for data," 3GPP TR 38.214, v15.4.0.
[5] 3GPP, "NR; Physical layer measurements," 3GPP TR 38.215, v15.4.0.
Aerial CUDA-Accelerated RAN is a set of software defined libraries that are optimized to run 5G gNB workloads on GPU. These libraries include cuPHY, cuMAC and pyAerial. In this section, we focus on layer-1 (L1), or physical (PHY) layer of 5G gNB software stack as defined by 3GPP [1-5].

cuPHY is the 5G L1 library of the Aerial CUDA-Accelerated RAN. It is designed as an inline accelerator to run on NVIDIA GPUs and it does not require any additional hardware accelerator. It is implemented according to the O-RAN 7.2 split option [8]. cuPHY library takes advantage of massively parallel GPU architecture to accelerate computationally heavy signal processing tasks. It also makes use of fast GPU I/O interface between the NVIDIA Bluefield-3 (BF3) NIC and GPU (GPU Direct RDMA [7]) to improve the latency.

BF3 NIC provides the fronthaul (FH) connectivity in addition to the IEEE 1588 compliant timing synchronization. The BF3 NIC also has a built-in SyncE and eCPRI windowing functionality, which meets G.8273.2 timing requirements.

In the following, we first give an overview of cuPHY library software stack. cuPHY library consists of L1 controller components running on the CPU and PHY layer functions running on the GPU. After providing the overview, we will go into details of each component and explain how L1 controller components interact with each other and L2. Finally, we will go over the PHY layer signal processing functions, which are accelerated as CUDA kernel implementations.

Fig. 19: Aerial CUDA-Accelerated Software Stack within 5G gNB DU
### 1.4.6. Acronyms and Definitions

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>Third Generation Partnership Project</td>
</tr>
<tr>
<td>5G NR</td>
<td>Fifth generation new radio</td>
</tr>
<tr>
<td>CB</td>
<td>Code Block</td>
</tr>
<tr>
<td>CSI</td>
<td>Channel State Information</td>
</tr>
<tr>
<td>CSI-RS</td>
<td>Channel State Information Reference Signal</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture</td>
</tr>
<tr>
<td>cuBB</td>
<td>CUDA base-band (L1 software stack consisting of L2 adapter, PHY control layer and PHY layer)</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture</td>
</tr>
<tr>
<td>cuPHY</td>
<td>CUDA PHY (L1 functionality on the GPU accelerator in inline mode)</td>
</tr>
<tr>
<td>DCI</td>
<td>Downlink Control Information</td>
</tr>
<tr>
<td>DL</td>
<td>Downlink</td>
</tr>
<tr>
<td>DMRS</td>
<td>Demodulation Reference Signal</td>
</tr>
<tr>
<td>DU or O-DU</td>
<td>O-RAN Distributed Unit (a logical node hosting RLC/MAC/High-PHY layers based on a lower layer functional split)</td>
</tr>
<tr>
<td>eCPRI</td>
<td>Ethernet Common Public Radio Interface</td>
</tr>
<tr>
<td>eAxC</td>
<td>Extended Antenna Carrier: a data flow for a single antenna (or spatial stream) for a single carrier in a single sector</td>
</tr>
<tr>
<td>FAPI</td>
<td>Functional Application Programming Interface</td>
</tr>
<tr>
<td>FH</td>
<td>Fronthaul</td>
</tr>
<tr>
<td>H2D</td>
<td>Host-to-device memory</td>
</tr>
<tr>
<td>LDPC</td>
<td>Low-density Parity Check</td>
</tr>
<tr>
<td>NIC</td>
<td>Network interface card</td>
</tr>
<tr>
<td>O-RAN</td>
<td>Open RAN</td>
</tr>
<tr>
<td>PBCH</td>
<td>Physical Broadcast Channel</td>
</tr>
<tr>
<td>PDCCH</td>
<td>Physical Downlink Control Channel</td>
</tr>
<tr>
<td>PDSCH</td>
<td>Physical Downlink Shared Channel</td>
</tr>
<tr>
<td>PRACH</td>
<td>Physical Random Access Channel</td>
</tr>
<tr>
<td>PUCCH</td>
<td>Physical Uplink Control Channel</td>
</tr>
<tr>
<td>PUSCH</td>
<td>Physical Uplink Shared Channel</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>RM</td>
<td>Reed-Muller</td>
</tr>
<tr>
<td>RU or O-RU</td>
<td>O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split</td>
</tr>
<tr>
<td>SCF</td>
<td>Small Cell Forum</td>
</tr>
</tbody>
</table>

continues on next page
1.5. Aerial cuMAC

1.5.1. Getting Started with cuMAC

All cuMAC data structures and scheduler module classes are included in the name space cumac
The header files api.h and cumac.h should be included in the application program of cuMAC

1.5.1.1 Data Flow

A diagram of cuMAC data flow for both CPU MAC scheduler host and GPU execution is given in following figure:

Each cuMAC scheduler module (UE selection, PRB allocation, layer selection, MCS selection, etc.) is implemented as a C++ class, consisting of constructors with different combinations of input arguments, a destructor, a setup () function to set up the CUDA kernels in each TTI and a run () function to execute the scheduling algorithms in each TTI.

All parameters and data buffers required by the cuMAC scheduler modules are wrapped into three cuMAC API data structures, including cumacCellGrpUeStatus, cumacCellGrpPrms, and cumacSchdSol. Each of these data structures contains a number of constant parameters, and a number of data buffers whose memories are allocated on GPU.

In the initialization phase, the objects of all cuMAC scheduler modules are created using their corresponding constructors. Meanwhile, the above-mentioned three API data structures are also created, with their constant parameters being properly set up and data buffers getting memory allocations on GPU.

In the per-TTI execution, the CPU MAC scheduler host first prepares all the required data in GPU memory for the three API data structures. Then the setup () function of each cuMAC scheduler module is called 1) to pass the required constant parameters and addresses of the data buffer GPU memories from the API data structures to the scheduler module objects, and 2) to complete the internal configuration of the CUDA kernels. Next, the run () function of each schedule module is called to execute the scheduling algorithms and obtain the scheduling solutions. Finally, the CPU MAC host transfers the computed scheduling solutions from GPU to CPU and applies them in the system.
1.5.1.2 Quick Setup

1.5.1.2.1 Prerequisites

1. **CMake (version 3.18 or newer)**
   
   If you have a version of CMake installed, the version number can be determined as follows:
   
   ```
   cmake --version
   ```
   
   You can download the latest version of CMake from the official CMake website.

2. **CUDA (version 12 or newer)**

   CMake intrinsic CUDA support will automatically detect a CUDA installation using a CUDA compiler (nvcc), which is located via the PATH environment variable. To check for nvcc in your PATH:
   
   ```
   which nvcc
   ```
   
   To use a non-standard CUDA installation path (or to use a specific version of CUDA):
   
   ```
   export PATH=/usr/local/cuda-12.0/bin:$PATH
   ```
   
   For more information on CUDA support in CMake, see [https://devblogs.nvidia.com/building-cuda-applications-cmake/](https://devblogs.nvidia.com/building-cuda-applications-cmake/). (The statement above is equivalent to " -gencode arch=compute_80,code=sm_80 -gencode arch=compute_90,code=sm_90 ".)

3. **cuMAC** requires a minimum GPU architecture of Ampere or newer.

4. **HDF5** (Hierarchical Data Format 5)
The cuMAC CMake system currently checks for a specific version (1.10) of HDF5. To install a specific version of HDF5 from a source code archive:

4.1. Remove the original hdf5 library (if necessary)
   ```bash
dpkg -l \| grep hdf5
   sudo apt-get remove <name of these libraries>
   ```

4.2. To build from source:
   ```bash
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10.5/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz
   tar -xzf hdf5-1.10.5.tar.gz
   cd hdf5-1.10.5
   ./configure --prefix=/usr/local --enable-cxx
              --enable-build-mode=production
   sudo make install
   ```

1.5.1.2.2 Getting and building cuMAC

1. To download cuMAC, you can use the following link:
   ```bash
git clone --recurse-submodules https://gitlab-master.nvidia.com/gputelecom/cumac
   ```

2. To build cuMAC, use the following commands:
   ```bash
   cd cumac
   mkdir build && cd build
   cmake ..
   make
   ```

Additional CMake options:

Creating a release build (using the default list of target architectures):
   ```bash
cmake -DCMAKE_BUILD_TYPE=Release ..
   ```

Creating a debug build (using the default list of target architectures):
   ```bash
cmake .. -DCMAKE_BUILD_TYPE=Debug
   ```

Specifying a single GPU architecture (e.g., to reduce compile time):
   ```bash
cmake .. -DCMAKE_CUDA_ARCHITECTURES="80"
   ```

Specifying multiple GPU architectures:
   ```bash
cmake .. -DCMAKE_CUDA_ARCHITECTURES="80;90"
   ```

(The statement above is equivalent to "-gencode arch=compute_80,code=sm_80 -gencode arch=compute_90,code=sm_90 ".)
1.5.2. cuMAC API Reference

1.5.2.1 cuMAC API Data Structures

1.5.2.1.1 CumacCellGrpPrms

API data structure containing cell group information of the coordinated cells.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nUe</td>
<td>uint16_t</td>
<td>Total number of selected UEs in a TTI of all coordinated cells. Value: 0 -&gt; 65535</td>
</tr>
<tr>
<td>nActiveUe</td>
<td>uint16_t</td>
<td>Total number of active UEs of all coordinated cells. Value: 0 -&gt; 65535</td>
</tr>
<tr>
<td>numUeSchdPerCellTTI</td>
<td>uint8_t</td>
<td>Number of UEs selected/scheduled per TTI per cell. Value: 0 -&gt; 255</td>
</tr>
<tr>
<td>nCell</td>
<td>uint16_t</td>
<td>Total number of coordinated cells. Value: 0 -&gt; 65535</td>
</tr>
<tr>
<td>nPrbGrp</td>
<td>uint16_t</td>
<td>Total number of PRGs per cell. Value: 0 -&gt; 65535</td>
</tr>
<tr>
<td>nBsAnt</td>
<td>uint8_t</td>
<td>Number of BS antenna ports. Value: 0 -&gt; 255</td>
</tr>
<tr>
<td>nUeAnt</td>
<td>uint8_t</td>
<td>Number of UE antenna ports. Value: 0 -&gt; 255</td>
</tr>
<tr>
<td>W</td>
<td>float</td>
<td>Frequency bandwidth (Hz) of a PRG. Value: 12 * subcarrier spacing * number of PRBs per PRG</td>
</tr>
<tr>
<td>sigmaSqrld</td>
<td>float</td>
<td>Noise variance. Value: noise variance value in watts</td>
</tr>
<tr>
<td>precodingScheme</td>
<td>uint8_t</td>
<td>Precoder type. Value: 0: No precoding 1: SVD precoder</td>
</tr>
<tr>
<td>receiver-Scheme</td>
<td>uint8_t</td>
<td>Receiver/equalizer type. Value: Currently only support 1: MMSE-IRC receiver</td>
</tr>
<tr>
<td>allocType</td>
<td>uint8_t</td>
<td>PRG allocation type. Value: 0: non-consecutive type-0 allocation 1: consecutive type-1 allocation</td>
</tr>
<tr>
<td>betaCoeff</td>
<td>float</td>
<td>Coefficient for adjusting the cell-edge UEs' performance in multi-cell scheduling. Value: non-negative real number. The default value is 1.0, representing the classic proportional-fairness scheduling.</td>
</tr>
<tr>
<td>sinValThr</td>
<td>float</td>
<td>Singular value threshold for layer selection. Value: in (0, 1). Default value is 0.1</td>
</tr>
<tr>
<td>prioWeight-Step</td>
<td>uint16_t</td>
<td>For priority-weight based scheduling algorithm. Step size for UE priority weight increment per TTI if UE does not get scheduled. Value: default 100</td>
</tr>
<tr>
<td>cellld</td>
<td>uint16_t[nCell]</td>
<td>IDs of coordinated cells. One dimensional array.</td>
</tr>
</tbody>
</table>
1.5.2.1.2 cumacCellGrpUeStatus

API data structure containing the per-UE information of the coordinated cell group.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>av-gRates</td>
<td>float[nUe]</td>
<td>Array of the long-term average data rates of the selected UEs in the coordinated cells. One dimensional array. Value of each element: Denote uIdx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. avgRates[uIdx] is the long-term average throughput of the uIdx-th selected UE in the coordinated cells.</td>
</tr>
<tr>
<td>av-gRate-sActUe</td>
<td>float[nActiveUe]</td>
<td>Array of the long-term average data rates of all active UEs in the coordinated cells. One dimensional array. Value of each element: Denote uIdx = 0, 1, ..., nActiveUe-1 as the global active UE index in the coordinated cells. avgRatesActUe[uIdx] is the long-term average throughput of the uIdx-th active UE in the coordinated cells.</td>
</tr>
<tr>
<td>prioWeigh-tActUe</td>
<td>u int16_t[nActiveUe]</td>
<td>For priority-based UE selection. Priority weights of all active UEs in the coordinated cells. One dimensional array. Value of each element: Denote uIdx = 0, 1, ..., nActiveUe-1 as the global UE index for all active UEs in the coordinated cells. prioWeightActUe[uIdx] is the uIdx-th active UE’s priority weight. 0xFFFF indicates an invalid element.</td>
</tr>
<tr>
<td>tbErrLas-rtLast</td>
<td>int8_t[nUe]</td>
<td>Array of the selected UEs’ transport block (TB) decoding error indicators of the last transmissions. One dimensional array. Value of each element: Denote uIdx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. tbErrLast[uIdx] is the uIdx-th selected UE’s TB decoding error indicator of the last transmission. -1 - the last transmission is not a new transmission (is a re-transmission) 0 - decoded correctly 1 - decoding error <strong>Note that if the last transmission of a UE is not a new transmission, tbErrLast of that UE should be set to -1.</strong></td>
</tr>
<tr>
<td>tbErrLas-rtActUe</td>
<td>int8_t[nActiveUe]</td>
<td>TB decoding error indicators of all active UEs in the coordinated cells. One dimensional array. Value of each element: Denote uIdx = 0, 1, ..., nActiveUe-1 as the global UE index for all active UEs in the coordinated cells. tbErrLastActUe[uIdx] is the uIdx-th active UE’s TB decoding error indicator.</td>
</tr>
</tbody>
</table>
1.5.2.1.3 cumacSchdSol

API data structure containing the scheduling solutions.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| setSchdUePerCellTTI   | `uint16_t[nCell*numUeSchdPerCellTTI]` | Set of global IDs of the selected UEs per cell per TTI. One dimensional array. Value of each element:  
Denote clIdx = 0, 1, ..., nCell-1 as the coordinated cell index. 
Denote i = 0, 1, ..., numUeSchdPerCellTTI-1 as the i-th selected UE in a given cell.  
`setSchdUePerCellTTI[clIdx*numUeSchdPerCellTTI + i]` is within `{0, 1, ..., nActiveUe-1}` and represents the global active UE index of the i-th selected UE in the clIdx-th coordinated cell. |
| allocSol              | For type-0 PRG allocation:  
`int16_t[nCell*nPrbGrp]` 
For type-1 PRG allocation:  
`int16_t[2*nUe]` | PRB group allocation solution for the selected UEs per TTI in the coordinated cells. One dimensional array. Value of each element:  
For type-0 PRG allocation:  
Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.  
Denote clIdx = 0, 1, ..., nCell-1 as the coordinated cell index.  
`allocSol[prgIdx*nCell + clIdx]` indicates the selected UE index (0, 1, ..., nUe-1) that the prgIdx-th PRG is allocated to in the clIdx-th coordinated cell.  
-1 indicates that a given PRG in a cell is not allocated to any UE.  
For type-1 PRG allocation:  
Denote uIdx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells.  
`allocSol[2*uIdx]` is the starting PRG index of the uIdx-th selected UE.  
`allocSol[2*uIdx + 1]` is the ending PRG index of the uIdx-th selected UE plus one.  
-1 indicates that a given UE is not being allocated to any PRG. |
| pfMetriccArr          | `float[array_size]`  
array size = nCell * the minimum power of 2 that is no less than nPrbGrp*n umUeSchdPerCellTTI | Array to store the computed PF metrics per UE and per PRG. Only used for type-1 PRG allocation. One dimensional array. GPU memory allocated for CUDA kernel execution. Not used externally. Memory should be allocated when initializing the cuMAC API. Value of each element: floating-type value of a computed PF metric. |
| pfIdArr               | `uint16_t [array_size]`  
array size = nCell * the minimum power of 2 that is no less than nPrbGrp*n umUeSchdPerCellTTI | Array to indicate the PRG and UE indices of the sorted PF metrics. Only used for type-1 PRG allocation. One dimensional array. GPU memory allocated for CUDA kernel execution. Not used externally. Memory should be allocated when initializing the cuMAC API. Value of each element: 0 -> 65535 |
1.5.2.2 cuMAC Scheduler Module API

1.5.2.2.1 Multi-cell proportional-fairness UE down-selection

Wrapper class and public member functions:

```cpp
class cumac::multiCellUeSelection
{
public:
    // constructor
    multiCellUeSelection();
    // destructor
    ~multiCellUeSelection();

    // setup() function for per-TTI algorithm execution
    void setup(cumac::cumacCellGrpUeStatus* cellGrpUeStatus,
               cumac::cumacSchdSol* schdSol,
               cumac::cumacCellGrpPrms* cellGrpPrms,
               uint8_t in_enableHarq,
               cudaStream_t strm);
    // requires external synchronization
    // set in_enableHarq to 1 if HARQ is enabled; 0 otherwise

    // run() function for per-TTI algorithm execution
    void run(cudaStream_t strm);
    // requires external synchronization

    // parameter/data buffer logging function for debugging purpose
    void debugLog();
    // for debugging only, printing out dynamic descriptor parameters
}
```

1.5.2.2.2 Multi-cell proportional-fairness PRB scheduler

Wrapper class and public member functions:

```cpp
class cumac::multiCellScheduler
{
public:
    // constructor
    multiCellScheduler();
    // destructor
    ~multiCellScheduler();

    // setup() function for per-TTI algorithm execution
    void setup(cumac::cumacCellGrpUeStatus* cellGrpUeStatus,
               cumac::cumacSchdSol* schdSol,
               cumac::cumacCellGrpPrms* cellGrpPrms,
               uint8_t in_DL,
               uint8_t in_columnMajor,
               uint8_t in_halfPrecision,
               uint8_t in_lightWeight,
               cudaStream_t strm);
    // (continues on next page)
```
// set in_DL to 1 if setup for DL scheduling; 0 otherwise
// in_columnMajor: 0 - row-major channel access, 1 - column-major channel access
// in_halfPrecision: 0 - call FP32 floating type kernel, 1 - call FP16 (bfloat162)
// half-precision kernel
// in_lightWeight: 0 - call heavy-weight kernel, 1 - call light-weight kernel
// in_enableHarq: 0 - HARQ disabled, 1 - HARQ enabled
// requires external synchronization

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);
// requires external synchronization

// parameter/data buffer logging function for debugging purpose
void debugLog();
// for debugging only, printing out dynamic descriptor parameters

1.5.2.2.3 Multi-cell layer selection

Wrapper class and public member functions:

class cumac::multiCellLayerSel

public:
// constructor
multiCellLayerSel();

// destructor
~multiCellLayerSel();

// setup() function for per-TTI algorithm execution
void setup(cumacCellGrpUeStatus* cellGrpUeStatus,
           cumacSchdSol* schdSol,
           cumacCellGrpPrms* cellGrpPrms,
           uint8_t in_enableHarq,
           cudaStream_t strm);
// in_enableHarq: 0 - HARQ disabled, 1 - HARQ enabled
// requires external synchronization

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);
// requires external synchronization

// parameter/data buffer logging function for debugging purpose
void debugLog();
// for debugging only, printing out dynamic descriptor parameters
1.5.2.2.4 Multi-cell MCS selection + outer-loop link adaptation (OLLA)

Wrapper class and public member functions:

```cpp
class cumac::mcsSelectionLUT
public:
    // constructor
    mcsSelectionLUT(uint16_t nActiveUe, cudaStream_t strm);
    // requires external synchronization
    // uint16_t nActiveUe is the (maximum) total number of active UEs in all
    // coordinated cells
    // destructor
    ~mcsSelectionLUT();

    // setup() function for per-TTI algorithm execution
    void setup(cumacCellGrpUeStatus* cellGrpUeStatus,
               cumacSchdSol* schdSol,
               cumacCellGrpPrms* cellGrpPrms,
               uint8_t in_DL,
               uint8_t in_baseline,
               cudaStream_t strm);
    // in_DL: 0 - UL, 1 - DL
    // in_baseline: 0 - not using baseline algorithm, 1 - using baseline
    // algorithm
    // requires external synchronization

    // run() function for per-TTI algorithm execution
    void run(cudaStream_t strm);

    // parameter/data buffer logging function for debugging purpose
    void debugLog();
    // for debugging only, printing out dynamic descriptor parameters
```

Outer-loop link adaptation (OLLA) data structure:

```cpp
struct ollaParam {
    float delta; // offset to SINR estimation
    float delta_ini; // initial value for delta parameter
    float delta_up; // step size for increasing delta parameter
    float delta_down; // step size for decreasing delta parameter
};
```

1.5.3. Examples

1.5.3.1 cuMAC test vectors generated as HDF5 files

Test vectors are located in the testVectors directory. Each test vector contains parameters and data arrays defined in the cuMAC API structures (aerial_sdk/cuMAC/src/api.h): cumacCellGrpUeStatus, cumacCellGrpPrms, and cumacSchdSol.

Parameter configurations can be specified the aerial_sdk/cuMAC/examples/parameters.h file.
Use the `multiCellSchedulerUeSelection` testbench (aerial_sdk/cuMAC/examples/multiCellSchedulerUeSelection) to create TVs:

- **DL TV:**
  ```
  ./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/multiCellSchedulerUeSelection -t 1
  ```

- **UL TV:**
  ```
  ./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/multiCellSchedulerUeSelection -d 0 -t 1
  ```

An H5 TV is created after the last simulated TTI. The assumption is that the simulation duration is long enough so that the scheduler algorithm's performance converges.

### 1.5.3.2 Single-TTI tests

Given the same input parameters of a single TTI, GPU and CPU implementations of the same scheduler algorithms should give the same output solution.

Two types of tests:

- **Per scheduler module tests:** DL/UL UE selection, DL/UL PRG allocation, DL/UL layer selection, and DL/UL MCS selection
- **Complete DL/UL scheduler pipeline tests**

**TV loading-based single-TTI testbench (aerial_sdk/cuMAC/examples/tvLoadingTest).**

After building cumac, use the following commend to check input arguments of the testbench: `.aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -h`

- **Per scheduler module tests:**
  
  - **DL UE selection:**
    ```
    ./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 1 -m 01000
    ```
  
  - **DL PRG allocation:**
    ```
    ./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 1 -m 00100
    ```
  
  - **DL layer selection:**
    ```
    ./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 1 -m 00010
    ```
  
  - **DL MCS selection:**
    ```
    ./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 1 -m 00001
    ```

  - UL scheduler modules can be tested by setting input argument: `-d 0`

- **Complete DL/UL scheduler pipeline tests**
  
  - DL/UL scheduler modules executed sequentially: UE selection > PRG allocation > layer selection > MCS selection
DL scheduler pipeline:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 1 -m 01111
```

UL scheduler pipeline:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV] -g 2 -d 0 -m 01111
```

Passing criteria:
Solutions computed by CPU and GPU should match exactly: testbench returns 1 (PASS) or 0 (FAIL)

1.5.3.3 Continuous-time tests

With the same initial state, GPU and CPU implementations of the same scheduler algorithms should achieve similar performance curves when running for a period of time.

Complete DL/UL scheduler pipeline tests

- Continuous-time testbench

```
(aerial_sdk/cuMAC/examples/multiCellSchedulerUeSelection)
```

- After building cumac, use the following commend to check input arguments of the testbench:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/multiCellSchedulerUeSelection -h
```

- No need to use pre-generated H5 TVs. All parameters are computed using cuMAC internal simulator.

- Simulator configuration can be specified using the `aerial_sdk/cuMAC/examples/parameters.h` file.

- DL/UL scheduler modules executed sequentially: UE selection > PRG allocation > layer selection > MCS selection

- DL scheduler pipeline test:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/multiCellSchedulerUeSelection
```

- UL scheduler pipeline test:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/multiCellSchedulerUeSelection -d 0
```

Passing criteria:
Performance curves achieved by GPU and CPU scheduler implementations should match: testbench returns 1 (PASS) or 0 (FAIL)

Two types of performance curves:

- Sum throughput of all cells
- CDF of per-UE throughput
Aerial cuMAC is a CUDA-based platform for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs. cuMAC supported scheduler functions include UE selection/grouping, PRB allocation, layer selection, MCS selection/link adaptation and dynamic beamforming, all designed for the joint scheduling of multiple coordinated cells. cuMAC offers a C/C++ based API for the offloading of scheduler functions from the L2 stack in the DUs to GPUs. In the future, cuMAC will evolve into a platform that combines AI/ML based scheduler enhancements with GPU acceleration.

Fig. 21: Aerial L2 scheduler acceleration data flow chart

cuMAC is the main component of the Aerial L2 scheduler acceleration solution. The figure above illustrates the overall data flow of the scheduler acceleration. The full solution consists of the following components: 1) Aerial Scheduler Acceleration API, which is a per-cell message passing-based interface between the 3rd party L2 stack on DU/CU and cuMAC-CP, 2) cuMAC-CP, 3) cell group-based cuMAC API, and 4) cuMAC multi-cell scheduler (cuMAC-sch) modules.

The 3rd party L2 stack sits on the CPU and contains a single-cell L2 scheduler for each individual cell under its control. To offload L2 scheduling to GPU for acceleration/performance purposes, in each time slot (TTI), the L2 stack host sends per-cell request messages to cuMAC-CP through the Aerial Scheduler Acceleration API, which consists of required scheduling input & config. information from each single-cell scheduler. Upon receiving the per-cell request messages, cuMAC-CP integrates all scheduler input information from those (coordinated) cells into the cuMAC API cell group data structures and populates the GPU data buffers contained in these structures. Next, the cuMAC multi-cell scheduler (cuMAC-sch) modules are called by cuMAC-CP through cuMAC API to compute scheduling solutions for the given time slot (TTI). After the cuMAC-sch modules complete the computation and the scheduling solutions become available in the GPU memory, cuMAC-CP converts them into per-cell
response messages and sends them back to the L2 stack host on CPU through the Aerial Scheduler Acceleration API. Finally, the L2 stack host uses the obtained solutions to schedule the cells under its control.

When there are multiple coordinated cell groups, a separate set of Aerial Scheduler Acceleration API, cuMAC-CP, cuMAC API and cuMAC instances should be constructed and maintained for each cell group.

**Implementation Details**

- **Multi-cell scheduling** - All cuMAC scheduling algorithms are implemented as CUDA kernels that are executed by GPU and jointly compute the scheduling solutions (PRB allocation, MCS selection, layer selection, etc.) for a group of cells at the same time. The algorithms can be constrained to single cell scheduling by configuring a single cell in the cell group. A comparison between the single-cell scheduler and multi-cell scheduler approaches is given in the below figure.

![Fig. 22: Single-cell scheduler approach vs. multi-cell scheduler approach](image)

- **Scheduling algorithm CUDA implementation**
  
  - **PF UE down-selection algorithm** - cuMAC offers a PF-based UE selection algorithm to down-select a subset of UEs for new transmissions or HARQ re-transmissions in each TTI from the pool of all active UEs in each cell of a cell group. The association of UEs and cells in the cell group is an input to the UE selection module. When selecting UEs for each cell in each TTI, the UE selection algorithm first assigns a priority weight to each active UE in a cell and then sorts all active UEs in descending order of the priority weight. The subset of UEs that have the highest priority weights in each cell are selected for scheduling in a TTI. The number of selected UEs per cell is an input parameter to this module. HARQ re-transmissions are always assigned with the highest priority weight. For the new-transmission UEs, their priority weights are the PF metrics, calculated as the ratio of each UE’s long-term average throughput and its instantaneous achievable data rate. The UE selection algorithm is implemented as CUDA kernels that run on GPU and jointly select UEs for all cells in a cell group at the same time.

  - **PF PRB allocation algorithms** - cuMAC offers algorithms to perform channel-aware and frequency-selective PRB allocation among a group of cells and their connected active UEs on a per-TTI basis. The input arguments to the PRB allocation algorithms include the narrowband SRS channel estimates (MIMO channel matrices) per cell-UE link, the association solutions between cells and UEs, and other UE status and cell group parameters. The output is the PRB allocation solution for the cell group, whose data format depends on the type
of allocation: 1) for type-0 allocation, a per UE binary bitmap indicating whether each PRB is allocated to the UE, and 2) for type-1 allocation, with 2 elements per UE indicating the starting and ending PRB indices for the UE’s allocation. Two versions of the PRB allocation algorithms are provided, one for single cell scheduling and the other for multi-cell joint scheduling. A major difference between the two versions is that the multi-cell algorithm considers the impact of inter-cell interference in the evaluation of per-PRB SINRs, which can be derived from the narrow-band SRS channel estimates. The single-cell version does not explicitly consider inter-cell interference and only utilizes information restricted to each individual cell. The multi-cell algorithm can lead to a globally optimized resource allocation in a cell group by leveraging all available information from the coordinated multiple cells. A prototyping CUDA kernel implementation of PRB allocation algorithms is provided in the figure below.

Layer selection algorithm - cuMAC offers layer selection algorithms that choose the best set of layers for transmission for a UE based on the singular value distribution across the UE’s multiple layers. A predetermined singular value threshold is used to find the number of layers (with descending singular values) that can be supported on each subband (PRB group). Then the minimum number of layers across all allocated subbands to the UE is chosen as the optimal layer selection solution. Input arguments to the layer selection algorithms include the PRB allocation solution per UE, the singular values of each UE’s channel on its allocated subbands, the association solutions between cells and UEs, and other UE status and cell group parameters. The output is the per-UE layer selection solution. The layer selection algorithm is implemented as CUDA kernels that run on GPU and jointly select layers for all UEs in a cell group at the same time.

MCS selection algorithm - cuMAC offers MCS selection algorithms that choose the best feasible MCS (highest level that can meet a given BLER target) per UE based on a given PRB allocation solution. An outer-loop link adaptation algorithm is integrated internally to the MCS selection algorithm, which offsets the SINR estimates based on previous transport block decoding results per UE link. Input arguments to the MCS selection algorithms include the PRB allocation solution per UE, the narrow-band SRS channel estimates (MIMO channel matrices) per cell-UE link, the association solutions between cells and UEs, the decoding results of the last transport block for each UE, and other UE status and cell group parameters. The output is the per-UE MCS selection solution. The MCS selection algorithm is implemented as CUDA kernels that run on GPU and jointly select MCS for all UEs in a cell group at the same time.

Support for HARQ - all the above cuMAC scheduler algorithms can support HARQ re-transmissions with non-adaptive mode, i.e., reusing the same scheduling solution of the initial transmission for re-transmissions.

CPU reference code - CPU C++ implementation of the above algorithms is also provided for verification and performance evaluation purposes.

Different CSI types - cuMAC offers scheduler algorithm CUDA kernels to work with different CSI types, including SRS channel coefficient estimates and CSI-RS based channel quality information.

Support for FP32 and FP16 - cuMAC offers scheduler algorithm CUDA kernels implemented in FP32 and FP16. Using FP16 kernels can help reduce scheduler latency with a minor performance loss.
Fig. 23: A prototyping CUDA kernel implementation of PRB allocation algorithms
Chapter 2. Aerial Data Lake

6G will be artificial intelligence (AI) native. AI and machine learning (ML) will extend through all aspects of next generation networks from the radio, baseband processing, the network core including system management, orchestration and dynamic optimization processes. GPU hardware, together with programming frameworks will be essential to realize this vision of a software defined native-AI communication infrastructure.

The application of AI/ML in the physical layer has particularly been a hot research topic.

There is no AI without data. While the synthetic data generation capabilities of Aerial Omniverse Digital Twin (AODT) and Sionna/SionnaRT are essential aspects of a research project, availability of over-the-air (OTA) waveform data from real-time systems is equally important. This is the role of Aerial Data Lake. It is a data capture platform supporting the capture of OTA radio frequency (RF) data from virtual radio access network (vRAN) networks built on the Aerial CUDA-Accelerated RAN. Aerial Data Lake consists of a data capture application (app) running on the base station (BS) distributed unit (DU), a database of samples collected by the app, and an application programming interface (API) for accessing the database.

2.1. Target Audience

Industry and university researchers and developers looking to bring ML to the physical layer with the end goal of benchmarking on OTA testbeds like NVIDIA ARC-OTA or other GPU-based BSs.

2.2. Key Features

Aerial Data Lake has the following features:

**Real-time capture of RF data from OTA testbed**

- Aerial Data Lake is designed to operate with gnBs built on the Aerial CUDA-Accelerated RAN and that employ the Small Cell Forum FAPI interface between L2 and L1. One example system being the NVIDIA ARC-OTA network testbed. I/Q samples from O-RUs connected to the GPU platform via a O-RAN 7.2x split fronthaul interface are delivered to the host CPU and exported to the Aerial Data Lake database.

**Aerial Data Lake APIs to access the RF database**

- The data passed to the layer-2 via RX_Data.Indication and UL_TTI.Request are exported to the database. The fields in these data structures form the basis of the database access APIs.
Scalable and time coherent over arbitrary number of BSs

- The data collection app runs on the same CPU that supports the DU. It runs on a single core, and the database runs on free cores. Because each BS is responsible for collecting its own uplink data, the collection process scales as more BSs are added to the network testbed. Database entries are time-stamped so data collected over multiple BSs can be used in a training flow in a time-coherent manner.

Use in conjunction with pyAerial to generate training data for neural network physical layer designs

- Aerial Data Lake can be used in conjunction with the NVIDIA pyAerial CUDA-Accelerated Python L1 library. Using the Data Lake database APIs, pyAerial can access RF samples in a Data Lake database and transform those samples into training data for all the signal processing functions in an uplink or downlink pipeline.

2.3. Design

Aerial Data Lake sits beside the Aerial L1 and copies out data that would be useful for machine learning into an external database.

Fig. 1: The Aerial Data Lake data capture platform as part of the gNB.

Uplink I/Q data from one or more O-RAN radio units (O-RUs) is delivered to GPU memory where it is both processed by the Aerial L1 PUSCH baseband pipeline and delivered to host CPU memory. The Aerial Data Lake collector process writes the I/Q samples to the Aerial Data Lake database in the $fh$ table. The $fh$ table has columns for SFN, Slot, IQ samples as $fhData$, and the start time of that SFN.slot as $TsTaN$. 
The collector app saves data that the L2 sent to L1 to describe UL OTA transmissions in UL_TTI.Request messages as well as data returned to L2 the via RX_Data.Indication and CRC.Indication. This data is then written to the fapi database table. These messages and the fields within them are described in SCF 5G FAPI PHY Spec version 10.02, sections 3.4.3, 3.4.7, and 3.4.8.

Each gNB in a network testbed collects data from all O-RUs associated with it. That is, data collection over the span of a network is performed in a distributed manner, each gNB is building its own local database. Training can be performed locally at each gNB, and site-specific optimizations can be realized with this approach. Since the data in a database is time-stamped, the local databases can be consolidated at a centralized compute resource and training performed using the time aligned aggregated data. In cases where the aerial pusch pipeline was unable to decode due to channel conditions, retransmissions can be used as ground truth as long as one of the retransmissions succeeds, allowing the user to test algorithms with better performance than the originals.

The Aerial Data Lake database storage requirements depend on the number of O-RUs, the antenna configuration of the O-RU, the carrier bandwidth, the TDD pattern and the number of samples to be collected. Collecting IQ samples of 1 million transmissions from a single RU 4T4R O-RU employing a single 100MHz carrier will consume approximately 660 GB of storage.

Aerial Data Lake database comprises the fronthaul RF data. However, for many training applications access to data at other nodes in the receive pipeline is required. A pyAerial pipeline, together with the Data Lake database APIs, can access samples from an Aerial Data Lake database and transform that data into training data for any function in the pipeline.

Figure 2 illustrates data ingress from a Data Lake database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft de-mapper.

Fig. 2: pyAerial is used in conjunction with the NVIDIA data collection platform, namely, Aerial Data Lake to build training data sets for any node in the layer-1 downlink or uplink signal processing pipeline. The example shows a Data Lake database of over-the-air samples transformed into training data for a neural network soft de-mapper.
2.4. Installation

Aerial Data Lake is compiled by default as part of cuphycontroller. If you would like to record fresh data every time cuphycontroller is started, see the section on Fresh Data.

Start by installing Clickhouse database on the server collecting the data. The command below will download and run an instance of the clickhouse server in a docker container.

```
docker run -d \
--network=host \
-v $(realpath ./ch_data):/var/lib/clickhouse/ \
-v $(realpath ./ch_logs):/var/log/clickhouse-server/ \
--cap-add=SYS_NICE --cap-add=NET_ADMIN --cap-add=IPC_LOCK \
--name my-clickhouse-server --ulimit nofile=262144:262144 clickhouse/clickhouse-server
```

By default clickhouse will not drop large tables, and will return an error if attempted. The clickhouse-cpp library does not return exceptions so to avoid what looks like a cuphycontroller crash we recommend allowing it to drop large tables using the following command:

```
sudo touch './ch_data/flags/force_drop_table' && sudo chmod 666 './ch_data/flags/force_drop_table'
```

2.5. Usage

In the cuphycontroller adapter yaml configuration file, enable data collection by specifying a core then start cuphycontroller and usual. The core should be on the same NUMA node as the rest of cuphycontroller, i.e. should follow the same pattern as the rest of the cores An example of this can be found commented out in cuphycontroller_P5G_FXN_R750.yaml.

```
cuphydriver_config:
  # Fields added for data collection
  datalake_core: 19 # Core on which data collection runs. E.g isolated odd on R750, any isolated core on gigabyte
  datalake_address: localhost
  datalake_samples: 1000000 # Number of samples to collect for each UE/RNTI. Defaults to 1M
```

When enabled the DataLake object is created and DataLake::dbInit() initializes the two tables in the database. After cuphycontroller runs the PUSCH pipeline, cuphycontroller calls DataLake::notify() with the addresses of the data to be saved, which DataLake then saves. When DataLake::waitForLakeData wakes up it calls DataLake::dbInsert() which appends data to respective Clickhouse columns, then sleeps waiting for more data. Once 20 PUSCH transmissions have been stored or a total of data lake_samples have been recived the columns are appended to a Clickhouse::Block and inserted into the respective table.
2.6. Using Data Lake in Notebooks

Follow pyAerial instructions and usual to build and launch that container. It must be run on a server with a GPU.

Two example notebooks for are included:
- `datalake_channel_estimation.ipynb` performs channel estimation and plots the result
- `datalake_chan_estimation_decoding.ipynb` goes further and runs the full PUSCH decoding pipeline, both a fused version and a version build up constituent parts.

2.7. Notes

2.7.1. Database Administration

2.7.1.1 Clickhouse client

A clickhouse client is needed to interact with the server. To download it and run it do the following:

```bash
curl https://clickhouse.com/ | sh
./clickhouse client
```

```
aerial@aerial-gnb:~$ ./clickhouse client
ClickHouse client version 24.3.1.1159 (official build).
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 24.3.1.
aerial-gnb :)
```

You are now at the clickhouse client prompt. Commands starting with `aerial-gnb :)` are entered at this prompt and those with `$` are run on the host.

2.7.1.2 Database Import

There are example `fapi` and `fh` tables included in Aerial CUDA-Accelerated RAN. These tables can be imported into the clickhouse database by copying them to the clickhouse user_files folder, using the client to import them:

```bash
$ docker cp c_aerial_$USER:/opt/nvidia/cuORB/pyaerial/notebooks/data/fh.parquet .
$ docker cp c_aerial_$USER:/opt/nvidia/cuORB/pyaerial/notebooks/data/fapi.parquet .
$ sudo cp *.parquet .ch_data/user_files/
```

```
aerial-gnb :) create table fapi ENGINE = MergeTree primary key TsTaiNs settings allow_ → nullable_key=1 as select * from file('fapi.parquet',Parquet)
Ok.
aerial-gnb :) create table fh ENGINE = MergeTree primary key TsTaiNs settings allow_ → nullable_key=1 as select * from file('fh.parquet',Parquet)
```

(continues on next page)
Ok.

aerial-gnb :) select table, formatReadableSize(sum(bytes)) as size from system.parts
            group by table

SELECT
    `table`,
    formatReadableSize(sum(bytes)) AS size
FROM system.parts
GROUP BY `table`

Query id: 95451ea7-6ea9-4eec-b297-15de78036ada

<table>
<thead>
<tr>
<th>table</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>fh</td>
<td>4.54 MiB</td>
</tr>
<tr>
<td>fapi</td>
<td>2.18 KiB</td>
</tr>
</tbody>
</table>

You now have five PUSCH transmissions loaded in the database and can run the example notebooks.

2.7.1.3 Database Queries

To show some information about the entries (rows) you can run the following:

# Show counts of transmissions for all RNTIs
aerial-gnb :) select rnti, count(*) from fapi group by rnti

SELECT
    rnti,
    count(*)
FROM fapi
GROUP BY rnti

Query id: 76cf63d8-7302-4d73-972e-8ba7392da7ac

<table>
<thead>
<tr>
<th>rnti</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td>55581</td>
<td>5</td>
</tr>
</tbody>
</table>

# Show select information from all rows of the fapi table
aerial-gnb :) from fapi select TsTaiNs,TsSwNs,SFN,Slot,pduData

SELECT
    TsTaiNs,
    TsSwNs,
    SFN,
    Slot,
    pduData
FROM fapi

Query id: af362836-b379-46fd-85ae-0e9f62deb8ab

→ TsTaiNs  TsSwNs  SFN  Slot  pduData
### 2.7. Notes

5 rows in set. Elapsed: 0.002 sec.

#Show start times of fh table

aerial-gnb :) from fh select TsTaiNs, TsSwNs, SFN, Slot

<table>
<thead>
<tr>
<th>TsTaiNs</th>
<th>TsSwNs</th>
<th>SFN</th>
<th>Slot</th>
</tr>
</thead>
</table>

5 rows in set. Elapsed: 0.002 sec.
2.7.1.4 Fresh Data

The database of IQ samples grows quite quickly. If you want fresh data every run the tables can be dropped automatically by uncommenting these lines in cuPHY-CP/data_lakes/data_lakes.cpp:

```cpp
//dbClient->Execute("DROP TABLE IF EXISTS fapi");
//dbClient->Execute("DROP TABLE IF EXISTS fh");
```

2.7.1.5 Dropping Data

You can manually drop all of the data from the database with these commands:

```
aerial-gnb :) drop table fh
Ok.
aerial-gnb :) drop table fapi
Ok.
```

2.7.2. Jupyter notebooks

Exceptions are not always displayed in jupyter notebooks the way that it would be if a python script had been run, so in some cases it can be easier to convert the notebook to a script and run that.

```
jupyter nbconvert --to script <notebook_name>.ipynb
```

To interact with the data and code in place, specific lines can be debugged by adding `breakpoint()` inline

2.8. Known Limitations

Currently datalakes records the first UE per TTI and has been tested with a single cell per gNB as supported by the Open Air Interface L2+ stack.
Chapter 3. pyAerial

PyAerial provides a Python API towards the 5G signal processing functionality included in the Aerial cuPHY library.

3.1. Overview

As 6G research gains momentum, and with many new technologies in its purvue, one thing is clear, AI/ML will feature prominently in the next generation RAN. It will play a pivotal role in realizing all parts of the network infrastructure from the radio units, baseband processing, the network core including system management, orchestration and dynamic optimization processes. GPU hardware, together with programming frameworks will be essential to realize this vision of a software defined native-AI communication infrastructure.

The application of AI/ML in the physical layer has in particular been a hot research topic. There is a lot of emphasis on neural network architectures and optimization strategies mostly performed in the context of simulation. The next step for the research community and commercial system developers is to bring AI/ML applied in layer-1 to reality in over-the-air real-time testbeds and operator-network scale systems.

This is where pyAerial enters the picture. pyAerial is a Python library of physical layer components that can be used as part of the workflow in taking a design from simulation to real-time operation. It helps with end-to-end verification of a neural network integration into a PHY pipeline and helps bridge the gap from the world of training and simulation in TensorFlow/PyTorch to real-time operation in an over-the-air testbed.

The pyAerial library provides a Python-callable bit-accurate GPU-accelerated library for all of the signal processing CUDA kernels in the NVIDIA cuBB layer-1 PDSCH and PUSCH pipelines. In other words, the pyAerial Python classes behave in a numerically identical manner to the kernels employed in cuBB because a pyAerial class employs the exact same CUDA code as the corresponding cuBB kernel: it is the CUDA kernel but with a Python API.

Using pyAerial library components complete layer-1 pipelines can be composed in Python. User code or inference engines, from NVIDIA TensorRT, or custom CUDA code, can be included in the datapath as shown in the lower part of Figure 1. This rapid prototyping design and verification flow is used for dataplane functional performance evaluation. It is a step in the workflow for verifying a physical layer design prior to deployment in a real-time over-the-air GPU base station.

pyAerial can also be used in conjunction with the NVIDIA data collection platform Aerial Data Lake. An Aerial Data Lake database consists of RF samples from a 7.2x fronthaul interface together with L2 meta-information to enable database search and query operations. A pyAerial pipeline can access samples from Aerial Data Lake database using the Data Lake Python APIs, and transform that data into training data for any function in the pipeline. Figure 2 illustrates data ingress from a Data Lake.
database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft
de-mapper.

3.1.1. Key Features

pyAerial has the following key features:

**Feature 1: Productive Python for rapid prototyping of layer-1 pipelines**

- pyAerial library components are CUDA kernels with Python bindings. The productive environ-
  ment of Python permits the rapid assembly of signal processing pipelines in Python. All of the
  analytic and visualization aspects of Python can be used for performance characterization, signal
  visualization and debugging.

**Feature 2: Simulate machine learning in the physical layer before over-the-air operation**

- With the goal of going from model training and simulation in TensorFlow or PyTorch to real-time
  over-the-air operation, pyAerial provides a convenient way to verify, evaluate and benchmark your
  physical layer prior to deployment in an OTA testbed.

**Feature 3: Fast simulation with CUDA optimized kernels**

- pyAerial library components are CUDA under the hood. Simulation is fast on a GPU. When you
  are simulating the coding chain, including for example an LDPC decoder, optimized CUDA code
  is implementing these computationally heavy functions.

**Feature 4: Generate data sets for any node in layer-1 uplink or downlink pipeline**

- pyAerial is designed to be used in conjunction with the NVIDIA data collection platform Aerial Data
  Lake. pyAerial can access RF samples in a Data Lake database and transform those samples into
  training data for all of the signal processing functions in and uplink or downlink pipeline.

**Feature 5: Bit accurate simulation**

- Because pyAerial is Python running on CUDA, the performance you observe in BLER and other
  characterization metrics is what is identical to the performance of the real-time over-the-air
  system.

3.1.2. Target Audience

Industry and university researchers and developers looking to bring machine learning to the physical
layer with the end goal of benchmarking on over-the-air testbeds like NVIDIA ARC-OTA or other GPU-
based base stations.

3.1.3. Value Proposition

Fast bit-accurate GPU accelerated simulation of neural-network downlink and uplink signal processing
pipelines. Rapid prototyping and functional verification of a real-time layer-1 in preparation for real-
time deployment. Convenient Python environment aids debugging and provides easy access to all
nodes in the pipeline for visualization and analysis. Easy to use Python environment for producing BLER
and other statistics of interest for a real-time bit-accurate GPU layer-1 implementation. Transform RF
sample captures for over-the-air captures into data for training layer-1 functions or compositions of
multiple functions.
Fig. 1: Figure 1: Using pyAerial to verify a neural pipeline context of a full uplink pipeline. This is one of the verification steps to moving to real-time operation over-the-air on a GPU base station.

Fig. 2: Figure 2: pyAerial is used in conjunction with the NVIDIA data collection platform Aerial Data Lake to build training data sets for any node in the layer-1 downlink or uplink signal processing pipeline. The example shows a Data Lake database of over-the-air samples transformed into training data for a neural network soft de-mapper, using pyAerial. Data gets extracted at the input and output of the de-mapper, and stored in the database.
3.1.4. Release Notes

- Release version: 24-2
- Supported configurations:
  - AX800, A100X and A100 GPUs with the x86 platform.
  - CUDA Toolkit: 12.5.0
  - GPU Driver (OpenRM): 550.54.15
- Limited support on the Grace Hopper platform: The pyAerial Python package is supported, but the container does not include TensorFlow or Sionna. Thus, for example only the Aerial Data Lake example notebooks can be run on the Grace Hopper platform.
- Supported features: pyAerial exposes a subset of the cuPHY API features to Python. Currently this subset includes the following features:
  - PUSCH receiver pipeline
  - PDSCH transmission pipeline
  - Channel estimation (note: The RKHS algorithm supported by cuPHY is currently not exposed through the pyAerial API)
  - Noise and interference estimation
  - Channel equalization and soft demapping
  - LDPC encoding
  - LDPC decoding
  - LDPC rate matching
  - SRS channel estimation
  - TensorRT inference engine

3.2. Getting Started with pyAerial

3.2.1. Pre-requisites

Running pyAerial requires its own container, which also contains machine learning tools commonly used together with pyAerial:

- NVIDIA Sionna (version 0.17.0)
- NVIDIA TensorRT (version 10.0.1)
- TensorFlow (version 2.15.1)

To create and launch the pyAerial container, the following are needed:

- NVIDIA Aerial CUDA-Accelerated RAN container, see instructions here.
- Docker installation, see instructions here.
- HPC Container Maker (HPCCM) installation
The source code needs to be copied from the NVIDIA Aerial CUDA-Accelerated RAN container to a directory outside the container. The source code can be copied into the cuBB directory as follows (note that you can omit the first command if the container is already running):

```bash
docker run --rm -d --name cuBB <container image file>
docker cp cuBB:/opt/nvidia/cuBB cuBB

docker stop cuBB

cd cuBB
```

The HPC Container Maker can be installed as follows:

```bash
pip install hpccm
```

### 3.2.2. Installing pyAerial

Once the above pre-requisites are fulfilled, the pyAerial container is built using the following script:

```bash
export cuBB_SDK=`pwd`
AERIAL_BASE_IMAGE=<container image file> $cuBB_SDK/pyaerial/container/build.sh
```

The container can then be launched using the following script:

```bash
$cuBB_SDK/pyaerial/container/run.sh
```

pyAerial is pre-installed within the pyAerial container. However, it can also be built and installed as follows (these commands are issued inside the pyAerial container):

```bash
cd $cuBB_SDK

cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=../cmake/toolchains/native -DNVIPC_FMTLOG_ENABLE=OFF

cmake --build build -t _pycuphy
./pyaerial/scripts/install_dev_pkg.sh
```

**Note:** Note that pyAerial, similarly to Aerial cuPHY, is by default built for GPUs with compute capabilities 8.0 or 9.0, and these are also what pyAerial has been tested against. There is no guarantee that pyAerial will work correctly with other GPUs. However, pyAerial can be built for other compute capabilities with an additional cmake option, for example for CC 8.9:

```bash
cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=../cmake/toolchains/native -DNVIPC_FMTLOG_ENABLE=OFF -DCMAKE_CUDA_ARCHITECTURES="89"
```

### 3.2.3. Testing the installation

To test that the installation works, the example Jupyter notebooks can be run as described below. Alternatively, the unit tests can be run as follows:

```bash
$cuBB_SDK/pyaerial/scripts/run_unit_tests.sh
```

**Note:** Unit tests are based on Aerial CUDA-Accelerated RAN test vectors. Those need to be mounted within the pyAerial container, and environment variable TEST_VECTOR_DIR set to point to the test
vector directory. Refer to the Aerial CUDA-Accelerated RAN documentation on how to generate the test vectors.

One simple way to test the installation is to run (within the pyAerial container):

```python3 -c "import aerial"
```

which should pass without errors.

### 3.2.4. Running the example Jupyter notebooks

NVIDIA pyAerial contains a number of example notebooks in Jupyter notebook format. The Jupyter notebooks can be run interactively within the pyAerial container using JupyterLab. This is done by starting a JupyterLab server as follows:

```
cd $cuBB_SDK/PyAerial/notebooks
jupyter lab --ip=0.0.0.0
```

and then pointing the browser to the given address. Note that the Aerial Data Lake notebooks require the example database to be created first. Refer to Aerial Data Lake documentation on how to start the clickhouse server and create the example database.

Pre-executed versions of the notebooks are found here: *Examples of Using pyAerial*.

### 3.3. Examples of Using pyAerial

We provide a number of examples of using NVIDIA pyAerial for GPU-accelerated 5G NR signal processing, and for machine learning experiments. The examples are in Jupyter notebook format. The notebooks here are pre-executed, but they can be also interactively run following the instructions in *Getting Started with pyAerial*.

**Note:** Note that when running the notebooks, exceptions are not always displayed in Jupyter notebooks the way that it would be if a python script had been run, so in some cases it can be easier to convert the notebook to a script and run that. This can be done as follows:

```
jupyter nbconvert --to script <notebook_name>.ipynb
```

To interact with the data and code in place, specific lines can be debugged by adding `breakpoint()` inline.
3.3.1. Running a PUSCH link simulation

The first example shows how to use pyAerial for modeling 5G NR compliant PUSCH transmission and reception. In this example, the whole PUSCH pipeline is modeled within pyAerial, using the cuPHY library as a backend for GPU acceleration.

The notebook shows two ways of running the PUSCH receiver pipeline: In the first, the user only needs to make a single call using the Python API, and the whole PUSCH receiver is run. In the other, the PUSCH receiver pipeline is split into its different receiver components, each called separately using the Python API. This approach enables replacing any of the PUSCH receiver components for example by an AI/ML model, and benchmarking that against the conventional receiver.

NVIDIA Sionna is used in the example for radio channel modeling.

3.3.1.1 Using pyAerial to run a PUSCH link simulation

This example shows how to use the pyAerial cuPHY Python bindings to run a PUSCH link simulation. PUSCH transmitter is emulated by PDSCH transmission with properly chosen parameters, that way making it a 5G NR compliant PUSCH transmission. Building a PUSCH receiver using pyAerial is demonstrated in two ways, first by using a fully fused, complete, PUSCH receiver called from Python using just a single function call. The same is then achieved by building the complete PUSCH receiver using individual separate Python function calls to individual PUSCH receiver components.

The NVIDIA Sionna library is utilized for simulating the radio channel based on 3GPP channel models.

3.3.1.1 Imports

```python
# Configure the notebook to use only a single GPU and allocate only as much memory as needed.
```

3.3. Examples of Using pyAerial
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)

## 3.3.1.1.2 Parameters

Set simulation parameters, numerology, PUSCH parameters and channel parameters here.

```python
[2]: # Simulation parameters.
esno_db_range = np.arange(-5.4, -4.4, 0.2)
num_slots = 10000
min_num_tb_errors = 250

# Numerology and frame structure. See TS 38.211.
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20

num_tx_ant = 1  # UE antennas
num_rx_ant = 2  # gNB antennas
cell_id = 41     # Physical cell ID

rnti = 1234      # UE RNTI
scid = 0         # DMRS scrambling ID
data_scid = 0     # Data scrambling ID
layers = 1       # Number of layers
mcs = 2          # MCS index as per TS 38.214 table
start_prb = 0    # Start PRB index.
num_prbs = 273   # Number of allocated PRBs.
start_sym = 2    # Start symbol index.
num_symbols = 12 # Number of symbols.
dmrs_scrm_id = 41 # DMRS scrambling ID
dmrs_position = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  # Indicates which symbols
                    # are used for DMRS.
dmrs_max_len=1
dmrs_add_ln_pos=0
num_dmrs_cdm_grps_no_data = 2
enable_pusch_tdi = 0  # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1     # Equalizer algorithm
```

# Channel parameters
```python
carrier_frequency = 3.5e9  # Carrier frequency in Hz.
delay_spread = 100e-9      # Nominal delay spread in [s]. Please see the CDL documentation
# about how to choose this value.
link_direction = "uplink" # Channel model: Suitable values:
channel_model = "Rayleigh" # "Rayleigh" - Rayleigh block fading channel model (sionna.
# "CDL-x", where x is one of ["A", "B", "C", "D", "E"] - for
# 3GPP CDL channel models
```
speed = 0.8333  # UE speed [m/s]. The direction of travel will be chosen randomly within the x-y plane.

3.3.1.1.3 Create the PUSCH pipelines

As mentioned, PUSCH transmission is emulated here by the PDSCH transmission chain. Note that the static cell parameters and static PUSCH parameters are given upon creating the PUSCH transmission/reception objects. Dynamically (per slot) changing parameters are however set when actually running the transmission/reception, see further below.

```python
pusch_tx = PdschTx(
    cell_id=cell_id,
    num_rx_ant=num_tx_ant,
    num_tx_ant=num_tx_ant,
)

# This is the fully fused PUSCH receiver chain.
pusch_rx = PuschRx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo,
    # To make this equal separate PUSCH Rx components configuration:
    ldpc_kernel_launch=PuschLdpcKernelLaunch.PUSCH_RX_LDPC_STREAM_SEQUENTIAL
)

# The PUSCH receiver chain built from separately called pyAerial Python components is defined here.
class PuschRxSeparate:
    """PUSCH receiver class."
    This class encapsulates the whole PUSCH receiver chain built using pyAerial components.
    """

    def __init__(self,
        num_rx_ant,
        enable_pusch_tdi,
        eq_coeff_algo):
        """"""Initialize the PUSCH receiver."""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver.
        self.channel_estimator = ChannelEstimator(
            num_rx_ant=num_rx_ant,
            cuda_stream=self.cuda_stream
        )

        self.channel_equalizer = ChannelEqualizer(
            num_rx_ant=num_rx_ant,
            enable_pusch_tdi=enable_pusch_tdi,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream
```

(continues on next page)
```python
self.noise_intf_estimator = NoiseIntfEstimator(
    num_rx_ant=num_rx_ant,
    eq_coeff_algo=eq_coeff_algo,
    cuda_stream=self.cuda_stream
)

self.derate_match = LdpcDeRateMatch(
    enable_scrambling=True,
    cuda_stream=self.cuda_stream
)

self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)

def run(
    self,
    rx_slot,
    num_ues,
    slot,
    num_dmrs_cdms_no_data,
    dmrs_scrm_id,
    start_prb,
    num_prbs,
    dmrs_syms,
    dmrs_max_len,
    dmrs_add_ln_pos,
    start_sym,
    num_symbols,
    scids,
    layers,
    dmrs_ports,
    rntis,
    data_scids,
    code_rates,
    mod_orders,
    tb_sizes
):
    """Run the receiver."""

    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_ln_pos=dmrs_add_ln_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        scids=scids,
        layers=layers,
        dmrs_ports=dmrs_ports
    )

    # Noise and interference estimation.
```

(continues on next page)
lw_inv, noise_var_pre_eq = self.noise_intf_estimator.estimate(
  rx_slot=rx_slot,
  channel_est=ch_est,
  num_ues=num_ues,
  slot=slot,
  num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
  dmrs_scrm_id=dmrs_scrm_id,
  start_prb=start_prb,
  num_prbs=num_prbs,
  dmrs_syms=dmrs_syms,
  dmrs_max_len=dmrs_max_len,
  dmrs_add_ln_pos=dmrs_add_ln_pos,
  start_sym=start_sym,
  num_symbols=num_symbols,
  scids=scids,
  layers=layers,
  dmrs_ports=dmrs_ports
)

# Channel equalization and soft demapping. The first return value are the LLRs,
# second are the equalized symbols. We only want the LLRs now.
llrs = self.channel_equalizer.equalize(
  rx_slot=rx_slot,
  channel_est=ch_est,
  lw_inv=lw_inv,
  noise_var_pre_eq=noise_var_pre_eq,
  num_ues=num_ues,
  num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
  start_prb=start_prb,
  num_prbs=num_prbs,
  dmrs_syms=dmrs_syms,
  dmrs_max_len=dmrs_max_len,
  dmrs_add_ln_pos=dmrs_add_ln_pos,
  start_sym=start_sym,
  num_symbols=num_symbols,
  layers=layers,
  mod_orders=mod_orders
)[0]

num_data_sym = (np.array(dmrs_syms[start_sym:start_sym + num_symbols]) == 0).sum()

rntis = [(rntis[ue] << 15) + data_scids[ue] for ue in range(num_ues)]
rate_match_lengths = [num_data_sym * mod_orders[ue] * num_prbs * 12 * layers[ue] for ue in range(num_ues)]
tb_sizes = [s * 8 for s in tb_sizes]

code_rates = [c / 10240. for c in code_rates]

rvs = [0,] * num_ues
ndis = [1,] * num_ues

coded_blocks = self.derate_match.derate_match(
  input_llrs=llrs,
  tb_sizes=tb_sizes,
  code_rates=code_rates,
  rate_match_lengths=rate_match_lengths,
  mod_orders=mod_orders,
)

(continues on next page)
num_layers = layers,
redundancy_versions = rvs,
ndis = ndis,
cinits = cinits
)

code_blocks = self.decoder.decode(
    input_llrs = coded_blocks,
    tb_sizes = tb_sizes,
    code_rates = code_rates,
    redundancy_versions = rvs,
    rate_match_lengths = rate_match_lengths
)

# TODO: Use the CRC kernel here.
decoded_tbs = []
for ue_idx in range(num_ues):
    # Combine the code blocks into a transport block.
    tb = code_block_desegment(
        code_blocks = code_blocks[ue_idx],
        tb_size = tb_sizes[ue_idx],
        code_rate = code_rates[ue_idx],
        return_bits = False,
    )

    # Remove CRC - no checking, check TBs/bits directly.
    tb = tb[:-3]
    decoded_tbs.append(tb)

return decoded_tbs

pusch_rx_separate = PuschRxSeparate(
    num_rx_ant = num_rx_ant,
    enable_pusch_tdi = enable_pusch_tdi,
    eq_coeff_algo = eq_coeff_algo
)

3.3.1.1.4 Channel generation using Sionna

Simulating the transmission through the radio channel takes advantage of the channel model implementations available in NVIDIA Sionna. In Sionna, the transmission can be simulated directly in frequency domain by defining a resource grid. In our case, reference signal patterns and data carrying resource elements are defined elsewhere within the Aerial code, hence we define resource grid as a simple dummy grid containing only data symbols.

See also: Sionna documentation

[4]: # Define the resource grid.
resource_grid = sionna.ofdm.ResourceGrid(
    num_ofdm_symbols = num_ofdm_symbols,
    fft_size = fft_size,
    subcarrier_spacing = subcarrier_spacing,
    num_tx = 1,
    num_streams_per_tx = 1,
cyclic_prefix_length = cyclic_prefix_length,
num_guard_carriers = num_guard_subcarriers,

dc_null = False,
pilot_pattern = None,
pilot_ofdm_symbol_indices = None
)

resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_grid)

# Define the antenna arrays.
ue_array = sionna.channel.tr38901.Antenna(
    polarization = "single",
polarization_type = "V",
antenna_pattern = "38.901",
carrier_frequency = carrier_frequency
)

gnb_array = sionna.channel.tr38901.AntennaArray(
    num_rows = 1,
    num_cols = int(num_rx_ant / 2),
polarization = "dual",
polarization_type = "cross",
antenna_pattern = "38.901",
carrier_frequency = carrier_frequency
)

if channel_model == "Rayleigh":
    ch_model = sionna.channel.RayleighBlockFading(
        num_rx = 1,
        num_rx_ant = num_rx_ant,
        num_tx = 1,
        num_tx_ant = num_tx_ant
    )

elif "CDL" in channel_model:
    cdl_model = channel_model[-1]

    # Configure a channel impulse response (CIR) generator for the CDL model.
    ch_model = sionna.channel.tr38901.CDL(
        cdl_model,
delay_spread,
carrier_frequency,
ue_array,
gnb_array,
link_direction,
min_speed = speed
    )

else:
    raise ValueError(f"Invalid channel model {channel_model}!")

channel = sionna.channel.OFDMChannel(
    ch_model,
resource_grid,
add_awgn = True,
normalize_channel = True,
return_channel = False
)
def apply_channel(tx_tensor, No):
    """Transmit the Tx tensor through the radio channel.""
    # Add batch and num_tx dimensions that Sionna expects and reshape.
    tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
    tx_tensor = tf.reshape(tx_tensor, [-1])
    tx_tensor = resource_grid_mapper(tx_tensor)
    rx_tensor = channel((tx_tensor, No))
    rx_tensor = remove_guard_subcarriers(rx_tensor)
    rx_tensor = rx_tensor[0, 0]
    rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
    return rx_tensor

3.3.1.1.5 Helper class for simulation monitoring

This helper class plots the simulation results and shows simulation progress in a table.

[5]: class SimulationMonitor:
    """Helper class to show the progress and results of the simulation.""
    markers = ['d', 'o', 's']
    linestyles = ['-', '--', ':']
    colors = ['blue', 'black', 'red']
    def __init__(self, cases, esno_db_range):
        """Initialize the SimulationMonitor.
        Initialize the figure and the results table.
        ""
        self.cases = cases
        self.esno_db_range = esno_db_range
        self.current_esno_db_range = []
        self.start_time = None
        self.esno_db = None
        self.bler = defaultdict(list)
        self._print_headers()
        def step(self, esno_db):
            """Start next Es/No value.""
            self.start_time = datetime.datetime.now()
            self.esno_db = esno_db
            self.current_esno_db_range.append(esno_db)
        def update(self, num_tbs, num_tb_errors):
            """Update current state for the current Es/No value.""
            self._print_status(num_tbs, num_tb_errors, False)
        def _print_headers(self):
            """Print result table headers.""
            cases_str = " " * 21
            separator = " " * 21
            for case in self.cases:
                cases_str += case.center(20) + " "
(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

```
separator += "-" * 20 + " ">
print(cases_str)
print(separator)
title_str = "Es/No (dB)".rjust(12) + " TBs".rjust(8) + " ">
for case in self.cases:
    title_str += "TB Errors".rjust(12) + " BLER".rjust(8)
print(title_str)
print(("=" * 20) + " " + ("=" * 20 + " ") * len(self.cases) + "=" * 8)

def _print_status(self, num_tbs, num_tb_errors, finish):
    """Print simulation status in a table."""
    end_time = datetime.datetime.now()
t_delta = end_time - self.start_time
    if finish:
        newline_char = '\n'
    else:
        newline_char = '\r'
    result_str = f"{self.esno_db:9.2f}".rjust(12) + f"{num_tbs:8d}".rjust(8) + " ">
    for case in self.cases:
        result_str += f"{num_tb_errors[case]:8d}".rjust(12)
        result_str += f"{{num_tb_errors[case]} / num_tbs:4f}".rjust(8) + " ">
        result_str += f"{(t_delta.total_seconds() * 1000 / num_tbs):6.1f}".rjust(8)
    print(result_str, end=newline_char)

def finish_step(self, num_tbs, num_tb_errors):
    """Finish simulating the current Es/No value and add the result in the plot."""
    self._print_status(num_tbs, num_tb_errors, True)
    for case_idx, case in enumerate(self.cases):
        self.bler[case].append(num_tb_errors[case] / num_tbs)

def finish(self):
    """Finish simulation and plot the results."""
    self.fig = plt.figure()
    for case_idx, case in enumerate(self.cases):
        plt.plot(
            self.current_esno_db_range,
            self.bler[case],
            marker=SimulationMonitor.markers[case_idx],
            linestyle=SimulationMonitor.linestyles[case_idx],
            color=SimulationMonitor.colors[case_idx],
            markersize=8,
            label=case
        )
        plt.yscale('log')
        plt.ylim([0.001, 1])
        plt.xlim(np.min(self.esno_db_range), np.max(self.esno_db_range))
        plt.title("Receiver BLER Performance vs. Es/No")
        plt.ylabel("BLER")
        plt.xlabel("Es/No [dB]")
        plt.grid()
        plt.legend()
        plt.show()
```

3.3. Examples of Using pyAerial
3.3.1.1.6 Run the actual simulation

Here we loop across the Es/No range, and simulate a number of slots for each Es/No value. A single transport block is simulated within a slot. The simulation starts over from the next Es/No value when a minimum number of transport block errors is reached.

```python
[6]: cases = ["Fused", "Separate"]
    monitor = SimulationMonitor(cases, esno_db_range)

    # Loop the Es/No range.
    bler = []
    for esno_db in esno_db_range:
        monitor.step(esno_db)
        num_tb_errors = defaultdict(int)

        # Run multiple slots and compute BLER.
        for slot_idx in range(num_slots):
            slot_number = slot_idx % num_slots_per_frame

            # Get modulation order and coderate.
            mod_order, coderate = get_mcs(mcs, table_idx=1)
            tb_input = random_tb(mod_order, coderate, dmrs_position, num_prbs, start_sym, num_symbols, layers)

            # Transmit PUSCH. This is where we set the dynamically changing parameters.
            # Input parameters are given as lists as the interface supports multiple UEs.
            tx_tensor = pusch_tx.run(
                tb_input, num_ues=1, slot_number, num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
                dmrs_scrm_id=dmrs_scrm_id, num_prbs=num_prbs, dmrs_syms=dmrs_position,
                start_sym=start_sym, num_symbols=num_symbols, scids=scid, layers=1, dmrs_ports=dmrs_port,
                rnti=rnti, data_scids=data_scid, code_rates=[coderate],
                mod_orders=[mod_order],
            )

            # Channel transmission using TF and Sionna.
            No = pow(10., -esno_db / 10.)
            rx_tensor = apply_channel(tx_tensor, No)
            rx_tensor = np.array(rx_tensor)

            # Run the fused PUSCH receiver.
            # Note that this is where we set the dynamically changing parameters.
            tb_crcs, tbs = pusch_rx.run(
                rx_slot=rx_tensor, num_ues=1,
            )
        
        # Channel transmission using TF and Sionna.
        No = pow(10., -esno_db / 10.)
        rx_tensor = apply_channel(tx_tensor, No)
        rx_tensor = np.array(rx_tensor)

        # Run the fused PUSCH receiver.
        # Note that this is where we set the dynamically changing parameters.
        tb_crcs, tbs = pusch_rx.run(
            rx_slot=rx_tensor, num_ues=1,
        )
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

```python
class pusch_rx_separate:
    def run(self, rx_slot, num_ues=1, slot=slot_number,
             num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
             dmrs_scmr_id=dmrs_scmr_id, start_prb=start_prb,
             num_prbs=num_prbs, dmrs Symms=dmrs_Symms,
             dmrs_max_len=dmrs_max_len, dmrs_add_ln_pos=dmrs_add_ln_pos,
             start_sym=start_sym, num_symbols=num_symbols,
             scids=[scid], layers=[layers],
             dmrs_ports=[dmrs_port], rntis=[rnti],
             data_scids=[data_scid], code_rates=[coderate],
             mod_orders=[mod_order],
             tb_sizes=[len(tb_input)])

    def finish_step(self, num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
        monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
        monitor.finish()
```

```
3.3. Examples of Using pyAerial

<table>
<thead>
<tr>
<th>Fused</th>
<th>Separate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Es/No (dB)</td>
<td>TBs</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
</tbody>
</table>
```

(continues on next page)
(continued from previous page)

\begin{tabular}{cccc}
-5.40 & 250 & 250 & 1.0000 & 250 & 1.0000 & 95.9 \\
-5.20 & 254 & 250 & 0.9843 & 251 & 0.9882 & 94.8 \\
-5.00 & 553 & 250 & 0.4521 & 257 & 0.4647 & 95.1 \\
-4.80 & 10000 & 169 & 0.0169 & 175 & 0.0175 & 94.7 \\
-4.60 & 10000 & 0 & 0.0000 & 0 & 0.0000 & 94.6 \\
\end{tabular}

**Receiver BLER Performance vs. Es/No**

![Receiver BLER Performance vs. Es/No](image)

Fused

Separate
3.3.2. LDPC encoding-decoding chain

The second example gives an example of using cuPHY’s GPU accelerated 5G NR LDPC encoding and decoding chain (including also rate matching) modules through the pyAerial Python API. The encoding/decoding modules are expected to be useful for example in AI/ML model validation when implementing some parts of the receiver using machine learning.

3.3.2.1 Using pyAerial for LDPC encoding-decoding chain

This example shows how to use the pyAerial Python bindings to run 5G NR LDPC encoding, rate matching and decoding. Information bits, i.e. a transport block, get segmented into code blocks, LDPC encoded and rate matched onto the available time-frequency resources (resource elements), all following TS 38.212 precisely. The bits are then transmitted over an AWGN channel using QPSK modulation. At the receiver side, log likelihood ratios are extracted from the received symbols, (de)rate matching is performed and LDPC decoder is run to get the transmitted information bits. Finally, the code blocks are concatenated back into a received transport block.

pyAerial utilizes the cuPHY library underneath for all components, except code block segmentation and concatenation are currently written in Python. Also, CRCs are just random blocks of bits in this example as we can compare the transmitted and received bits directly to compute block error rates.

The NVIDIA Sionna library is utilized for simulating the radio channel.

```python
[1]: # Check platform.
    import platform
    if platform.machine() != 'x86_64':
        raise SystemExit("Unsupported platform!")

3.3.2.1.1 Imports

[2]: %matplotlib widget
    from cuda import cudart
    from collections import defaultdict
    import datetime
    import os
    os.environ["CUDA_VISIBLE_DEVICES"] = "0"
    os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"  # Silence TensorFlow.
    import numpy as np
    import sionna
    import tensorflow as tf
    import matplotlib.pyplot as plt

    from aerial.phy5g.ldpc import LdpcEncoder
    from aerial.phy5g.ldpc import LdpcDecoder
    from aerial.phy5g.ldpc import LdpcRateMatch
    from aerial.phy5g.ldpc import LdpcDeRateMatch
    from aerial.phy5g.ldpc import get_mcs
    from aerial.phy5g.ldpc import random_tb
    from aerial.phy5g.ldpc import code_block_segment
    from aerial.phy5g.ldpc import code_block_desegment
    from aerial.phy5g.ldpc import get_crc_len

(continues on next page)
# Configure the notebook to use only a single GPU and allocate only as much memory as needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)

from tensorflow.python.ops.numpy_ops import np_config
np_config.enable_numpy_behavior()

## 3.3.2.1.2 Parameters

Set simulation parameters, some numerology parameters, enable/disable scrambling etc.

```python
# Simulation parameters.
esno_db_range = np.arange(2.8, 3.5, 0.1)
num_slots = 10000
min_num_tb_errors = 250

# Numerology and frame structure. See TS 38.211.
um_prb = 100
start_sym = 0
num_symbols = 14
num_slots_per_frame = 20
num_layers = 1
dmrs_sym = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

# Rate matching procedure includes scrambling if this flag is set.
enable_scrambling = True

gernti = 20000
data_scid = 41

rnti = (rnti << 15) + data_scid
cinit = rnti
rv = 0
mcs = 10
mod_order, code_rate = get_mcs(mcs)

# The scrambling initialization value is computed as per TS 38.211
# using the RNTI and data scrambling ID.
cinit = (rnti << 15) + data_scid
```

### 3.3.2.1.3 Helper class for simulation monitoring

This helper class plots the simulation results and shows simulation progress in a table.

```python
class SimulationMonitor:
    """Helper class to show the progress and results of the simulation.""
    markers = ['d', 'o', 's']
    linestyles = ['-', '--', ':']
    colors = ['blue', 'black', 'red']
```

(continues on next page)
def __init__(self, cases, esno_db_range):
    """Initialize the SimulationMonitor.
    Initialize the figure and the results table.
    """
    self.cases = cases
    self.esno_db_range = esno_db_range
    self.current_esno_db_range = []
    self.start_time = None
    self.esno_db = None
    self.bler = defaultdict(list)
    self._print_headers()

def step(self, esno_db):
    """Start next Es/No value."""
    self.start_time = datetime.datetime.now()
    self.esno_db = esno_db
    self.current_esno_db_range.append(esno_db)

def update(self, num_tbs, num_tb_errors):
    """Update current state for the current Es/No value."""
    self._print_status(num_tbs, num_tb_errors, False)

def _print_headers(self):
    """Print result table headers."""
    cases_str = " " * 21
    separator = " " * 21
    for case in self.cases:
        cases_str += case.center(20) + " "
        separator += "-" * 20 + " "
    print(cases_str)
    print(separator)
    title_str = "Es/No (dB).rjust(12) + "TBs".rjust(8) + "",
    for case in self.cases:
        title_str += "TB Errors".rjust(12) + "BLER".rjust(8) + "",
        title_str += "ms/TB".rjust(8)
    print(title_str)
    print(("=" * 20) + " " + ("=" * 20 + " ") * len(self.cases) + ":=" * 8)

def _print_status(self, num_tbs, num_tb_errors, finish):
    """Print simulation status in a table.""
    end_time = datetime.datetime.now()
    t_delta = end_time - self.start_time
    if finish:
        newline_char = '\n'
    else:
        newline_char = '\r'
    result_str = f"{self.esno_db:.2f}".rjust(12) + f"{num_tbs:8d}".rjust(8) + "",
    for case in self.cases:
        result_str += f"{num_tb_errors[case]:8d}".rjust(12)
        result_str += f"{(num_tb_errors[case] / num_tbs):.4f}".rjust(8) + "",
        result_str += f"{(t_delta.total_seconds() * 1000 / num_tbs):6.1f}".rjust(8)
print(result_str, end=newline_char)

def finish_step(self, num_tbs, num_tb_errors):
    """Finish simulating the current Es/No value and add the result in the plot."""
    self._print_status(num_tbs, num_tb_errors, True)
    for case_idx, case in enumerate(self.cases):
        self.bler[case].append(num_tb_errors[case] / num_tbs)

def finish(self):
    """Finish simulation and plot the results."""
    self.fig = plt.figure()
    for case_idx, case in enumerate(self.cases):
        plt.plot(
            self.current_esno_db_range,
            self.bler[case],
            marker=SimulationMonitor.markers[case_idx],
            linestyle=SimulationMonitor.linestyles[case_idx],
            color=SimulationMonitor.colors[case_idx],
            markersize=8,
            label=case
        )
        plt.yscale('log')
        plt.ylim(0.001, 1)
        plt.xlim(np.min(self.esno_db_range), np.max(self.esno_db_range))
        plt.title('Receiver BLER Performance vs. Es/No')
        plt.ylabel('BLER')
        plt.xlabel('Es/No [dB]')
        plt.grid()
        plt.legend()
        plt.show()

3.3.2.1.4 Create the LDPC coding chain objects

The LDPC coding chain objects are created here. This includes the following: * LdpcEncoder which takes the information bits, i.e. the transport block, segmented into code blocks as its input, and outputs encoded code blocks. * LdpcRateMatch which takes encoded code blocks as its input and outputs a rate matched (and optionally scrambled) stream of bits. * LdpcDerateMatch which takes the received stream of log-likelihood ratios (LLRs) as its input and outputs derate matched code blocks of LLRs which can be fed to the LDPC decoding. This block performs also descrambling if scrambling is enabled in the pipeline. * LdpcDecoder which takes the output of LDPC derate matching and decodes the LLRs into code blocks that can then be further concatenated into a received transport block.

All components are based on TS 38.212 and thus can be used for transmitting/receiving 5G NR compliant bit streams.

Also the Sionna channel components and modulation mapper are created here.

[5]: # Create also the CUDA stream that running the objects requires.
cudart.cudaSetDevice(0)
cuda_stream = cudart.cudaStreamCreate()[1]
cudart.cudaStreamSynchronize(cuda_stream)

# Create the Aerial Python LDPC objects.
ldpc_encoder = LdpcEncoder(cuda_stream=cuda_stream)
ldpc_decoder = LdpcDecoder(cuda_stream=cuda_stream)
ldpc_rate_match = LdpcRateMatch(enable_scrambling=enable_scrambling, cuda_stream=cuda_stream)
ldpc_derate_match = LdpcDeRateMatch(enable_scrambling=enable_scrambling, cuda_stream=cuda_stream)

# Create the Sionna modulation mapper/demapper and the AWGN channel.
mapper = sionna.mapping.Mapper("qam", 2)
demapper = sionna.mapping.Demapper("app", "qam", 2)
channel = sionna.channel.AWGN()

[6]: case = "LDPC decoding perf."
monitor = SimulationMonitor([case], esno_db_range)

# Loop the Es/No range.
for esno_db in esno_db_range:
    monitor.step(esno_db)
    num_tb_errors = defaultdict(int)

    # Run multiple slots and compute BLER.
    for slot_idx in range(num_slots):
        slot_number = slot_idx % num_slots_per_frame

        # Generate a random transport block (in bits).
        transport_block = random_tb(
            mod_order=mod_order,
            code_rate=code_rate * 1024,
            dmrs_syms=dmrs_sym,
            num_prbs=num_prb,
            start_sym=start_sym,
            num_symbols=num_symbols,
            num_layers=num_layers,
            return_bits=True
        )
        tb_size = transport_block.shape[0]

        # Attach a CRC. This is emulated to get the TB size with CRC right, however the
        # CRC is in this case just random
        # bits as we are comparing the transmitted and received bits directly to get
        # the BLER (instead of doing an actual
        # CRC check).
        crc_length = get_crc_len(tb_size)
        crc = np.random.randint(0, 1, size=crc_length, dtype=np.uint8)
        transport_block = np.concatenate((transport_block, crc))

        # Code block segmentation happens here. Note: This is just Python at the moment.
        code_blocks = code_block_segment(tb_size, transport_block, code_rate)

        # Run the LDPC encoding. The LDPC encoder takes a K x C array as its input,
        # where K is the number of bits per code
        # and C is the number of code blocks. Its output is N x C where N is the
        # number of coded bits per code block.
        # If there is more than one code block, a code block CRC (random in this case
        # as we do not need an actual CRC) is
        # attached to
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

coded_bits = ldpc_encoder.encode(
    input_data=code_blocks,
    tb_size=tb_size,
    code_rate=code_rate,
    redundancy_version=rv
)

# Run rate matching. This needs rate matching length as its input, meaning the
# number of bits that can be
# transmitted within the allocated resource elements. The input data is fed as
# 32-bit floats.
num_data_sym = (np.array(dmrs_sym[start_sym:start_sym + num_symbols]) == 0).sum()
rate_match_len = num_data_sym * num_prb * 12 * num_layers * mod_order
rate_matched_bits = ldpc_rate_match.rate_match(
    input_data=coded_bits,
    tb_size=tb_size,
    code_rate=code_rate,
    rate_match_len=rate_match_len,
    mod_order=mod_order,
    num_layers=num_layers,
    redundancy_version=rv,
    cinit=cinit
)

# Map the bits to symbols and transmit through an AWGN channel. All this in
# Sionna.
rate_matched_bits = rate_matched_bits[:, 0]
no = sionna.utils.ebnod2no(esno_db, num_bits_per_symbol=1, coderate=1)
tx_symbols = mapper(rate_matched_bits[None])
rx_symbols = channel([tx_symbols, no])
llr = -1. * demapper([rx_symbols, no])[0, :].numpy()[0, None]

# Run receiver side (de)rate matching. The input is the received array of bits
directly, and the output
# is a NumPy array of size N x C of log likelihood ratios, represented as 32-
# bit floats. Descrambling
# is also performed here in case scrambling is enabled.
derate_matched_bits = ldpc_derate_match.derate_match(
    input_llrs=llr,
    tb_sizes=[tb_size],
    code_rates=[code_rate],
    rate_match_lengths=[rate_match_len],
    mod_orders=[mod_order],
    num_layers=[num_layers],
    redundancy_versions=[rv],
    ndis=[1],
    cinit=[cinit]
)

# Run LDPC decoding. The decoder takes the derate matching output as its input
and returns
decoded_bits = ldpc_decoder.decode(
    input_llrs=derate_matched_bits,
    tb_sizes=[tb_size],
)

(continues on next page)
code_rates=[code_rate],
redundancy_versions=[rv],
rate_match_lengths=[rate_match_len]
)[0]

decoded_tb = code_block_desegment(decoded_bits, tb_size, code_rate)
tb_error = not np.array_equal(decoded_tb[:-24], transport_block[:-24])
num_tb_errors[case] += tb_error
monitor.update(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
if (np.array(list(num_tb_errors.values())) >= min_num_tb_errors).all():
    break  # Next Es/No value.

monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
monitor.finish()

LDPC decoding perf.

<table>
<thead>
<tr>
<th>Es/No (dB)</th>
<th>TBs</th>
<th>TB Errors</th>
<th>BLER</th>
<th>ms/TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.80</td>
<td>250</td>
<td>250</td>
<td>1.0000</td>
<td>19.5</td>
</tr>
<tr>
<td>2.90</td>
<td>250</td>
<td>250</td>
<td>1.0000</td>
<td>15.1</td>
</tr>
<tr>
<td>3.00</td>
<td>250</td>
<td>250</td>
<td>1.0000</td>
<td>15.1</td>
</tr>
<tr>
<td>3.10</td>
<td>255</td>
<td>250</td>
<td>0.9804</td>
<td>15.1</td>
</tr>
<tr>
<td>3.20</td>
<td>438</td>
<td>250</td>
<td>0.5708</td>
<td>15.1</td>
</tr>
<tr>
<td>3.30</td>
<td>2128</td>
<td>250</td>
<td>0.1175</td>
<td>15.1</td>
</tr>
<tr>
<td>3.40</td>
<td>1000</td>
<td>107</td>
<td>0.0107</td>
<td>15.0</td>
</tr>
<tr>
<td>3.50</td>
<td>1000</td>
<td>3</td>
<td>0.0003</td>
<td>15.0</td>
</tr>
</tbody>
</table>
3.3.3. Dataset generation by simulation

3.3.3.1 Using pyAerial for data generation by simulation

This notebook generates a fully 5G NR compliant PUSCH/PDSCH dataset using NVIDIA cuPHY through its Python bindings in pyAerial for PUSCH/PDSCH slot generation and NVIDIA Sionna for radio channel modeling. PUSCH/PDSCH slots get generated and transmitted through different radio channels. Usually, in order to make models as generalizable as possible, it is desirable to train the models with as wide variety of different channel models as possible. This notebook enables generation of a dataset containing samples generated with a number of different channel models, including e.g. those used by 3GPP, as well as with different MCS classes and other transmission parameters.

```python
# Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

[1]: # Check platform.
3.3.3.1.1 Imports

[2]: import warnings
import itertools
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"

import numpy as np
import pandas as pd
import sionna
import tensorflow as tf
from tqdm.notebook import tqdm
from aerial.phy5g.pdsch import PdschTx
from aerial.phy5g.ldpc.util import get_mcs, random_tb
from aerial.util.fapi import dmrs_bit_array_to_fapi
from aerial.util.data import PuschRecord
from aerial.util.data import save_pickle

# This is for Sionna and pyAerial to coexist on the same GPU:
# Configure the notebook to use only a single GPU and allocate only as much memory as needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices("GPU")
tf.config.experimental.set_memory_growth(gpus[0], True)

3.3.3.1.2 Dataset generation parameters

The parameters used to generate the dataset are modified here. Note that some parameters are given as lists, meaning that multiple values may be given for those parameters. Typically one would like the training dataset to be as diverse as possible in order to make the models generalize well to various channel conditions and to different transmission parameters.

[3]: # This is the target directory. It gets created if it does not exist.
dataset_dir = 'data/example_simulated_dataset/QPSK'
os.makedirs(dataset_dir, exist_ok=True)

# Number of samples is divided roughly evenly between the options below.
um_samples = 12000

# A list of channel models: Suitable values:
# "Rayleigh" - Rayleigh block fading channel model (sionna.channel.RayleighBlockFading)
# "CDL-x", where x is one of ["A", "B", "C", "D", "E"] - for 3GPP CDL channel models
# as per TR 38.901.
channel_models = ["CDL-D"]

# Speeds to include in the dataset
# This is UE speed in m/s. The direction of travel will be chosen randomly within the x-y plane.
speeds = [0.8333]

# Delay spreads to include in the dataset.
# This is the nominal delay spread in [s]. Please see the CDL documentation about how to choose this value.
delay_spreads = [100e-9]

# A list of MCS indices (as per TS 38.214) to include in the dataset.  # MCS table value refers to TS 38.214 as follows:
# 1: TS38.214, table 5.1.3.1-1.
# 2: TS38.214, table 5.1.3.1-2.
# 3: TS38.214, table 5.1.3.1-3.
mcss = [1] # 1, 10, 19 used for QPSK, 16QAM and 64QAM, respectively.
mcs_table = 2

# Es/No values to include in the dataset.
esnos = [-0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0] # MCS 10
esnos = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5] # MCS 1

# These are fixed for the dataset.
num_tx_ant = 1
num_rx_ant = 4
cell_id = 41
carrier_frequency = 3.5e9 # Carrier frequency in Hz.
link_direction = "uplink"
layers = 1
rnti = 20001
scid = 0
data_scid = 41
dmrs_port = 1
dmrs_position = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]
start_sym = 0
num_symbols = 14
start_prb = 0
num_prbs = 273

# Numerology and frame structure. See TS 38.211.
mu = 1
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20

---

3.3.3.1.3 Channel generation

Radio channel generation is done using NVIDIA Sionna.

```
[4]: class Channel(sionna.channel.OFDMChannel):
    def __init__(self,
        link_direction,
        channel_model,
        num_tx_ant,
        num_rx_ant,
```

(continues on next page)
carrier_frequency,
delay_spread,
speed,
resource_grid):
    self.resource_grid = resource_grid
    self.resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
    self.remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_grid)

    # Define the antenna arrays.
    ue_array = sionna.channel.tr38901.Antenna(
        polarization="single",
        polarization_type="V",
        antenna_pattern="38.901",
        carrier_frequency=carrier_frequency
    )
    gnb_array = sionna.channel.tr38901.AntennaArray(
        num_rows=1,
        num_cols=int(num_rx_ant/2),
        polarization="dual",
        polarization_type="cross",
        antenna_pattern="38.901",
        carrier_frequency=carrier_frequency
    )

    if channel_model == "Rayleigh":
        ch_model = sionna.channel.RayleighBlockFading(
            num_rx=1,
            num_rx_ant=num_rx_ant,
            num_tx=1,
            num_tx_ant=num_tx_ant
        )
    elif "CDL" in channel_model:
        cdl_model = channel_model[-1]

        # Configure a channel impulse reponse (CIR) generator for the CDL model.
        ch_model = sionna.channel.tr38901.CDL(
            cdl_model,
            delay_spread,
            carrier_frequency,
            ue_array,
            gnb_array,
            link_direction,
            min_speed=speed
        )
    else:
        raise ValueError(f"Invalid channel model {channel_model}!")

super().__init__(
    ch_model,
    resource_grid,
    add_awgn=True,
    normalize_channel=True,
    return_channel=False
)
def __call__(self, tx_tensor, No):
    # Add batch and num_tx dimensions that Sionna expects and reshape.
    tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
    tx_tensor = tf.reshape(tx_tensor, (1, -1))[[None, None]
    tx_tensor = self.resource_grid_mapper(tx_tensor)
    rx_tensor = super().__call__((tx_tensor, No))
    rx_tensor = self.remove_guard_subcarriers(rx_tensor)
    rx_tensor = rx_tensor[0, 0]
    rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
    return rx_tensor

    # Define the resource grid.
    resource_grid = sionna.ofdm.ResourceGrid(
        num_ofdm_symbols=num_ofdm_symbols,
        fft_size=fft_size,
        subcarrier_spacing=subcarrier_spacing,
        num_tx=1,
        num_streams_per_tx=1,
        cyclic_prefix_length=cyclic_prefix_length,
        num_guard_carriers=num_guard_subcarriers,
        dc_null=False,
        pilot_pattern=None,
        pilot_ofdm_symbol_indices=None
    )

3.3.3.1.4 PDSCH transmitter

This creates the PDSCH transmitter. However due to the symmetry of 5G NR PDSCH and PUSCH, this may be used to generate also PUSCH frames with certain parameterization. In this notebook this is used as a PUSCH transmitter to generate uplink slots.

pxsch_tx = PdschTx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_tx_ant,
)

3.3.3.1.5 Dataset generation

The actual dataset generation is done here. The different channel, SNR and MCS parameters are swept through, with a number of samples per parameterization chosen such that the total number of samples will be close to the desired number.

The PxSCH transmitter created above is used to generate a Tx frame. This Tx frame is then fed through the Sionna-generated radio channel. The resulting data is recorded in a Parquet file containing PUSCH records following roughly the Small Cell Forum FAPI specification format.

num_cases = len(channel_models) * len(esnos) * len(speeds) * len(delay_spreads) * len(mcss)
num_samples_per_param = num_samples // num_cases
# loop different channel models, speeds, delay spreads, MCS levels etc.
pusch_records = []
for (channel_model, esno, speed, delay_spread, mcs) in \
    (pbar := tqdm(itertools.product(channel_models, esnos, speeds, delay_spreads, 
    mcss), total=num_cases)):
    status_str = f"Generating... ({channel_model} | {esno} dB | {speed} m/s | {delay_ 
    spread} s | MCS {mcs})"
    pbar.set_description(status_str)

    # Create the channel model.
    channel = Channel(
        link_direction=link_direction,
        channel_model=channel_model,
        num_tx_ant=num_tx_ant,
        num_rx_ant=num_rx_ant,
        carrier_frequency=carrier_frequency,
        delay_spread=delay_spread,
        speed=speed,
        resource_grid=resource_grid
    )

    for sample in range(num_samples_per_param):
        # Generate the dataframe.
        slot_number = sample % num_slots_per_frame

        # Get modulation order and coderate.
        mod_order, coderate = get_mcs(mcs, mcs_table)
        tb_input = random_tb(mod_order, coderate, dmrs_position, num_prbs, start_sym,
                          num_symbols, layers)

        # Transmit PxSCH. This is where we set the dynamically changing parameters.
        # Input transport block in bytes.
        tx_tensor = pxsch_tx.run(
            tb_inputs=[tb_input],
            num_ues=1,
            slot=slot_number,
            dmsr_syms=dmrs_position,
            symbols=dmrs,
            start_sym=start_sym,
            num_symbols=num_symbols,
            scids=[scid],
            layers=[layers],
            dmsr_ports=[dmsr_port],
            rnts=[rnti],
            data_scids=[data_scid],
            code_rates=[coderate],
            mod_orders=[mod_order]
        )[0]

        # Channel transmission and noise.
        No = pow(10., -esno / 10.)
        rx_tensor = channel(tx_tensor, No)
        rx_tensor = np.array(rx_tensor)
# Save the sample.
xr = os.path.join(dataset_dir, rx_iq_data_filename)
save_pickle(data=rx_tensor, filename=xr)

# Save noise power and SNR data as user data.
xr = os.path.join(dataset_dir, user_data_filename)
xr = dict(snr=esno, noise_var=No)
save_pickle(data=xr, filename=xr)

pusch_record = PuschRecord(
    pduIdx=0, SFN=(sample // num_slots_per_frame) % 1023, Slot=slot_number,
    nPDUs=1, RachPresent=0, nULSCH=1, nULCCH=0, nGroup=1, PDUSize=0, pduBitmap=1,
    RNTI=rnti, Handle=0, BWPSize=273, BWPSave=0, SubcarrierSpacing=mu,
    CyclicPrefix=0, targetCodeRate=coderate * 10, qamModOrder=mod_order,
    mcsIndex=mcs, mcsTable=mcs_table - 1, dataScramblingId=data_scid,
    nrOfLayers=1, ulDmrsSymbPos=dmrs_bit_array_to_fapi(dmrs_position),
    dmrsConfigType=0, ulDmrsScramblingId=cell_id, pushcIdentity=cell_id,
    SCID=sci, numDmrsCdmGrpsNoData=2, dmrsPorts=1, ulDmrsScramblingId=cell_id,
    SCID=sci, numDmrsCdmGrpsNoData=2, dmrsPorts=1, ulDmrsScramblingId=cell_id,
    SCID=sci, numDmrsCdmGrpsNoData=2, dmrsPorts=1, # Note that FAPI uses a different format compared to cuPHY.
    resourceAlloc=1, rbBitmap=np.array([6 * [0]]),
    rbStart=0, rbSize=273, VRBtoPRBMapping=0,
    FrequencyHopping=0, txDirectCurrentLocation=0, uplinkFrequencyShift7p5khz=0,
StartSymbolIndex=start_sym,
NrOfSymbols=num_symbols,
puschData=None,
puschUci=None,
puschPtrrs=None,
dftsOfdm=None,
Beamforming=None,

# SCF FAPI 10.02 RxData.indication message parameters:
HarqID=0,
PDULen=len(tb_input),
UL_CQI=255,  # Set to invalid 0xFF.
TimingAdvance=0,
RSSI=65535,  # Set to invalid 0xFFFF.
macPdu=tb_input,

TbCrcStatus=0,
NumCb=0,
CbCrcStatus=None,

rx_iq_data_filename=rx_iq_data_filename,
user_data_filename=user_data_filename,

errInd = ""
)
pusch_records.append(pusch_record)

print("Saving...")
df_filename = os.path.join(dataset_dir, "l2_metadata.parquet")
df = pd.DataFrame.from_records(pusch_records, columns=PuschRecord._fields)
df.to_parquet(df_filename, engine="pyarrow")
print("All done!")

0%| | 0/6 [00:00<?, ?it/s]
Saving...
All done!

This notebook generates a fully 5G NR compliant PUSCH/PDSCH dataset using pyAerial. The cuPHY library is used through its Python bindings in pyAerial for PUSCH/PDSCH slot generation, and NVIDIA Sionna is used for radio channel modeling. The PUSCH/PDSCH slots get generated and transmitted through different radio channels.

The example stores the dataset for use in the consequent LLRNet examples. Equally well the data could be generated on the fly during simulation.

3.3. Examples of Using pyAerial
3.3.4. Dataset generation for LLRNet

In this example, pyAerial is used to generate a log-likelihood ratio dataset based on the PUSCH/PDSCH dataset generated in the previous example. Using pyAerial, the complete PUSCH receiver chain is formed, and LLR data is collected after the channel equalizer. The log-likelihood ratio data is used to train an LLRNet model in the next example. LLRNet, published in


is a simple neural network model that takes equalizer outputs, i.e. the complex-valued equalized symbols, as its input and outputs the corresponding log-likelihood ratios (LLRs) for each bit, basically replacing the conventional soft demapper in the receiver chain.

**Note:** This notebook requires that the former example in Dataset generation by simulation has been run first.

3.3.4.1 LLRNet: Dataset generation

The wireless ML design flow using Aerial is depicted in the figure below.

In this notebook, we take data generated in the Using pyAerial for data generation by simulation example and generate a dataset for training LLRNet using pyAerial. **Note that the data is assumed to have been generated prior to running this notebook.**

LLRNet, published in


is a simple neural network model that takes equalizer outputs, i.e. the complex-valued equalized symbols, as its input and outputs the corresponding log-likelihood ratios (LLRs) for each bit. This model is used to demonstrate the whole ML design flow using Aerial, from capturing the data to deploying the model into 5G NR PUSCH receiver, replacing the conventional soft demapper in cuPHY. In this
Aerial CUDA-Accelerated RAN, Release 24-2

A notebook a dataset is generated. We use pyAerial to call cuPHY functionality to get equalized symbols out for pre-captured/-simulated Rx data, as well as the corresponding log-likelihood ratios from a conventional soft demapper.

```python
# Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!"")
```

3.3.4.1.1 Imports

```python
[2]: import os
    os.environ["CUDA_VISIBLE_DEVICES"] = "0"
    from cuda import cudart
    import numpy as np
    import pandas as pd
    from tqdm.notebook import tqdm
    from IPython.display import Markdown
    from IPython.display import display

    from aerial.phy5g.algorithms import ChannelEstimator
    from aerial.phy5g.algorithms import NoiseIntfEstimator
    from aerial.phy5g.algorithms import ChannelEqualizer
    from aerial.phy5g.algorithms import Demapper
    from aerial.phy5g.ldpc import LdpcDeRateMatch
    from aerial.phy5g.ldpc import LdpcDecoder
    from aerial.util.data import code_block_desegment
    from aerial.util.data import PuschRecord
    from aerial.util.data import load_pickle
    from aerial.util.data import save_pickle
    from aerial.util.fapi import dmrs_fapi_to_bit_array

    import warnings
    warnings.filterwarnings("error")
```

3.3.4.1.2 Load the source data

The source data can be either real data collected from an over the air setup, or synthetic data generated by simulation.

**Note:** This notebook uses data generated using this notebook: *Using pyAerial for data generation by simulation*, which needs to be run before this notebook.

```python
[3]: # This is the source data directory which is assumed to contain the source data.
    DATA_DIR = "data/
    source_dataset_dir = DATA_DIR + "example_simulated_dataset/QPSK/"
    # This is the target dataset directory. It gets created if it does not exist.
    target_dataset_dir = DATA_DIR + "example_llrnet_dataset/QPSK/"
    os.makedirs(target_dataset_dir, exist_ok=True)

    # Load the main data file.
    try:
        df = pd.read_parquet(source_dataset_dir + "l2_metadata.parquet", engine="pyarrow")
```

3.3. Examples of Using pyAerial
except FileNotFoundError:
    display(Markdown("**Data not found - has example_simulated_dataset.ipynb been run?...**"))

print(f"Loaded {df.shape[0]} PUSCH records.")

Loaded 12000 PUSCH records.

### 3.3.4.1.3 Dataset generation

Here, pyAerial is used to run channel estimation, noise/interference estimation and channel equalization to get the equalized symbols, corresponding to the LLRNet input, as well as the log-likelihood ratios, corresponding to the LLRNet target output.

[4]: cuda_stream = cudart.cudaStreamCreate()[1]

    # Take modulation order from the first record. The assumption is that all entries have the same modulation order here.
    mod_order = df.loc[0].qamModOrder
    # These hard-coded too.
    num_rx_ant = 2
    enable_pusch_tdi = 1
    eq_coeff_algo = 1

    # Create the PUSCH Rx components for extracting the equalized symbols and log-likelihood ratios.
    channel_estimator = ChannelEstimator(
        num_rx_ant=num_rx_ant,
        cuda_stream=cuda_stream
    )
    noise_intf_estimator = NoiseIntfEstimator(
        num_rx_ant=num_rx_ant,
        eq_coeff_algo=eq_coeff_algo,
        cuda_stream=cuda_stream
    )
    channel_equalizer = ChannelEqualizer(
        num_rx_ant=num_rx_ant,
        eq_coeff_algo=eq_coeff_algo,
        enable_pusch_tdi=enable_pusch_tdi,
        cuda_stream=cuda_stream
    )
    derate_match = LdpcDeRateMatch(enable_scrambling=True, cuda_stream=cuda_stream)
    demapper = Demapper(mod_order=mod_order)
    decoder = LdpcDecoder(cuda_stream=cuda_stream)

    # Loop through the PUSCH records and create new ones.
    pusch_records = []
    tb_errors = []
    snrs = []
    for pusch_record in (pbar := tqdm(df.itertuples(index=False), total=df.shape[0])):
        pbar.set_description("Running cuPHY to get equalized symbols and log-likelihood ratios...")
        num_ues = 1

(continues on next page)
start_prb = pusch_record.rbStart
num_prbs = pusch_record.rbSize
start_sym = pusch_record.StartSymbolIndex
num_symbols = pusch_record.NrOfSymbols
dmrs_sym = dmrs_fapi_to_bit_array(pusch_record.ulDmrsSymbPos)
dmrs_scrm_id = pusch_record.ulDmrsScramblingId
dmrs_max_len = 1
dmrs_add_ln_pos = 1
num_dmrs_cdms_no_data = pusch_record.numDmrsCdmGrpsNoData
num_layers = pusch_record.nrOfLayers
scid = pusch_record.SCID
dmrs_ports = pusch_record.dmrsPorts
slot = pusch_record.Slot
tbs = len(pusch_record.macPdu)
code_rate = pusch_record.targetCodeRate / 10240.
rv = 0
ndi = 1
rnti = pusch_record.RNTI
data_scrm_id = pusch_record.dataScramblingId
ref_tb = pusch_record.macPdu

# Just making sure the hard-coded value is correct.
assert mod_order == pusch_record.qamModOrder

# Load received IQ samples.
rx_iq_data_filename = source_dataset_dir + pusch_record.rx_iq_data_filename
rx_slot = load_pickle(rx_iq_data_filename)
um_rx_ant = rx_slot.shape[2]

# Load user data.
user_data_filename = source_dataset_dir + pusch_record.user_data_filename
user_data = load_pickle(user_data_filename)

# Run the channel estimation (cuPHY).
ch_est = channel_estimator.estimate(
    rx_slot=rx_slot,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_sym=dmrs_sym,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    scids=[scid],
    layers=[num_layers],
    dmrs_ports=[dmrs_ports]
)

# Run noise/interference estimation (cuPHY), needed for equalization.
lw_inv, noise_var_pre_eq = noise_intf_estimator.estimate(
    rx_slot=rx_slot,
    channel_est=ch_est,
num_ues = num_ues,
slot = slot,
num_dmrs_cdm_grps_no_data = num_dmrs_cdm_grps_no_data,
dmrs_scrm_id = dmrs_scrm_id,
start_prb = start_prb,
num_prbs = num_prbs,
dmrs_syms = dmrs_sym,
dmrs_max_len = dmrs_max_len,
dmrs_add_ln_pos = dmrs_add_ln_pos,
start_sym = start_sym,
um_symbols = num_symbols,
scids = [scid],
layers = [num_layers],
dmrs_ports = [dmrs_ports]
)

# Run equalization and mapping to log-likelihood ratios.
llrs, equalized_sym = channel_equalizer.equalize(
    rx_slot = rx_slot,
    channel_est = ch_est,
    lw_inv = lw_inv,
    noise_var_pre_eq = noise_var_pre_eq,
    num_ues = num_ues,
    num_dmrs_cdm_grps_no_data = num_dmrs_cdm_grps_no_data,
    start_prb = start_prb,
    num_prbs = num_prbs,
    dmrs_syms = dmrs_sym,
    dmrs_max_len = dmrs_max_len,
    dmrs_add_ln_pos = dmrs_add_ln_pos,
    start_sym = start_sym,
    num_symbols = num_symbols,
    layers = [num_layers],
    mod_orders = [mod_order]
)

ree_diag_inv = channel_equalizer.ree_diag_inv[0]

# Just pick one (first) symbol from each PUSCH record for the LLRNet dataset.
# This is simply to reduce the size of the dataset - training LLRNet does not
# require a lot of data.
user_data["llrs"] = llrs[0][:, mod_order, 0, :, 0]
user_data["eq_syms"] = equalized_sym[0][0, :, 0]
map_llrs = demapper.demap(equalized_sym[0][0, :, 0], ree_diag_inv[0, ...])
user_data["map_llrs"] = map_llrs

# Save pickle files for the target dataset.
rx_iq_data_fullpath = target_dataset_dir + pusch_record.rx_iq_data_filename
user_data_fullpath = target_dataset_dir + pusch_record.user_data_filename
save_pickle(data=rx_slot, filename=rx_iq_data_fullpath)
save_pickle(data=user_data, filename=user_data_fullpath)

pusch_records.append(pusch_record)

# Run through the rest of the receiver pipeline to verify that this was legit LLR
data.
# De-rate matching and descrambling.
cinit = (rnti << 15) + data_scrm_id
num_data_sym = (np.array(dmrs_sym[start_sym:start_sym + num_symbols]) == 0).sum()
rate_match_len = num_data_sym * mod_order * num_prbs * 12 * num_layers
coded_blocks = derate_match.derate_match(
    input_llrs=llrs,
    tb_sizes=[tbs * 8],
    code_rates=[code_rate],
    rate_match_lengths=[rate_match_len],
    mod_orders=[mod_order],
    num_layers=[num_layers],
    redundancy_versions=[rv],
    ndis=[1],
    cinits=[cinit]
)

# LDPC decoding of the derate matched blocks.
code_blocks = decoder.decode(
    input_llrs=coded_blocks,
    tb_sizes=[tbs * 8],
    code_rates=[code_rate],
    redundancy_versions=[rv],
    rate_match_lengths=[rate_match_len]
)[0]

# Combine the code blocks into a transport block.
tb = code_block_desegment(
    code_blocks=coded_blocks,
    tb_size=tbs * 8,
    code_rate=code_rate,
    return_bits=False
)

tb_errors.append(not np.array_equal(tb[:tbs], ref_tb[:tbs]))
snrs.append(user_data['snr'])

0% | 0/12000 [00:00<?, ?it/s]

[5]: print("Saving...")
df_filename = os.path.join(target_dataset_dir, "l2_metadata.parquet")
df = pd.DataFrame.from_records(pusch_records, columns=PuschRecord._fields)
df.to_parquet(df_filename, engine="pyarrow")
print("All done!")

Saving...
All done!

3.3. Examples of Using pyAerial
3.3.5. LLRNet model training

In this example, the LLR data from the previous example is used to train and validate an LLRNet model for computing log-likelihood ratios. The trained LLRNet is plugged in the PUSCH receiver chain, replacing the conventional soft demapper, and its performance is validated. The model also gets exported into ONNX format consumed by the NVIDIA TensorRT inference engine.

The example shows essentially how to use pyAerial for validating a component of the PUSCH receiver chain, and how to export a model in a format that is ready to be integrated in a real system.

**Note:** This notebook requires that the former example in *Dataset generation for LLRNet* has been run first - that generates the data for this notebook.

3.3.5.1 LLRNet: Model training and testing

The wireless ML design flow using Aerial is depicted in the figure below.

![Wireless ML design flow diagram](image)

In this notebook, we use the generated LLRNet data for training and validating LLRNet as part of the PUSCH receiver chain, implemented using pyAerial, with Aerial SDK cuPHY library working as the backend. The LLRNet is plugged in the PUSCH receiver chain in place of the conventional soft demapper. So this notebook works as an example of using pyAerial for model validation.

Finally, the model is exported into a format consumed by the TensorRT inference engine that is used for integrating the model into Aerial SDK for testing the model with real hardware in an over the air environment.

**Note 1:** This notebook requires that the Aerial test vectors have been generated. The test vector directory is set below in `AERIAL_TEST_VECTOR_DIR` variable. **Note 2:** This notebook also requires that the notebook example on LLRNet dataset generation has been run first.

[1]: # Check platform.
    import platform
    if platform.machine() != 'x86_64':
        raise SystemExit("Unsupported platform!")
3.3.5.1.1 Imports

```python
[2]: %matplotlib widget
import os
os.environ['CUDA_VISIBLE_DEVICES'] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3"  # Silence TensorFlow.
os.environ['CUDA_MODULE_LOADING'] = "LAZY"

import cuda
import h5Spy as h5
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers
import tf2onnx
import onnx
from IPython.display import Markdown
from IPython.display import display

# PyAerial components
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.algorithms import TrtEngine
from aerial.phy5g.algorithms import TrtTensorPrms
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.util.cuda import get_cuda_stream
from aerial.util.data import load_pickle
from aerial.util.fapi import dmrs_fapi_to_bit_array

# Configure the notebook to use only a single GPU and allocate only as much memory as needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

[3]:
```
tb_errors = dict(aerial=dict(), llrnet=dict(), logmap=dict())
tb_count = dict(aerial=dict(), llrnet=dict(), logmap=dict())
```

[4]:
```
# Dataset root directory.
DATA_DIR = "data/"

# Aerial test vector directory.
AERIAL_TEST_VECTOR_DIR = "/mnt/cicd_tvs/develop/GPU_test_input/

# LLRNet dataset directory.
dataset_dir = DATA_DIR + "example_llrnet_dataset/QPSK/"

# LLRNet model target path
llrnet_onnx_file = f"../models/llrnet.onnx"
llrnet_trt_file = f"../models/llrnet.trt"
```

(continues on next page)
# Training vs. testing SNR. Assume these exist in the dataset.

\[
\text{train}_{\text{snr}} = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5]
\]

\[
\text{test}_{\text{snr}} = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5]
\]

# Training, validation and test split in percentages if the same SNR is used for
# training and testing.

\[
\text{train}_{\text{split}} = 45
\]

\[
\text{val}_{\text{split}} = 5
\]

\[
\text{test}_{\text{split}} = 50
\]

# Training hyperparameters.

\[
\text{batch}_{\text{size}} = 32
\]

\[
\text{epochs} = 5
\]

\[
\text{step} = \text{tf.Variable}(0, \text{ trainable}=\text{False})
\]

\[
\text{boundaries} = [350000, 450000]
\]

\[
\text{values} = [5e-4, 1e-4, 1e-5]
\]

# values = [0.05, 0.01, 0.001]

\[
\text{learning}_{\text{rate}}_{\text{fn}} = \text{tf.keras.optimizers.schedules.PiecewiseConstantDecay}(
\text{boundaries},
\text{values})
\]

\[
\text{optimizer} = \text{tf.keras.optimizers.Adam}(\text{learning}_{\text{rate}}=\text{learning}_{\text{rate}}_{\text{fn}}, \text{ weight}_{\text{decay}}=1e-4)
\]

# optimizer = tf.keras.optimizers.experimental.SGD(learning_rate=0.05, weight_decay=1e-4, momentum=0.9)

# Modulation order. LLRNet needs to be trained separately for each modulation order.

\[
\text{mod}_{\text{order}} = 2
\]

3.3.5.1.2 Define the LLRNet model

The LLRNet model follows the original paper


and is a very simple MLP model with a single hidden layer. It takes the equalized symbols in its input with the real and imaginary parts separated, and outputs soft bits (log-likelihood ratios) that can be further fed into LDPC (de)rate matching and decoding.

```python
[5]:
model = keras.Sequential(    
    [layers.Dense(16, input_dim=2, activation="relu"),
     layers.Dense(8, activation="linear")
    ]
)
def loss(llr, predictions):
    mae = tf.abs(predictions[:, :mod_order] - llr)
    mse = tf.reduce_mean(tf.square(mae))
    return mse
```
### 3.3.5.1.3 Training, validation and testing datasets

Here, the dataset gets loaded and split into training, validation and testing datasets, as well as put in the right format for the model.

```python
[6]: # Load the main data file
    try:
        df = pd.read_parquet(dataset_dir + "l2_metadata.parquet", engine="pyarrow")
    except FileNotFoundError:
        display(Markdown("**Data not found - has llrnet_dataset_generation.ipynb been run? **
                        
# Query the entries for the selected modulation order.
    df = df[df["qamModOrder"] == mod_order]

# Collect the dataset by SNR.
    llrs = dict()
    eq_syms = dict()
    indices = dict()
    for pusch_record in df.itertuples():
        user_data_filename = dataset_dir + pusch_record.user_data_filename
        user_data = load_pickle(user_data_filename)

        if user_data["snr"] not in llrs.keys():
            llrs[user_data["snr"]] = []
            eq_syms[user_data["snr"]]) = []
            indices[user_data["snr"]]) = []

                llrs[user_data["snr"]].append(user_data["map_llrs"])
                eq_syms[user_data["snr"]]) = append(user_data["eq_sym"])]

                indices[user_data["snr"]]) = append(pusch_record.Index)

    llr_train, llr_val = [], []
    sym_train, sym_val = [], []
    testIndices = []
    for key in llrs.keys():
        llrs[key] = np.stack(llrs[key])
        eq_sym[key] = np.stack(eq_sym[key])

        # Randomize the order.
        permutation = np.arange(llrs[key].shape[0])
        np.random.shuffle(permutation)
        llrs[key] = llrs[key][permutation, ...]
        eq_sym[key] = eq_sym[key][permutation, ...]
        indices[key] = list(np.array(indices[key])[permutation])

        # Separate real and imaginary parts of the symbols.
        eq_sym[key] = np.stack((np.real(eq_sym[key]), np.imag(eq_sym[key])))

        num_slots = llrs[key].shape[0]
        if key in train_snr and key in test_snr:
            num_train_slots = int(np.round(train_split / 100 * num_slots))
            num_val_slots = int(np.round(val_split / 100 * num_slots))
            num_test_slots = int(np.round(test_split / 100 * num_slots))
        elif key in train_snr:
            num_train_slots = int(np.round(train_split / (train_split + val_split) * num_slots))
```

(continues on next page)
num_val_slots = int(np.round(val_split / (train_split + val_split) * num_slots))
num_test_slots = 0
elif key in test_snr:
    num_train_slots = 0
    num_val_slots = 0
    num_test_slots = num_slots
else:
    num_train_slots = 0
    num_val_slots = 0
    num_test_slots = 0

# Collect training/validation/testing sets.
llr_train.append(llrs[key][..., num_train_slots])
llr_val.append(llrs[key][..., num_train_slots + num_val_slots])
sym_train.append(eq_syms[key][..., num_train_slots])
sym_val.append(eq_syms[key][..., num_train_slots + num_val_slots])

# Just indices for the test set.
test_indices += indices[key][..., num_train_slots + num_val_slots + num_test_slots]

llr_train = np.transpose(np.concatenate(llr_train, axis=0), (1, 0, 2))
llr_val = np.transpose(np.concatenate(llr_val, axis=0), (1, 0, 2))
sym_train = np.concatenate(sym_train, axis=1)
sym_val = np.concatenate(sym_val, axis=1)

# Fetch the total number of slots in each set.
num_train_slots = llr_train.shape[1]
num_val_slots = llr_val.shape[1]
num_test_slots = len(test_indices)

normalizer = 1.0 / np.sqrt(np.var(llr_train))
llr_train = llr_train / normalizer
llr_val = llr_val / normalizer

# Reshape into samples x mod_order array.
llr_train = llr_train.reshape(mod_order, -1).T
llr_val = llr_val.reshape(mod_order, -1).T
# Reshape into samples x 2 array.
sym_train = sym_train.reshape(2, -1).T
sym_val = sym_val.reshape(2, -1).T

print(f"Total number of slots in the training set: {num_train_slots}")
print(f"Total number of slots in the validation set: {num_val_slots}")
print(f"Total number of slots in the test set: {num_test_slots}")

Total number of slots in the training set: 5400
Total number of slots in the validation set: 600
Total number of slots in the test set: 6000
3.3.5.1.4 Model training and validation

Model training is done using Keras here.

```python
[7]: print("Training...")
model.compile(loss=loss, optimizer=optimizer, metrics=[loss])
model.fit(
    x=sym_train,
    y=llr_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(sym_val, llr_val),
    shuffle=True
)
```

Training...
Epoch 1/5
```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1718218041.098979 27464 device_compiler.h:186] Compiled cluster using _XLA! This line is logged at most once for the lifetime of the process.
552825/552825 [==============================] - 764s 1ms/step - loss: 84.4521 - val_loss: 81.8378
Epoch 2/5
552825/552825 [==============================] - 768s 1ms/step - loss: 81.9108 - val_loss: 81.7775
Epoch 3/5
552825/552825 [==============================] - 766s 1ms/step - loss: 81.8526 - val_loss: 81.7239
Epoch 4/5
552825/552825 [==============================] - 767s 1ms/step - loss: 81.7989 - val_loss: 81.6738
Epoch 5/5
552825/552825 [==============================] - 765s 1ms/step - loss: 81.7505 - val_loss: 81.6269
```

[7]: <keras.src.callbacks.History at 0x7f31227f2170>

3.3.5.1.5 Export to TensorRT

Finally, the model gets exported to ONNX format. The ONNX format needs to be converted to TRT engine format to be consumed by the TensorRT inference engine, this is done here using the command line tool trtexec.

```python
[8]: input_signature = [tf.TensorSpec([None, 2], tf.float32, name="input")]
onnx_model, _ = tf2onnx.convert.from_keras(model, input_signature)
onnx.save(onnx_model, llrnet_onnx_file)
print("ONNX model created. Converting to TRT engine file...")
command = f"trtexec " + \n    f"--onnx={llrnet_onnx_file} " + \n    f"--saveEngine={llrnet_trt_file} " + \n    f"--skipInference " + \n    f"--minShapes=input:1x2 " + \n    f"--optShapes=input:42588x2 " + \n    f"--maxShapes=input:85176x2 " + \n```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

```python
f"--inputIOFormats=fp32:chw " + \\
f"--outputIOFormats=fp32:chw" + \\
f"> /dev/null"
return_val = os.system(command)
if return_val == 0:
    print("TRT engine model created.")
else:
    raise SystemExit("Failed to create the TRT engine file!")
```

ONNX model created. Converting to TRT engine file...

TRT engine model created.

3.3.5.1.6 Define a PUSCH receiver chain using pyAerial

This class encapsulates the whole PUSCH receiver chain. The components include channel estimation, noise and interference estimation, channel equalization and soft demapping, LDPC (de)rate matching and LDPC decoding. The receiver outputs the received transport block in bytes.

The soft demapping part can be replaced by LLRNet.

[9]: class Receiver:
    
    """PUSCH receiver class.
    This class encapsulates the whole PUSCH receiver chain built using pyAerial components.""
    
    def __init__(self,
              llrnet_model_file,
              num_rx_ant,
              enable_pusch_tdi,
              eq_coeff_algo):
        """Initialize the PUSCH receiver.""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver.
        self.channel_estimator = ChannelEstimator(  
            num_rx_ant=num_rx_ant,
            cuda_stream=self.cuda_stream
        )

        self.channel_equalizer = ChannelEqualizer(  
            num_rx_ant=num_rx_ant,
            enable_pusch_tdi=enable_pusch_tdi,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream
        )

        self.noise_intf_estimator = NoiseIntfEstimator(  
            num_rx_ant=num_rx_ant,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream
        )

        self.demapper = Demapper(mod_order=mod_order)
        self.trt_engine = TrtEngine(  
            llrnet_model_file,
            max_batch_size=85176,
        )
```

(continues on next page)
input_tensors=[TrtTensorPrms('input', (2,), np.float32)],
output_tensors=[TrtTensorPrms('dense_1', (8,), np.float32)]
)

self.derate_match = LdpcDeRateMatch(
enable_scrambling=True,
cuda_stream=self.cuda_stream
)
self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)
self.llr_method = "llrnet"

def set_llr_method(self, method):
    """Set the used LLR computation method.
    Args:
        method (str): Either "aerial" meaning the conventional log-likelihood
                    ratio computation, or "llrnet" for using LLRNet instead.
    """
    if method not in ["aerial", "logmap", "llrnet"]:
        raise ValueError("Invalid LLR computation method!")
    self.llr_method = method

def run(
    self,
    rx_slot,
    num_ues,
    slot,
    num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id,
    start_prb,
    num_prbs,
    dmrs_sym,
    dmrs_max_len,
    dmrs_add_ln_pos,
    start_sym,
    num_symbols,
    scids,
    layers,
    dmrs_ports,
    rntis,
    data_scids,
    code_rates,
    mod_orders,
    tb_sizes,
    rvs,
    ndis):
    """Run the receiver.""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
num_prbs=num_prbs,
dmrs_syms=dmrs_syms,
dmrs_max_len=dmrs_max_len,
dmrs_add_ln_pos=dmrs_add_ln_pos,
start_sym=start_sym,
num_symbols=num_symbols,
scids=scids,
layers=layers,
dmrs_ports=dmrs_ports
)

# Noise and interference estimation.
lw_inv, noise_var_pre_eq = self.noise_intf_estimator.estimate(
    rx_slot=rx_slot,
    channel_est=ch_est,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    scids=scids,
    layers=layers,
    dmrs_ports=dmrs_ports
)

# Channel equalization and soft demapping. Note that the cuPHY kernel actually computes both
# the equalized symbols and the LLRs.
llr, eq_sym = self.channel_equalizer.equalize(
    rx_slot=rx_slot,
    channel_est=ch_est,
    lw_inv=lw_inv,
    noise_var_pre_eq=noise_var_pre_eq,
    num_ues=num_ues,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    layers=layers,
    mod_orders=mod_orders
)

# Use the LLRNet model here to get the log-likelihood ratios.
num_data_sym = (np.array(dmrs_syms[start_sym:start_sym + num_symbols]) == 0).sum()

if self.llr_method == "llrnet":
# Put the input in the right format.
eq_sym_input = np.stack((np.real(eq_sym[0]), np.imag(eq_sym[0]))).
→ reshape(2, -1).

# Run the model.
llr_output = self.trt_engine.run({"input": eq_sym_input})["dense_1"]

# Reshape the output in the right format for the LDPC decoding process.
llr_output = np.array(llr_output)[..., :mod_orders[0], layers[0], num_prbs * 12, num_data_sym].reshape(mod_... orders[0], layers[0], num_prbs * 12, num_data_sym)
llr_output *= normalizer

elif self.llr_method == "aerial":
    llr_output = llr[0]

elif self.llr_method == "logmap":
    inv_noise_var_lin = self.channel_equalizer.ree_diag_inv[0]
    llr_output = self.demapper.demap(eq_sym[0], inv_noise_var_lin[...], None)

# De-rate matching and descrambling.
cinit = (rntis[0] << 15) + data_scids[0]
rate_match_len = num_data_sym * mod_orders[0] * num_prbs * 12 * layers[0]
coded_blocks = self.derate_match.derate_match(  
    input_llrs=[llr_output],  
    tb_sizes=tb_sizes,  
    code_rates=code_rates,  
    rate_match_lengths=[rate_match_len],  
    mod_orders=mod_orders,  
    num_layers=layers,  
    redundancy_versions=rvs,  
    ndis=ndis,  
    cinits=[cinit]  
)

# LDPC decoding of the derate matched blocks.
code_blocks = self.decoder.decode(  
    input_llrs=coded_blocks,  
    tb_sizes=tb_sizes,  
    code_rates=code_rates,  
    redundancy_versions=rvs,  
    rate_match_lengths=[rate_match_len]  
)[0]

# Combine the code blocks into a transport block.
tb = code_block_desegment(  
    code_blocks=coded_blocks,  
    tb_size=tb_sizes[0],  
    code_rate=code_rates[0],  
    return_bits=False  
)

return tb
3.3.5.1.7 Model testing on Aerial test vectors

```python
[10]: if mod_order == 2:
    test_vector_filename = "TVnr_7201_PUSCH_gNB_CUPHY_s0p0.h5"
elif mod_order == 4:
    test_vector_filename = "TVnr_7916_PUSCH_gNB_CUPHY_s0p0.h5"
elif mod_order == 6:
    test_vector_filename = "TVnr_7203_PUSCH_gNB_CUPHY_s0p0.h5"
filename = AERIAL_TEST_VECTOR_DIR + test_vector_filename
input_file = h5.File(filename, "r")
num_rx_ant = input_file["gnb_pars"]["nRx"]
enable_pusch_tdi = input_file["gnb_pars"]["TdiMode"]
eq_coef_algo = input_file["gnb_pars"]["eqCoeffAlgoIdx"]
receiver = Receiver(
    l1rnet_trt_file,
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coef_algo=eq_coef_algo
)

# Extract the test vector data and parameters.
rx_slot = np.array(input_file["DataRx"])["re"] + 1j * np.array(input_file["DataRx"])["im"]
rx_slot = rx_slot.transpose(2, 1, 0)
num_ues = input_file["ueGrp_pars"]["nUes"]
start_prb = input_file["ueGrp_pars"]["startPrb"]
num_prbs = input_file["ueGrp_pars"]["nPrb"]
start_sym = input_file["ueGrp_pars"]["StartSymbolIndex"]
num_symbols = input_file["ueGrp_pars"]["NrOfSymbols"]
dmrs_sym_loc_bmsk = input_file["ueGrp_pars"]["dmrsSymLocBmsk"]
dmrs_scram_id = input_file["tb_pars"]["dmrsScramId"]
dmrs_max_len = input_file["tb_pars"]["dmrsMaxLength"]
dmrs_add_ln_pos = input_file["tb_pars"]["dmrsAddlPosition"]
num_dmrs_cdm_grps_no_data = input_file["tb_pars"]["numDmrsCdmGrpsNoData"]
mod_orders = input_file["tb_pars"]["qamModOrder"]
layers = input_file["tb_pars"]["numLayers"]
scids = input_file["tb_pars"]["nSCID"]
dmrs_ports = input_file["tb_pars"]["dmrsPortBmsk"]
slot = np.array(input_file["gnb_pars"]["slotNumber"])
tb_sizes = 8 * input_file["tb_pars"]["nTbByte"]
code_rates = input_file["tb_pars"]["targetCodeRate"] / 1024
rsv = input_file["tb_pars"]["rsv"]
ndis = input_file["tb_pars"]["ndi"]
rntis = input_file["tb_pars"]["nRnti"]
data_scids = input_file["tb_pars"]["dataScramId"]
dmrs_syms = dmrs_fapi_to_bit_array(dmrs_sym_loc_bmsk)

# Run the receiver with the test vector parameters.
receiver.set_1llr_method("1l1rnet")
tb = receiver.run(
    rx_slot=rx_slot,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
(continues on next page)
dmrs_scrm_id=dmrs_scrm_id,
start_prb=start_prb,
num_prbs=num_prbs,
dmrs_syms=dmrs_syms,
dmrs_max_len=dmrs_max_len,
dmrs_add_ln_pos=dmrs_add_ln_pos,
start_sym=start_sym,
um_symbols=num_symbols,
scids=scids,
layers=layers,
dmrs_ports=dmrs_ports,
rntis=rntis,
data_scids=data_scids,
code_rates=code_rates,
mod_orders=mod_orders,
tb_sizes=tb_sizes,
rvs=rvs,
ndis=ndis
)

# Check that the received TB matches with the transmitted one.
if np.array_equal(np.array(input_file['tb_data'])[:tb_sizes[0]//8, 0], tb[:tb_sizes[0]//8]):
    print("CRC check passed!")
else:
    print("CRC check failed!")

CRC check passed!

3.3.5.1.8 Model testing on synthetic/simulated data

[11]: for pushch_record in df.take(test_indices).itertuples(index=False):
    user_data_filename = dataset_dir + pushch_record.user_data_filename
    user_data = load_pickle(user_data_filename)
    snr = user_data['snr']
    rx_iq_data_filename = dataset_dir + pushch_record.rx_iq_data_filename
    rx_slot = load_pickle(rx_iq_data_filename)
    num_ues = 1
    start_prb = pushch_record.rbStart
    num_prbs = pushch_record.rbSize
    start_sym = pushch_record.StartSymbolIndex
    num_symbols = pushch_record.NrOfSymbols
    dmrs_syms = dmrs_fapi_to_bit_array(pusch_record.ulDmrsSymbPos)
    dmrs_scrm_id = pushch_record.ulDmrsScramblingId
    dmrs_max_len = 1
    dmrs_add_ln_pos = 1
    num_dmrs_cdm_grps_no_data = pushch_record.numDmrsCdmGrpsNoData
    layers = [pusch_record.nrOfLayers]
    scids = [pusch_record.SCID]
    dmrs_ports = [pusch_record.dmrsPorts]
    slot = pushch_record.Slot
    tb_sizes = [len(pusch_record.macPdu)]

(continues on next page)
mod_orders = [pusch_record.qamModOrder]
code_rates = [pusch_record.targetCodeRate / 10240.]
rvs = [0]
ndis = [1]
rntis = [pusch_record.RNTI]
data_scids = [pusch_record.dataScramblingId]
ref_tb = pusch_record.macPdu

for llr_method in ["aerial", "llrnet", "logmap"]:
    if snr not in tb_errors[llr_method].keys():
        tb_errors[llr_method][snr] = 0
        tb_count[llr_method][snr] = 0

        receiver.set_llr_method(llr_method)
        tb = receiver.run(
            rx_slot=rx_slot,
            num_ues=num_ues,
            slot=slot,
            num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
            dmrs_scrm_id=dmrs_scrm_id,
            start_prb=start_prb,
            num_prbs=num_prbs,
            dmrs_syms=dmrs_syms,
            dmrs_max_len=dmrs_max_len,
            dmrs_add_ln_pos=dmrs_add_ln_pos,
            start_sym=start_sym,
            num_symbols=num_symbols,
            scids=scids,
            layers=layers,
            dmrs_ports=dmrs_ports,
            rntis=rntis,
            data_scids=data_scids,
            code_rates=code_rates,
            mod_orders=mod_orders,
            tb_sizes=[tb_sizes[0] * 8],
            rvs=rvs,
            ndis=ndis
        )
        tb_count[llr_method][snr] += 1
        tb_errors[llr_method][snr] += (not np.array_equal(tb[:tb_sizes[0]], ref_tb[:
                     -tb_sizes[0]])

    esno_dbs = tb_count["aerial"].keys()
    bler = dict(aerial=[], llrnet=[], logmap=[])
    for esno_db in esno_dbs:
        bler["aerial"].append(tb_errors["aerial"][esno_db] / tb_count["aerial"][esno_db])
        bler["llrnet"].append(tb_errors["llrnet"][esno_db] / tb_count["llrnet"][esno_db])
        bler["logmap"].append(tb_errors["logmap"][esno_db] / tb_count["logmap"][esno_db])

    fig = plt.figure(figsize=(10, 10))
    plt.yscale('log')
    plt.ylim(0.01, 1)
    plt.xlim(np.min(esno_dbs), np.max(esno_dbs))
plt.title("BLER Performance vs. Es/No")
plt.ylabel("BLER")
plt.xlabel("Es/No [dB]")
plt.grid()
plt.plot(esno_dbs, bler['aerial'], marker="d", linestyle="-", color="blue", markersize=8)
plt.plot(esno_dbs, bler['llrnet'], marker="s", linestyle="-", color="black", markersize=8)
plt.plot(esno_dbs, bler['logmap'], marker="o", linestyle="-", color="red", markersize=8)
plt.legend(["Aerial", "LLRNet", "Log-MAP"])

3.3. Examples of Using pyAerial
3.3.6. Neural receiver validation

In this example, a trained neural network-based PUSCH receiver is validated using pyAerial. The model is based on the following paper:


The neural receiver is compared against the conventional PUSCH receiver using pyAerial. For running inference, we use pyAerial’s bindings to cuPHY’s TensorRT wrappers.

3.3.6.1 Using pyAerial to evaluate a PUSCH neural receiver

This example shows how to use the pyAerial cuPHY Python bindings to evaluate a trained neural network-based PUSCH receiver. In this example, the neural network is used to replace channel estimation, noise and interference estimation and channel equalization, and thus outputs log-likelihood ratios directly. The model is a variant of what has been proposed in


The rest of the PUSCH receiver pipeline following the neural receiver, meaning LDPC decoding chain, is modeled using pyAerial. Also, the neural receiver takes LS channel estimates as inputs in addition to the received PUSCH slot. These are also obtained using pyAerial. The neural receiver-based PUSCH receiver is compared against the conventional PUSCH receiver, which is built using pyAerial’s (fully fused) PUSCH pipeline.

PUSCH transmitter is emulated by PDSCH transmission with properly chosen parameters, that way making it a 5G NR compliant PUSCH transmission. The NVIDIA Sionna library is utilized for simulating the radio channel based on 3GPP channel models.

```
[1]: # Check platform.
    import platform
    if platform.machine() != 'x86_64':
        raise SystemExit("Unsupported platform!")

3.3.6.1.1 Imports

[2]: %matplotlib widget
    import datetime
    from collections import defaultdict
    import os
    os.environ['CUDA_VISIBLE_DEVICES'] = "0"
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3"  # Silence TensorFlow.
    os.environ['CUDA_MODULE_LOADING'] = "LAZY"

    import numpy as np
    import matplotlib.pyplot as plt
    import sionna
    import tensorflow as tf

    from aerial.phy5g.pdsch import PdschTx
    from aerial.phy5g.pusch import PuschRx
```

(continues on next page)
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import TrtEngine
from aerial.phy5g.algorithms import get_mcs
from aerial.phy5g.ldpc import random_tb
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.types import PuschLdpcKernelLaunch
from aerial.util.cuda import get_cuda_stream

# Configure the notebook to use only a single GPU and allocate only as much memory as needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)

3.3.6.1.2 Parameters

Set simulation parameters, numerology, PUSCH parameters and channel parameters here.

```python
# Simulation parameters.
esno_db_range = np.arange(-4, -2.8, 0.2)
num_slots = 10000
min_num_pb_errors = 250

# Numerology and frame structure. See TS 38.211.
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20

num_tx_ant = 1  # UE antennas
num_rx_ant = 4  # gNB antennas
cell_id = 41    # Physical cell ID
enable_pusch_tdi = 1  # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1  # Equalizer algorithm

# PUSCH parameters
rnti = 1234      # UE RNTI
scid = 0         # DMRS scrambling ID
data_scid = 0    # Data scrambling ID
layers = 1       # Number of layers
mcs = 7          # MCS index as per TS 38.214 table
dmrs_port = 1    # Used DMRS port.
start_prb = 0    # Start PRB index.
num_prbs = 273   # Number of allocated PRBs.
start_sym = 0    # Start symbol index.
num_symbols = 12 # Number of symbols.
dmrs_scrm_id = 41 # DMRS scrambling ID
dmrs_position = [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0] # Indicates which symbols are used for DMRS.
dmrs_max_len = 1
```

(continues on next page)
dmrs_add_ln_pos = 2
num_dmrs_cdm_grps_no_data = 2

# Channel parameters
carrier_frequency = 3.5e9  # Carrier frequency in Hz.
delay_spread = 100e-9      # Nominal delay spread in [s]. Please see the CDL documentation
                          # about how to choose this value.
link_direction = "uplink"  # Channel model: Suitable values:
channel_model = "Rayleigh"  # "Rayleigh" - Rayleigh block fading channel model (sionna.
                            # 3GPP CDL channel models as per TR 38.901.
speed = 0.8333            # UE speed [m/s]. The direction of travel will chosen randomly within the x-y plane.

3.3.6.1.3 Create the model file for the TRT engine

The TRT engine is built based on TensorRT plan files which are not portable across different platforms. Hence the plan file is created here from a supplied ONNX file.

```python
MODEL_DIR = "../models"
nrx_onnx_file = f"{MODEL_DIR}/neural_rx.onnx"
nrx_trt_file = f"{MODEL_DIR}/neural_rx.trt"
command = f"trtexec " + \
          f"--onnx={nrx_onnx_file} " + \
          f"--saveEngine={nrx_trt_file} " + \
          f"--skipInference " + \
          f"--inputIOFormats=fp32:chw,fp32:chw,fp32:chw,fp32:chw,fp32:chw,int32:chw,int32:chw " + \
          f"--outputIOFormats=fp32:chw,fp32:chw " + \
          f"--shapes=rx_slot_real:1x3276x12x4,rx_slot_imag:1x3276x12x4,h_hat_real:1x4914x1x4,h_hat_imag:1x4914x1x4 " + \
          f"> /dev/null"
return_val = os.system(command)
if return_val == 0:
    print("TRT engine model created.")
else:
    raise SystemExit("Failed to create the TRT engine file!")

TRT engine model created.
```
3.3.6.1.4 Create the PUSCH pipelines

As mentioned, PUSCH transmission is emulated here by the PDSCH transmission chain. Note that the static cell parameters and static PUSCH parameters are given upon creating the PUSCH transmission/reception objects. Dynamically (per slot) changing parameters are however set when actually running the transmission/reception, see further below.

```python
pusch_tx = PdschTx(  
    cell_id=cell_id,  
    num_rx_ant=num_tx_ant,  
    num_tx_ant=num_tx_ant,
)

# This is the fully fused PUSCH receiver chain.
pusch_rx = PuschRx(  
    cell_id=cell_id,  
    num_rx_ant=num_rx_ant,  
    num_tx_ant=num_rx_ant,  
    enable_pusch_tdi=enable_pusch_tdi,  
    eq_coeff_algo=eq_coeff_algo,  
    # To make this equal separate PUSCH Rx components configuration:  
    ldpc_kernel_launch=PuschLdpcKernelLaunch.PUSCH_RX_LDPC_STREAM_SEQUENTIAL
)

class NeuralRx:
    '''PUSCH neural receiver class.
    This class encapsulates the PUSCH neural receiver chain built using
    pyAerial components.''

    def __init__(self,  
        num_rx_ant,  
        enable_pusch_tdi,  
        eq_coeff_algo):
        '''Initialize the neural receiver.''
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver. The channel estimator outputs just the
        # LS
        # channel estimates.
        self.channel_estimator = ChannelEstimator(  
            num_rx_ant=num_rx_ant,  
            ch_est_algo=3,  
            # This is LS channel estimation.
            cuda_stream=self.cuda_stream
        )

        # Create the pyAerial TRT engine object. This wraps TensorRT and links it
        # together
        # with the rest of cuPHY. Here pyAerial's Python bindings to the engine are used
        # to run inference with the neural receiver model.
        # The inputs of the neural receiver are:
        # - LS channel estimates
        # - The Rx slot
        # - Active DMRS ports (active layers out of the layers that the neural receiver
        # supporting)
        # - DMRS OFDM symbol locations (indices)
```

(continues on next page)
### - DMRS subcarrier positions within a PRB (indices)
# Note that the shapes are given without batch size.

```python
self.trt_engine = TrtEngine(
    trt_model_file="../models/neural_rx.trt",
    max_batch_size=1,
    input_tensors=[TrtTensorPrms('rx_slot_real', (3276, 12, 4), np.float32),
                  TrtTensorPrms('rx_slot_imag', (3276, 12, 4), np.float32),
                  TrtTensorPrms('h_hat_real', (4914, 1, 4), np.float32),
                  TrtTensorPrms('h_hat_imag', (4914, 1, 4), np.float32),
                  TrtTensorPrms('active_dmrs_ports', (1,), np.float32),
                  TrtTensorPrms('dmrs_ofdm_pos', (3,), np.int32),
                  TrtTensorPrms('dmrs_subcarrier_pos', (6,), np.int32)],
    output_tensors=[TrtTensorPrms('output_1', (8, 1, 3276, 12), np.float32),
                    TrtTensorPrms('output_2', (1, 3276, 12, 8), np.float32)]
)
```

# LDPC (de)rate matching and decoding implemented using pyAerial.

```python
self.de_rate_match = LdpcDeRateMatch(
    enable_scrambling=True,
    cuda_stream=self.cuda_stream
)
```

```python
def run(
    self,
    rx_slot,
    num_ues,
    slot,
    num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id,
    start_prb,
    num_prbs,
    dmrs_syms,
    dmrs_max_len,
    dmrs_add_ln_pos,
    start_sym,
    num_symbols,
    scids,
    layers,
    dmrs_ports,
    rntis,
    data_scids,
    code_rates,
    mod_orders,
    tb_sizes
):
    """Run the receiver."""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
```

(continues on next page)
# This is the neural receiver part.
# It outputs the LLRs for all symbols.

```python
# This is the neural receiver part.
# It outputs the LLRs for all symbols.
dmrs_ofdm_pos = np.where(np.array(dmrs_position))[0].astype(np.int32)
dmrs_ofdm_pos = dmrs_ofdm_pos[None, ...]
dmrs_subcarrier_pos = np.array([[0, 2, 4, 6, 8, 10]], dtype=np.int32)
rx_slot_in = rx_slot[None, :, start_sym:start_sym+num_symbols, :]
ch_est_in = np.transpose(ch_est[0], (0, 3, 1, 2)).reshape(ch_est[0].shape[0] - ch_est[0].shape[1], ch_est[0].shape[2])
ch_est_in = ch_est_in[None, ...]
input_tensors = {
    "rx_slot_real": np.real(rx_slot_in).astype(np.float32),
    "rx_slot_imag": np.imag(rx_slot_in).astype(np.float32),
    "h_hat_real": np.real(ch_est_in).astype(np.float32),
    "h_hat_imag": np.imag(ch_est_in).astype(np.float32),
    "active_dmrs_ports": active_dmrs_ports.astype(np.float32),
    "dmrs_ofdm_pos": dmrs_ofdm_pos.astype(np.int32),
    "dmrs_subcarrier_pos": dmrs_subcarrier_pos.astype(np.int32)
}
```

data_syms = np.array(dmrs_syms[start_sym:start_sym+num_symbols]) == 0
num_data_sym = data_syms.sum()
outputs = self.trt_engine.run(input_tensors)

# The neural receiver outputs some values also for DMRS symbols, remove those from the output.
llrs = np.take(outputs["output_1"][:, ...], np.where(data_syms)[0], axis=3)

```python
cinits = [(rntis[ue] << 15) + data_scids[ue] for ue in range(num_ues)]
rate_match_lengths = [num_data_sym * mod_orders[ue] * num_prbs * 12 * layers[ue] for ue in range(num_ues)]
tb_sizes = [s * 8 for s in tb_sizes]
code_rates = [c / 1024. for c in code_rates]
rvs = [0,] * num_ues
ndis = [1,] * num_ues
coded_blocks = self.derate_match.derate_match(
    input_llrs=[llrs],
    tb_sizes=tb_sizes,
    code_rates=code_rates,
    rate_match_lengths=rate_match_lengths,
    mod_orders=mod_orders,
    num_layers=layers,
    redundancy_versions=rvs,
    ndis=ndis,
    cinits=cinits
)
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

code_blocks = self.decoder.decode(
    input_llrs=coded_blocks,
    tb_sizes=tb_sizes,
    code_rates=code_rates,
    redundancy_versions=rvs,
    rate_match_lengths=rate_match_lengths
)

# TODO: Use the CRC kernel here.
decoded_tbs = []
for ue_idx in range(num_ues):
    # Combine the code blocks into a transport block.
    tb = code_block_desegment(
        code_blocks=coded_blocks[ue_idx],
        tb_size=tb_sizes[ue_idx],
        code_rate=code_rates[ue_idx],
        return_bits=False,
    )
    
    # Remove CRC - no checking, check TBs/bits directly.
    tb = tb[:-3]
    decoded_tbs.append(tb)

return decoded_tbs

neural_rx = NeuralRx(
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo
)

3.3.6.1.5 Channel generation using Sionna

Simulating the transmission through the radio channel takes advantage of the channel model implementations available in NVIDIA Sionna. In Sionna, the transmission can be simulated directly in frequency domain by defining a resource grid. In our case, reference signal patterns and data carrying resource elements are defined elsewhere within the Aerial code, hence we define resource grid as a simple dummy grid containing only data symbols.

See also: Sionna documentation

[6]: # Define the resource grid.
resource_grid = sionna.ofdm.ResourceGrid(
    num_ofdm_symbols=num_ofdm_symbols,
    fft_size=fft_size,
    subcarrier_spacing=subcarrier_spacing,
    num_tx=1,
    num_streams_per_tx=1,
    cyclic_prefix_length=cyclic_prefix_length,
    num_guard_carriers=num_guard_subcarriers,
    dc_null=False,
    pilot_pattern=None,
)
pilot_ofdm_symbol_indices=None
)
resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_grid)

# Define the antenna arrays.
ue_array = sionna.channel.tr38901.Antenna(
    polarization="single",
    polarization_type="V",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)

gnb_array = sionna.channel.tr38901.AntennaArray(
    num_rows=1,
    num_cols=int(num_rx_ant/2),
    polarization="dual",
    polarization_type="cross",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)

if channel_model == "Rayleigh":
    ch_model = sionna.channel.RayleighBlockFading(
        num_rx=1,
        num_rx_ant=num_rx_ant,
        num_tx=1,
        num_tx_ant=num_tx_ant
    )

elif "CDL" in channel_model:
    cdl_model = channel_model[-1]

    # Configure a channel impulse response (CIR) generator for the CDL model.
    ch_model = sionna.channel.tr38901.CDL(
        cdl_model,
        delay_spread,
        carrier_frequency,
        ue_array,
        gnb_array,
        link_direction,
        min_speed=speed
    )
else:
    raise ValueError(f"Invalid channel model {channel_model}!")

channel = sionna.channel.OFDMChannel(
    ch_model,
    resource_grid,
    add_awgn=True,
    normalize_channel=True,
    return_channel=False
)

def apply_channel(tx_tensor, No):
    """Transmit the Tx tensor through the radio channel.""
    # Add batch and num_tx dimensions that Sionna expects and reshape.
Aerial CUDA-Accelerated RAN, Release 24-2

(tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
  tx_tensor = tf.reshape(tx_tensor, (1, -1)))[None, None]
  tx_tensor = resource_grid_mapper(tx_tensor)
  rx_tensor = channel((tx_tensor, No))
  rx_tensor = remove_guard_subcarriers(rx_tensor)
  rx_tensor = rx_tensor[0, 0]
  rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
return rx_tensor

3.3.6.1.6 Helper class for simulation monitoring

This helper class plots the simulation results and shows simulation progress in a table.

[7]: class SimulationMonitor:
    """Helper class to show the progress and results of the simulation.""
    markers = ['d', 'o', 's']
    linestyles = ['-', '--', ':']
    colors = ['blue', 'black', 'red']
    def __init__(self, cases, esno_db_range):
        """Initialize the SimulationMonitor.
        Initialize the figure and the results table.
        ""
        self.cases = cases
        self.esno_db_range = esno_db_range
        self.current_esno_db_range = []
        self.start_time = None
        self.esno_db = None
        self.bler = defaultdict(list)
        self._print_headers()
    def step(self, esno_db):
        """Start next Es/No value.""
        self.start_time = datetime.datetime.now()
        self.esno_db = esno_db
        self.current_esno_db_range.append(esno_db)
    def update(self, num_tbs, num_tb_errors):
        """Update current state for the current Es/No value.""
        self._print_status(num_tbs, num_tb_errors, False)
    def _print_headers(self):
        """Print result table headers.""
        cases_str = " " * 21
        separator = " " * 21
        for case in self.cases:
            cases_str += case.center(20) + " "
            separator += "-" * 20 + " "
        print(cases_str)
p = print(separator)
title_str = "Es/No (dB)".rjust(12) + "TBs".rjust(8) + " "

(continues on next page)
for case in self.cases:
    title_str += "TB Errors".rjust(12) + "BLER".rjust(8) + ""
    title_str += "ms/TB".rjust(8)
print(title_str)
print(""" + """ + """ + """ + """") * len(self.cases) + """

def _print_status(self, num_tbs, num_tb_errors, finish):
    """Print simulation status in a table."""
    end_time = datetime.datetime.now()
t_delta = end_time - self.start_time
if finish:
    newline_char = '\n'
else:
    newline_char = '\r'
result_str = f"{self.esno_db:.9.2f}".rjust(12) + f"{num_tbs:.8d}".rjust(8) + ""
for case in self.cases:
    result_str += f"{num_tb_errors[case]:8d}".rjust(12)
    result_str += f"{num_tb_errors[case] / num_tbs:.4f}".rjust(8) + ""
    result_str += f"{t_delta.total_seconds() * 1000 / num_tbs:.6.1f}".rjust(8)
print(result_str, end=newline_char)

def finish_step(self, num_tbs, num_tb_errors):
    """Finish simulating the current Es/No value and add the result in the plot."""
    self._print_status(num_tbs, num_tb_errors, True)
for case_idx, case in enumerate(self.cases):
    self.bler[case].append(num_tb_errors[case] / num_tbs)

def finish(self):
    """Finish simulation and plot the results."""
    self.fig = plt.figure()
    for case_idx, case in enumerate(self.cases):
        plt.plot(
            self.current_esno_db_range,
            self.bler[case],
            marker=SimulationMonitor.markers[case_idx],
            linestyle=SimulationMonitor.linestyles[case_idx],
            color=SimulationMonitor.colors[case_idx],
            markersize=8,
            label=case
        )
    plt.yscale('log')
    plt.ylim(0.001, 1)
    plt.xlim(np.min(self.esno_db_range), np.max(self.esno_db_range))
    plt.title("Receiver BLER Performance vs. Es/No")
    plt.ylabel("BLER")
    plt.xlabel("Es/No [dB]")
    plt.grid()
    plt.legend()
    plt.show()
3.3.6.1.7 Run the actual simulation

Here we loop across the Es/No range, and simulate a number of slots for each Es/No value. A single transport block is simulated within a slot. The simulation starts over from the next Es/No value when a minimum number of transport block errors is reached.

```python
[8]: cases = ["PUSCH Rx", "Neural Rx"]
monitor = SimulationMonitor(cases, esno_db_range)

# Loop the Es/No range.
bler = []
for esno_db in esno_db_range:
    monitor.step(esno_db)
    num_tb_errors = defaultdict(int)

    # Run multiple slots and compute BLER.
    for slot_idx in range(num_slots):
        slot_number = slot_idx % num_slots_per_frame

        # Get modulation order and coderate.
        mod_order, coderate = get_mcs(mcs=mcs, table_idx=1)
        tb_input = random_tb(mod_order, coderate, dmrs_position, num_prbs, start_sym, __num_symbols, layers)

        # Transmit PUSCH. This is where we set the dynamically changing parameters.
        # Input parameters are given as lists as the interface supports multiple UEs.
        tx_tensor = pusch_tx.run(
            tb_inputs=[tb_input],            # Input transport block in bytes.
            num_ues=1,                       # We simulate only one UE here.
            slot=slot_number,               # Slot number.
            num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
            num_dmrs_cdms_no_data=num_dmrs_cdms_no_data,
            dmrs_scrm_id=dmrs_scrm_id,      # DMRS scrambling ID.
            start_prb=start_prb,           # Start PRB index.
            num_prbs=num_prbs,             # Number of allocated PRBs.
            num_prbs=num_prbs,             # Number of allocated PRBs.
            dmrs_syms=dmrs_position,       # List of binary numbers indicating which
        )

        # Channel transmission using TF and Sionna.
        No = pow(10., -esno_db / 10.)
        rx_tensor = apply_channel(tx_tensor, No)
        rx_tensor = np.array(rx_tensor)

        # Run the fused PUSCH receiver.
        # Note that this is where we set the dynamically changing parameters.
        tb_crcs, tbs = pusch_rx.run(
            rx_slot=rx_tensor,
            num_ues=1,
        )
```

(continues on next page)
slot=slot_number,
num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
dmrs_scrm_id=dmrs_scrm_id,
start_prb=start_prb,
num_prbs=num_prbs,
dmrs_syms=dmrs_position,
dmrs_max_len=dmrs_max_len,
dmrs_add_ln_pos=dmrs_add_ln_pos,
start_sym=start_sym,
um_symbols=num_symbols,
scids=[scid],
layers=[layers],
dmrs_ports=[dmrs_port],
rntis=[rnti],
data_scids=[data_scid],
code_rates=[coderate],
mod_orders=[mod_order],
tb_sizes=[len(tb_input)]
)
num_tb_errors["PUSCH Rx"] += int(np.array_equal(tbs[0][:-3], tb_input) == False)

# Run the neural receiver.
tbs = neural_rx.run(
    rx_slot=rx_tensor,
    num_ues=1,
    slot=slot_number,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
dmrs_scrm_id=dmrs_scrm_id,
start_prb=start_prb,
num_prbs=num_prbs,
dmrs_syms=dmrs_position,
dmrs_max_len=dmrs_max_len,
dmrs_add_ln_pos=dmrs_add_ln_pos,
start_sym=start_sym,
um_symbols=num_symbols,
scids=[scid],
layers=[layers],
dmrs_ports=[dmrs_port],
rntis=[rnti],
data_scids=[data_scid],
code_rates=[coderate],
mod_orders=[mod_order],
tb_sizes=[len(tb_input)]
)
num_tb_errors["Neural Rx"] += int(np.array_equal(tbs[0], tb_input) == False)

monitor.update(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
if (np.array(list(num_tb_errors.values())) >= min_num_tb_errors).all():
    break  # Next Es/No value.

monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
monitor.finish()

<table>
<thead>
<tr>
<th>Es/No (dB)</th>
<th>TBs</th>
<th>TB Errors</th>
<th>BLER</th>
<th>TB Errors</th>
<th>BLER</th>
<th>ms/TB</th>
</tr>
</thead>
</table>

(continues on next page)
3.3.7. Channel estimation on transmissions captured using Aerial Data Lake

This example shows how to query PUSCH data from an Aerial Data Lake database and perform channel estimation on that PUSCH data using pyAerial.

**Note:** This notebook requires that the clickhouse server used by Aerial Data Lake is running, and that the example data has been imported into a database. Refer to the Aerial Data Lake installation docs on how to do this.
3.3.7.1 Using pyAerial for channel estimation on Aerial Data Lake data

This example shows how to use the pyAerial bindings to run cuPHY GPU accelerated channel estimation for 5G NR PUSCH. 5G NR PUSCH data is read from an example over the air captured PUSCH dataset collected and stored using Aerial Data Lake, and the channel is estimated using pyAerial and cuPHY based on the corresponding PUSCH parameters.

**Note:** This example requires that the clickhouse server is running and that the example data has been stored in the database. Refer to the Aerial Data Lake documentation on how to do this.

```python
[1]: # Check platform.
    import platform
    if platform.machine() not in ['x86_64', 'aarch64']:
        raise SystemExit("Unsupported platform!")

3.3.7.1.1 Imports

```python
[2]:
    import math
    import os
    os.environ["CUDA_VISIBLE_DEVICES"] = "0"

    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt

    # Connecting to clickhouse on remote server
    import clickhouse_connect

    # Import the channel estimator and some utilities for converting
    # the DMRS fields in the right format from the SCF FAPI format that the dataset follows.
    from aerial.phy5g.algorithms import ChannelEstimator
    from aerial.util.fapi import dmrs_fapi_to_bit_array

3.3.7.1.2 Data

We use an example dataset which has been captured from a real over the air PUSCH transmission. The "fapi" table in the database contains the metadata for each PUSCH transmission and the "fh" table contains all of the samples for that slot.

```python
[3]: # Create the pyAerial (cuPHY) channel estimator.
    num_ues = 1
    num_rx_ant = 4
    channel_estimator = ChannelEstimator(num_rx_ant=num_rx_ant)

    # Connect to the local database
    client = clickhouse_connect.get_client(host='localhost')

    # Pick some pusch records from the database
    pusch_records = client.query_df('select * from fapi order by TsTaiNs limit 10')
```
3.3.7.1.3 Run channel estimation

From the PUSCH record we extract the PUSCH DMRS parameters and use the TAI time entry to select the IQ samples for that slot. Channel estimation is then run using the extracted parameters, and the absolute values of the estimated channels are plotted in the same figure.

```python
for index, pusch_record in pusch_records.iterrows():
    query = f"select TsTaiNs,fhData from fh where TsTaiNs == {pusch_record.TsTaiNs.timestamp()}"
    fh = client.query_df(query)

    # Make sure that the fronthaul database is complete for the SFN.Slot we've chosen
    if fh.index.size < 1:
        pusch_records = pusch_records.drop(index)
        continue;

    fh_samp = np.array(fh['fhData'][0], dtype=np.float32)
    rx_slot = np.swapaxes(fh_samp.view(np.complex64).reshape(4, 14, 273*12), 2, 0)

    # Extract all the needed parameters from the PUSCH record.
    slot = int(pusch_record.Slot)
    rntis = [pusch_record.rnti]
    layers = [pusch_record.nrOfLayers]
    start_prb = pusch_record.rbStart
    num_prbs = pusch_record.rbSize
    start_sym = pusch_record.StartSymbolIndex
    num_symbols = pusch_record.NrOfSymbols
    scids = [int(pusch_record.SCID)]
    data_scids = [pusch_record.dataScramblingId]
    dmrs_scrm_id = pusch_record.ulDmrsScramblingId
    num_dmrs_cdm_grps_no_data = pusch_record.numDmrsCdmGrpsNoData
    dmrs_syms = dmrs_fapi_to_bit_array(int(pusch_record.ulDmrsSymbPos))
    dmrs_ports = [pusch_record.dmrsPorts]
    dmrs_max_len = 1
    dmrs_add_ln_pos = 2
    num_subcarriers = num_prbs * 12
    mcs_tables = [pusch_record.mcsTable]
    mcs_indices = [pusch_record.mcsIndex]
    coderates = [pusch_record.targetCodeRate/10.]
    tb_sizes = [pusch_record.TBSize]
    mod_orders = [pusch_record.qamModOrder]
    tb_input = np.array(pusch_record.pduData)

    # Run PyAerial (cuPHY) channel estimation.
    ch_est = channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        layers=layers,
        scids=scids,
        slot=slot,
        dmrs_ports=dmrs_ports,
        dmrs_syms=dmrs_syms,
        dmrs_scrm_id=dmrs_scrm_id,
        dmrs_max_len=dmrs_max_len,
    )
```

(continues on next page)
Aerial CUDA-Accelerated RAN, Release 24-2

(continued from previous page)

dmrs_add_ln_pos = dmrs_add_ln_pos,
num_dmrs_cdm_grps_no_data = num_dmrs_cdm_grps_no_data,
start_prb = start_prb,
num_prbs = num_prbs,
start_sym = start_sym,
um_symbols = num_symbols

fig, axs = plt.subplots(1)
fig.suptitle("Channel estimates for SFN.Slot: "+str(pusch_record.SFN)+".
axs.set_title(pusch_record.TsTaiNs)
for ant in range(4):
    axs.plot(np.abs(ch_est[0][ant, 0, :, 0]))
axs.grid(True)
plt.show()
Channel estimates for SFN.Slot: 195.4
2024-03-21 12:18:39.192000
Channel estimates for SFN.Slot: 201.4
2024-03-21 12:18:39.252000
3.3.8. Decoding PUSCH transmissions captured using Aerial Data Lake

3.3.8.1 Using pyAerial for PUSCH decoding on Aerial Data Lake data

This example shows how to use the pyAerial bindings to run cuPHY GPU accelerated PUSCH decoding for 5G NR PUSCH. The 5G NR PUSCH data is read from an example over the air captured PUSCH dataset collected and stored using Aerial Data Lake. Building a PUSCH receiver using pyAerial is demonstrated in two ways, first by using a fully fused, complete, PUSCH receiver called from Python using just a single function call. The same is then achieved by building the complete PUSCH receiver using individual separate Python function calls to individual PUSCH receiver components.

Note: This example requires that the clickhouse server is running and that the example data has been stored in the database. Refer to the Aerial Data Lake documentation on how to do this.

[1]: # Check platform.
    import platform

(continues on next page)
if platform.machine() not in ['x86_64', 'aarch64']:
    raise SystemExit("Unsupported platform!")

3.3.8.1 Imports

```python
[2]:
import math
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
import numpy as np
import pandas as pd
from IPython.display import Markdown
from IPython.display import display

# Connecting to clickhouse on remote server
import clickhouse_connect

# Plotting with Bokeh.
import matplotlib.pyplot as plt

# pyAerial imports
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.pusch import PuschRx
from aerial.util.cuda import get_cuda_stream
from aerial.util.fapi import dmrs_fapi_to_bit_array

# Hide log10(10) warning
_= np.seterr(divide='ignore', invalid='ignore')
```

3.3.8.1.2 Create the PUSCH pipelines

This is a PUSCH receiver pipeline made up of separately called pyAerial PUSCH receiver components.

```python
[3]:
# Whether to plot intermediate results within the PUSCH pipeline, such as channel estimates and equalized symbols.
plot_figures = True

num_ues = 1
num_tx_ant = 2 # UE antennas
num_rx_ant = 4 # gNB antennas
cell_id = 41 # Physical cell ID
enable_pusch_tdi = 0 # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1 # Equalizer algorithm

# The PUSCH receiver chain built from separately called pyAerial Python components is defined here.
```
class PuschRxSeparate:

    """PUSCH receiver class. This class encapsulates the whole PUSCH receiver chain built using pyAerial components."
    """

    def __init__(self, num_rx_ant, enable_pusch_tdi, eq_coeff_algo, plot_figures):
        """Initialize the PUSCH receiver."""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver.
        self.channel_estimator = ChannelEstimator(
            num_rx_ant=num_rx_ant,
            cuda_stream=self.cuda_stream)
        self.channel_equalizer = ChannelEqualizer(
            num_rx_ant=num_rx_ant,
            enable_pusch_tdi=enable_pusch_tdi,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream)
        self.noise_intf_estimator = NoiseIntfEstimator(
            num_rx_ant=num_rx_ant,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream)
        self.derate_match = LdpcDeRateMatch(
            enable_scrambling=True,
            cuda_stream=self.cuda_stream)
        self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)

        # Whether to plot the intermediate results.
        self.plot_figures = plot_figures

    def run(self, rx_slot, num_ues, slot, num_dmrs_cdm_grps_no_data, dmrs_scrm_id, start_prb, num_prbs, dmrs_syms, dmrs_max_len, dmrs_add_ln_pos, start_sym, num_symbols, scids, layers, dmrs_ports, rntis, data_scids, code_rates,
(continues on next page)
mod_orders,
    tb_sizes
):
    
    """Run the receiver.""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_in_pos=dmrs_add_in_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        scids=scids,
        layers=layers,
        dmrs_ports=dmrs_ports
    )

    # Noise and interference estimation.
    lw_inv, noise_var_pre_eq = self.noise_intf_estimator.estimate(
        rx_slot=rx_slot,
        channel_est=ch_est,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_in_pos=dmrs_add_in_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        scids=scids,
        layers=layers,
        dmrs_ports=dmrs_ports
    )

    # Channel equalization and soft demapping. The first return value are the LLRs,
    # second are the equalized symbols. We only want the LLRs now.
    llrs, sym = self.channel_equalizer.equalize(
        rx_slot=rx_slot,
        channel_est=ch_est,
        lw_inv=lw_inv,
        noise_var_pre_eq=noise_var_pre_eq,
        num_ues=num_ues,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
dmrs_add_ln_pos = dmrs_add_ln_pos,
start_sym = start_sym,
um_symbols = num_symbols,
layers = layers,
mod_orders = mod_orders
)

if self.plot_figures:
    fig, axs = plt.subplots(1, 4)
    for ant in range(4):
        axs[ant].imshow(10 * np.log10(np.abs(rx_slot[:, :, ant] ** 2)), aspect='auto')
        axs[ant].set_ylim([pusch_record.rbStart * 12, pusch_record.rbSize * 12])
        axs[ant].set_title('Ant ' + str(ant))
        axs[ant].set_xlabel('Symbol', ylabel='Resource Element')
        axs[ant].label_outer()
fig.suptitle('Power in RU Antennas')
    fig, axs = plt.subplots(1, 2)
    axs[0].scatter(rx_slot.reshape(-1).real, rx_slot.reshape(-1).imag)
    axs[0].set_title('Pre-Equalized samples')
    axs[0].set_aspect('equal')
    axs[1].scatter(np.array(sym).reshape(-1).real, np.array(sym).reshape(-1).imag)
    axs[1].set_title('Post-Equalized samples')
    axs[1].set_aspect('equal')
    fig.set_title('Channel estimates from the PUSCH pipeline')
    for ant in range(4):
        axs.plot(np.abs(ch_est[0][ant, 0, :, 0]))
        axs.legend(['Rx antenna 0, estimate', 'Rx antenna 1, estimate', 'Rx antenna 2, estimate', 'Rx antenna 3, estimate'])
        axs.grid(True)
    plt.show()

num_data_sym = (np.array(dmrs_syms[start_sym:start_sym + num_symbols]) == 0).sum()
cinits = [(rntis[ue] << 15) + data_scids[ue] for ue in range(num_ues)]
rate_match_lengths = [num_data_sym * mod_orders[ue] * num_prbs * 12 * layers[ue] for ue in range(num_ues)]
tb_sizes = [s * 8 for s in tb_sizes]
code_rates = [c / 1024. for c in code_rates]
rvs = [0,] * num_ues
ndis = [1,] * num_ues
coded_blocks = self.derate_match.derate_match(
    input_llrs=llrs,
    tb_sizes=tb_sizes,
)

(continues on next page)
code_rates=code_rates,
rate_match_lengths=rate_match_lengths,
mod_orders=mod_orders,
num_layers=layers,
redundancy_versions=rvs,
ndis=ndis,
cinits=cinits
)

code_blocks = self.decoder.decode(
    input_llrs=coded_blocks,
tb_sizes=tb_sizes,
code_rates=code_rates,
redundancy_versions=rvs,
rate_match_lengths=rate_match_lengths
)

# TODO: Use the CRC kernel here.
decoded_tbs = []
for ue_idx in range(num_ues):
    # Combine the code blocks into a transport block.
    tb = code_block_desegment(
        code_blocks=coded_blocks[ue_idx],
tb_size=tb_sizes[ue_idx],
code_rate=code_rates[ue_idx],
return_bits=False,
    )

decoded_tbs.append(tb)

return decoded_tbs

pusch_rx_separate = PuschRxSeparate(
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
eq_coeff_algo=eq_coeff_algo,
plot_figures=plot_figures
)

# This is the fully fused PUSCH receiver chain.
pusch_rx = PuschRx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_tx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
eq_coeff_algo=eq_coeff_algo
)
3.3.8.1.3 Querying the database

Below shows how to connect to the clickhouse database and querying the data from it.

```python
# Connect to the local database
client = clickhouse_connect.get_client(host='localhost')

# Pick a packet from the database
pusch_records = client.query_df('select * from fapi where mcsIndex != 0 order by TsTaiNs limit 10')
```

3.3.8.1.4 Extract the PUSCH parameters and run the pipelines

```python
for index, pusch_record in pusch_records.iterrows():
    query = f'""'\select TsTaiNs,FhData from fh where TsTaiNs == '{pusch_record.TsTaiNs.timestamp()}""
    fh = client.query_df(query)
    display(Markdown("### Example {} - SFN.Slot {}\{} from time {}".".format(index + 1, pusch_record.SFN, pusch_record.Slot, pusch_record.TsTaiNs)))

    # Make sure that the fronthaul database is complete for the SFN.Slot we've chosen
    if fh.index.size < 1:
        pusch_records = pusch_records.drop(index)
        continue;

    fh_samp = np.array(fh['FhData'][0], dtype=np.float32)
    rx_slot = np.swapaxes(fh_samp.view(np.complex64).reshape(4, 14, 273 * 12), 2, 0)

    # Extract all the needed parameters from the PUSCH record.
    slot = int(pusch_record.Slot)
    rntis = [pusch_record.rnti]
    layers = [pusch_record.nrOfLayers]
    start_prb = pusch_record.rbStart
    num_prbs = pusch_record.rbSize
    start_sym = pusch_record.StartSymbolIndex
    num_symbols = pusch_record.NrOfSymbols
    scids = [int(pusch_record.SCID)]
    data_scids = [pusch_record.dataScramblingId]
    dmrs_scram_id = pusch_record.ulDmrsScramblingId
    num_dmrs_cdm_grps_no_data = pusch_record.numDmrsCdmGrpsNoData
    dmrs_syms = dmrs_fapi_to_bit_array(int(pusch_record.ulDmrsSymbPos))
    dmrs_ports = [pusch_record.dmrsPorts]
    dmrs_max_len = 1
    dmrs_add_ln_pos = 2
    mcs_tables = [pusch_record.mcsTable]
    mcs_indices = [pusch_record.mcsIndex]
    coderates = [pusch_record.targetCodeRate / 10.]
    tb_sizes = [pusch_record.TBSize]
    mod_orders = [pusch_record.qamModOrder]
    tb_input = np.array(pusch_record.pduData)

(continues on next page)
# Run the receiver built from separately called components.

tbs = pusch_rx_separate.run(
    rx_slot=rx_slot,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    scids=scids,
    layers=layers,
    dmrs_ports=dmrs_ports,
    rntis=rntis,
    data_scids=data_scids,
    code_rates=code_rates,
    mod_orders=mod_orders,
    tb_sizes=tb_sizes
)

if np.array_equal(tbs[0][:tb_input.size], tb_input):
    display(Markdown("**Separated kernels PUSCH decoding success** for SFN.Slot {}, →{}
from time {}\cdot\text{format}(pusch_record.SFN, pusch_record.Slot, pusch_record.TsTaiNs))
else:
    display(Markdown("**Separated kernels PUSCH decoding failure**"))
    print("Output bytes:")
    print(tbs[0][:tb_input.size])
    print("Expected output:")
    print(tb_input)

# Run the fused PUSCH receiver.
# Note that this is where we set the dynamically changing parameters.

tb_crcs, tbs = pusch_rx.run(
    rx_slot=rx_slot,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    scids=scids,
    layers=layers,
    dmrs_ports=dmrs_ports,
    rntis=rntis,
    data_scids=data_scids,
    code_rates=code_rates,
    mod_orders=mod_orders,
    tb_sizes=tb_sizes
)
if np.array_equal(tbs[0][:tb_input.size], tb_input):
    display(Markdown("**Fused PUSCH decoding success** for SFN.Slot {} from time {}".format(pusch_record.SFN, pusch_record.Slot, pusch_record.TsTaiNs)))
else:
    display(Markdown("**Fused PUSCH decoding failure**"))
    print("Output bytes: ")
    print(tbs[0][:tb_input.size])
    print("Expected output: ")
    print(tb_input)

3.3.8.1.4.1 Example 1 - SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

3.3. Examples of Using pyAerial
Separated kernels PUSCH decoding success for SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

Fused PUSCH decoding success for SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

3.3.8.1.4.2 Example 2 - SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000
3.3. Examples of Using pyAerial

Power in RU Antennas

Pre-Equalized samples

Post-Equalized samples
Separated kernels PUSCH decoding success for SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000

Fused PUSCH decoding success for SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000

3.3.8.1.4.3 Example 3 - SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000
3.3. Examples of Using pyAerial
Separated kernels PUSCH decoding success for SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000

Fused PUSCH decoding success for SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000

3.3.8.1.4.4 Example 4 - SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000
3.3. Examples of Using pyAerial
Separated kernels PUSCH decoding success for SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000

Fused PUSCH decoding success for SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000

3.3.8.1.4.5 Example 5 - SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000
Similarly to the previous example, this example illustrates the use of pyAerial in context of Aerial Data Lake. In this example, the PUSCH data queried from the database is run through a full PUSCH receiver pipeline implemented using the pyAerial API. The example also illustrates how the pyAerial PUSCH components enable fetching intermediate results from the receiver pipeline.

**Separated kernels PUSCH decoding success** for SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000

**Fused PUSCH decoding success** for SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000

*Note:* Similarly to the previous notebook, this notebook requires that the clickhouse server used by Aerial Data Lake is running, and that the example data has been imported into a database. Refer to the Aerial Data Lake installation docs on how to do this.

For more information, refer to the *Aerial Data Lake section.*
3.4. API Reference

3.4.1. Physical layer pipelines for 5G

This module contains classes implementing the 5G NR physical layer using GPU acceleration through the cuPHY library. The module contains full PDSCH transmitter and PUSCH receiver pipelines in `PdschTx` and `PuschRx`, respectively. The other parts of this module contain individual components of the transmitter-receiver chain, such as for example the LDPC encoder and decoder in `LdpcEncoder` and `LdpcDecoder`, and the channel estimator in `ChannelEstimator`.

3.4.1.1 Receiver algorithms

This module contains a number of receiver algorithms implemented in cuPHY, thus using GPU acceleration.

```python
class aerial.phy5g.algorithms.channel_estimator.ChannelEstimator
    Channel estimator class.
    This class implements traditional MMSE-based channel estimation on the DMRS symbols of the received slot signal. It outputs the channel estimates for all resource elements in the DMRS symbols. Similarly to many other classes in pyAerial, this class handles groups of UEs sharing the same time-frequency resources with one call, i.e. it supports MU-MIMO.

__init__(num_rx_ant, ch_est_algo=1, cuda_stream=None, chest_filter_h5=None, w_freq_array=None, w_freq4_array=None, w_freq_small_array=None, shift_seq_array=None, unshift_seq_array=None, shift_seq4_array=None, unshift_seq4_array=None)
```

Initialize ChannelEstimator.

The channel estimation filters can be given as an H5 file or directly as Numpy arrays. If neither is given, the channel estimator is using default filters.

**Parameters**

- **num_rx_ant**(int) – Number of receive antennas.
- **ch_est_algo**(int) – Channel estimation algorithm.
  - MMSE (~ 0 ~)
  - estimation (~ 1 ~ Multi-stage MMSE with delay)
  - yet (~ 2 ~ RKHS not supported by pyAerial)
  - only (~ 3 ~ LS channel estimation)
- **cuda_stream**(int) – The CUDA stream. If not given, one will be created.
- **chest_filter_h5**(str) – Filename of an HDF5 file containing channel estimation filters.
- **w_freq_array**(np.ndarray)
- **w_freq4_array**(np.ndarray)
- **w_freq_small_array**(np.ndarray)
Run the channel estimation.

This runs the cuPHY channel estimation for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

**Parameters**

- **rx_slot (np.ndarray)** – Input received data as a frequency x time x Rx antenna Numpy array with type np.complex64 entries.
- **num_ues (int)** – Number of UEs in the UE group.
- **layers (List[int])** – Number of layers for each UE. The length of the list equals the number of UEs.
- **scids (List[int])** – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE. Value is 0 or 1.
- **dmrs_ports (List[int])** – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- **slot (int)** – Slot number.
- **dmrs_syms (List[int])** – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- **dmrs_scram_id (int)** – DMRS scrambling ID.
- **dmrs_max_len (int)** – The maxLength parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos (int)** – Number of additional DMRS positions.
- **num_dmrs_cdm_grps_no_data (int)** – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **start_prb (int)** – Start PRB index of the UE allocation.
- **num_prbs (int)** – Number of allocated PRBs for the UE group.
- **start_sym (int)** – Start symbol index for the UE group allocation.
- **num_symbols (int)** – Number of symbols in the UE group allocation.
Returns
The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.

Return type
List[np.ndarray]

class aerial.phy5g.algorithms.noise_intf_estimator.NoiseIntfEstimator
Noise and interference estimator class.

This class implements an algorithm for noise and interference estimation. It calls the corresponding cuPHY algorithms and provides the estimates as needed for cuPHY equalization and soft demapping.

It needs channel estimates as its input, along with the received data symbols.

__init__(num_rx_ant, eq_coeff_algo, cuda_stream=None)
Initialize NoiseIntfEstimator.

Parameters

▶ num_rx_ant (int) – Number of receive antennas.
▶ eq_coeff_algo (int) – Algorithm used to compute equalizer coefficients.
   ▶ 0: Zero-forcing equalizer.
   ▶ 1: MMSE with noise variance only.
   ▶ 2: MMSE-IRC.
▶ cuda_stream (int) – The CUDA stream. If not given, one will be created.

Return type
None

estimate(rx_slot, channel_est, num_ues, slot, num_dmrs_cdm_grps_no_data, dmrs_scrm_id, start_prb, num_prbs, dmrs_syms, dmrs_max_len, dmrs_add_len_pos, start_sym, num_symbols, scids, layers, dmrs_ports)
Estimate noise and interference.

This runs the cuPHY noise and interference estimation for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

Parameters

▶ rx_slot (np.ndarray) – Input received data as a frequency x time x Rx antenna Numpy array with type np.complex64 entries.
▶ channel_est (List[np.ndarray]) – The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.
▶ num_ues (int) – Number of UEs in the UE group.
▶ slot (int) – Slot number.
▶ num_dmrs_cdm_grps_no_data (int) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
Aerial CUDA-Accelerated RAN, Release 24-2

- **dmrs_scrm_id** (int) – DMRS scrambling ID.
- **start_prb** (int) – Start PRB index of the UE allocation.
- **num_prbs** (int) – Number of allocated PRBs for the UE group.
- **dmrs_sym** (List[int]) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- **dmrs_max_len** (int) – The maxLength parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos** (int) – Number of additional DMRS positions.
- **start_sym** (int) – Start symbol index for the UE group allocation.
- **num_symbols** (int) – Number of symbols in the UE group allocation.
- **scids** (List[int]) – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE. Value is 0 or 1.
- **layers** (List[int]) – Number of layers for each UE. The length of the list equals the number of UEs.
- **dmrs_ports** (List[int]) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.

**Returns**

A tuple containing:

- List[np.ndarray]: Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs. Note that since only one UE group is currently supported, the length of this list is one.
- np.ndarray: Pre-equalization wideband noise variance estimate per UE group, i.e. one value per UE group averaged over the whole frequency allocation. This value is in dB.

**Return type**

List[np.ndarray], np.ndarray

```
class aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer
    Channel equalizer class.
    This class implements MMSE-based channel equalization along with soft demapping to get log-likelihood ratios for channel decoding.
    It needs channel estimates and noise and interference estimates as its input, along with the received data symbols.
    __init__(num_rx_ant, eq_coeff_algo, enable_pusch_tdi, cuda_stream=None)
    Initialize ChannelEqualizer.

    Parameters

    - num_rx_ant (int) – Number of receive antennas.
```
**eq_coeff_algo** (int) – Algorithm used to compute equalizer coefficients.
- 0: Zero-forcing equalizer.
- 1: MMSE with noise variance only.
- 2: MMSE-IRC.

**enable_pusch_tdi** (int) – Whether to use time-domain interpolation.

**cuda_stream** (int) – The CUDA stream. If not given, one will be created.

**Return type**
None

**equalize**(*rx_slot*, *channel_est*, *lw_inv*, *noise_var_pre_eq*, *num_ues*,
*num_dmrs_cdm_grps_no_data*, *start_prb*, *num_prbs*, *dmrs_syms*,
*dmrs_max_len*, *dmrs_add_ln_pos*, *start_sym*, *num_symbols*,
*layers*, *mod_orders*)

Run equalization and soft demapping.

This runs the cuPHY equalization for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

**Parameters**

- **rx_slot** (*np.ndarray*) – Input received data as a frequency x time x Rx antenna Numpy array with type np.complex64 entries.
- **channel_est** (*List[np.ndarray]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.
- **lw_inv** (*List[np.ndarray]*) – Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs.
- **noise_var_pre_eq** (*np.ndarray*) – Average pre-equalizer noise variance in dB. One value per UE group.
- **num_ues** (int) – Number of UEs in the UE group.
- **num_dmrs_cdm_grps_no_data** (int) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **start_prb** (int) – Start PRB index of the UE allocation.
- **num_prbs** (int) – Number of allocated PRBs for the UE group.
- **dmrs_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- **dmrs_max_len** (int) – The maxLength parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos** (int) – Number of additional DMRS positions.
- **start_sym** (int) – Start symbol index for the UE group allocation.
- **num_symbols** (int) – Number of symbols in the UE group allocation.
layers (List[int]) – Number of layers for each UE.

mod_orders (List[int]) – QAM modulation order for each UE.

Returns

A tuple containing:

- List[np.ndarray]: Log-likelihood ratios for the received bits to be fed into decoding (rate matching). One Numpy array per UE group and the size of each Numpy array is 8 x number of layers x number of subcarriers x number of data symbols. The size of the first dimension is fixed to eight as modulations up to 256QAM are supported and cuPHY returns the same size independently of modulation. Only the first entries corresponding to the actual number of bits are used.

- List[np.ndarray]: Equalized symbols, one Numpy array per UE group. The size of each Numpy array is equal to number of layers x number of subcarriers x number of data symbols.

Return type

List[np.ndarray], List[np.ndarray]

class aerial.phy5g.algorithms.demapper.Demapper

This class provides demapping of symbols to log-likelihood ratios.

The algorithm used is the exact log-MAP mapping, which is computationally intensive. Note also that this is currently implemented purely in Python so it may be slow.

__init__ (mod_order)

Initialize demapper.

Parameters

mod_order (int) – Modulation order. Supported values: 2, 4, 6, 8.

Return type

None

demap (syms, noise_var_inv)

Run demapping.

Parameters

- syms (np.ndarray) – An array of modulation symbols.

- noise_var_inv (np.ndarray) – Inverse of noise variance per subcarrier. The size of this array must broadcast with syms.

Returns

Log-likelihood ratios. The first dimension is modulation order, otherwise the dimensions are the same as those of syms.

Return type

np.ndarray

class aerial.phy5g.algorithms.trt_engine.TrtTensorPrms

Class to hold the TRT input and output tensor parameters.

property cuphy_data_type: aerial.pycuphy.DataType

Convert data type to cuPHY data type format.
__init__(name, dims, data_type=numpy.float32)

Parameters

- **name** (str) -
- **dims** (List[int]) -
- **data_type** (type) -

Return type
None

class aerial.phy5g.algorithms.trt_engine.TrtEngine

TensorRT engine class.

This class implements a simple wrapper around NVIDIA’s TensorRT and its cuPHY API. It takes a TRT engine file as its input, along with the names and dimensions of the input and output tensors. The TRT engine file can be generated offline from an .onnx file using the trtexec tool.

__init__(trt_model_file, max_batch_size, input_tensors, output_tensors, cuda_stream=None)

Initialize TrtEngine.

Parameters

- **trt_model_file** (str) – This is TRT engine (model) file.
- **max_batch_size** (int) – Maximum batch size.
- **input_tensors** (List[TrtTensorPrms]) – A mapping from tensor names to input tensor dimensions. The names are strings that must match with those found in the TRT model file, and the shapes are iterables of integers. The batch dimension is skipped.
- **output_tensors** (List[TrtTensorPrms]) – A mapping from tensor names to output tensor dimensions. The names are strings that must match with those found in the TRT model file, and the shapes are iterables of integers. The batch dimension is skipped.
- **cuda_stream** (int) – The CUDA stream. If not given, one will be created.

Return type
None

run(input_tensors)

Run the TensorRT model.

This runs the model using NVIDIA TensorRT engine.

Parameters

- **input_tensors** (dict) – A mapping from input tensor names to the actual input tensors. The tensor names must match with those given in the initialization, and with those found in the TRT model. Actual batch size is read from the tensor size.

Returns

A mapping from output tensor names to the actual output tensors.

Return type
dict
class aerial.phy5g.algorithms.srs_channel_estimator.SrsCellPrms
SRS cell parameters.

A list of SRS cell parameters is given to the SRS channel estimator as input, one entry per cell.

Parameters

▶ slot_num (np.uint16) – Slot number.
▶ frame_num (np.uint16) – Frame number.
▶ srs_start_sym (np.uint8) – SRS start symbol.
▶ num_srs_sym (np.uint8) – Number of SRS symbols.
▶ num_rx_ant_srs (np.uint16) – Number of SRS Rx antennas.
▶ mu (np.uint8) – Subcarrier spacing parameter, see TS 38.211.

class aerial.phy5g.algorithms.srs_channel_estimator.UeSrsPrms
UE SRS parameters.

A list of UE SRS parameters is given to the SRS channel estimator as input, one entry per UE.

Parameters

▶ cell_idx (np.uint16) – Index of cell user belongs to.
▶ num_ant_ports (np.uint8) – Number of SRS antenna ports. 1, 2, or 4.
▶ num_sym (np.uint8) – Number of SRS symbols. 1, 2, or 4.
▶ num_repetitions (np.uint8) – Number of repetitions. 1, 2, or 4.
▶ comb_size (np.uint8) – SRS comb size. 2 or 4.
▶ start_sym (np.uint8) – Starting SRS symbol. 0 - 13.
▶ sequence_id (np.uint16) – SRS sequence ID. 0 - 1023.
▶ config_idx (np.uint8) – SRS bandwidth configuration index. 0 - 63.
▶ bandwidth_idx (np.uint8) – SRS bandwidth index. 0 - 3.
▶ comb_offset (np.uint8) – SRS comb offset. 0 - 3.
▶ cyclic_shift (np.uint8) – Cyclic shift. 0 - 11.
▶ frequency_position (np.uint8) – Frequency domain position. 0 - 67.
▶ frequency_shift (np.uint16) – Frequency domain shift. 0 - 268.
▶ frequency_hopping (np.uint8) – Frequency hopping options. 0 - 3.
▶ resource_type (np.uint8) – Type of SRS allocation. 0: Aperiodic. 1: Semi-persistent. 2: Periodic.
▶ periodicity (np.uint16) – SRS periodicity in slots. 0, 2, 3, 5, 8, 10, 16, 20, 32, 40, 64, 80, 160, 320, 640, 1280, 2560.
▶ offset (np.uint16) – Slot offset value. 0 - 2569.
▶ group_or_sequence_hopping (np.uint8) – Hopping configuration. 0: No hopping. 1: Group hopping. 2: Sequence hopping.
- `ch_est_buff_idx (np.uint16)` – Index of which buffer to store SRS estimates into.
- `srs_ant_port_to_ue_ant_map (np.ndarray)` – Mapping between SRS antenna ports and UE antennas in channel estimation buffer: Store estimates for SRS antenna port i in srs_ant_port_to_ue_ant_map[i].
- `prg_size (np.uint8)` – Number of PRBs per PRG.

class aerial.phy5g.algorithms.srs_channel_estimator.SrsReport
SRS output report.
This report is returned by the SRS channel estimator.

Parameters
- `to_est_micro_sec (np.float32)` – Time offset estimate in microseconds.
- `wideband_snr (np.float3)` – Wideband SNR.
- `wideband_noise_energy (np.float32)` – Wideband noise energy.
- `wideband_signal_energy (np.float32)` – Wideband signal energy.
- `wideband_sc_corr (np.complex64)` – Wideband subcarrier correlation.
- `wideband_cs_corr_ratio_db (np.float32)` –
- `wideband_cs_corr_use (np.float32)` –
- `wideband_cs_corr_not_use (np.float32)` –

class aerial.phy5g.algorithms.srs_channel_estimator.SrsChannelEstimator
SrsChannelEstimator class.
This class implements SRS channel sounding for 5G NR.

__init__ (chest_params=None)
Initialize SrsChannelEstimator.

Parameters
`chest_params (dict)` – Dictionary of channel estimation filters and parameters. Set to None to use defaults.

Return type
None

estimate (rx_data, num_srs_ues, num_srs_cells, num_prb_grps, start_prb_grp, srs_cell_prms, srs_ue_prms)
Run SRS channel estimation.

Parameters
- `rx_data (np.ndarray)` – Input RX data, size num_subcarriers x num_srs_sym x num_rx_ant.
- `num_srs_ues (int)` – Number of UEs.
- `num_srs_cells (int)` – Number of SRS cells.
- `num_prb_grps (int)` – Number of PRB groups.
- `start_prb_grp (int)` – Start PRB group.
Aerial CUDA-Accelerated RAN, Release 24-2

- **srs_cell_prms** (*List[SrsCellPrms]*) – List of SRS cell parameters, one per cell.
- **srs_ue_prms** (*List[UeSrsPrms]*) – List of UE SRS parameters, one per UE.

**Returns**

A tuple containing:
- **List[np.ndarray]**: A list of channel estimates, one per UE. The channel estimate is a num_prb_grps x num_rx_ant x num_tx_ant numpy array.
- **np.ndarray**: SNRs per RB per UE.
- **List[SrsReport]**: A list of SRS wideband statistics reports, one per UE.

**Return type**

List[np.ndarray], np.ndarray, List[SrsReport]

### 3.4.1.2 PDSCH

This module contains classes related to the Physical Downlink Shared Channel, PDSCH.

**class aerial.phy5g.pdsch.pdsch_tx.PdschTx**

PDSCH transmitter.

This class implements the whole PDSCH transmission pipeline from the transmitted transport block to the transmitted frequency-domain symbols.

```python
__init__(cell_id, num_rx_ant, num_tx_ant, num_ul_bwp=273, num_dl_bwp=273, mu=1)
```

Initialize PdschTx.

**Parameters**

- **cell_id** (*int*) – Physical cell ID.
- **num_rx_ant** (*int*) – Number of receive antennas.
- **num_tx_ant** (*int*) – Number of transmit antennas.
- **num_ul_bwp** (*int*) – Number of PRBs in a uplink bandwidth part. Default: 273.
- **num_dl_bwp** (*int*) – Number of PRBs in a downlink bandwidth part. Default: 273.
- **mu** (*int*) – Numerology. Values in [0, 3]. Default: 1.

**Return type**

None

```python
run(tb_inputs, num_ues, slot, num_dmrs_cdm_grps_no_data=2, dmrs_scrm_id=41,
    resource_alloc=1, prb_bitmap=None, start_prb=0, num_prbs=273, dmrs_syms=None,
    start_sym=2, num_symbols=12, scids=None, layers=None, dmrs_ports=None,
    bwp_starts=None, ref_points=None, rntis=None, data_scids=None,
    precoding_matrices=None, mcs_tables=None, mcs_indices=None, code_rates=None,
    mod_orders=None, rvs=None, num_prb_lbrms=None, max_layers=None, max_qms=None)
```

Run PDSCH transmission.

Set dynamic PDSCH parameters and call cuPHY to run the PDSCH transmission.
Parameters

- **tb_inputs (List[np.ndarray])** – Transport blocks in bytes for each UE.
- **num_ues (int)** – Number of UEs.
- **slot (int)** – Slot number.
- **num_dmrs_cdm_grps_no_data (int)** – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **dmrs_scram_id (int)** – Downlink DMRS scrambling ID.
- **resource_alloc (int)** – Resource allocation type.
- **prb_bitmap (List[int])** – Array of bytes indicating bitmask for allocated RBs.
- **start_prb (int)** – Start PRB index for the UE group.
- **num_prbs (int)** – Number of allocated PRBs for the UE group.
- **dmrs_sym (List[int])** – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol.
- **start_sym (int)** – Start OFDM symbol index of the UE group allocation.
- **num_symbols (int)** – Number of symbols in the allocation, starting from start_sym.
- **scids (List[int])** – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- **layers (List[int])** – Number of layers for each UE.
- **dmrs_ports (List[int])** – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- **bwp_starts (List[int])** – Bandwidth part start (PRB number starting from 0). Used only if reference point is 1.
- **ref_points (List[int])** – DMRS reference point per UE. Value 0 or 1.
- **rntis (List[int])** –
- **data_scids (List[int])** – Data scrambling IDs for each UE, more precisely dataScramblingIdentityPdsch [TS38.211, sec 7.3.1.1].
- **precoding_matrices (List[np.ndarray])** – Precoding matrices, one per UE. The shape of each precoding matrix is number of layers x number of Tx antennas. If set to None, precoding is disabled.
- **mcs_tables (List[int])** – MCS table per UE.
- **mcs_indices (List[int])** – MCS index per UE.
- **code_rates (List[int])** – Code rate for each UE in 3GPP format, i.e. code rate x 1024.
- **mod_orders (List[int])** – Modulation order for each UE.
- **rvs (List[int])** – Redundancy version per UE (default: 0 for each UE).
- **num_prb_lbrms (List[int])** – Number of PRBs used for LBRM TB size computation. Possible values: {32, 66, 107, 135, 162, 217, 273}. 
max_layers (List[int]) – Number of layers used for LBRM TB size computation (at most 4).

max_qms (List[int]) – Modulation order used for LBRM TB size computation. Value: 6 or 8.

Returns

A tuple containing:

- np.ndarray: Transmitted OFDM symbols in a frequency x time x antenna tensor.
- np.ndarray: Coded bits in a num_codewords x num_bits_per_codeword tensor.

Return type

np.ndarray, np.ndarray

classmethod cuphy_to_tx(tx_slot, num_ues, dmrs_ports, scids, precoding_matrices=None)

Map cuPHY outputs to Tx antenna ports.

Parameters

- tx_slot (numpy.ndarray) – Transmit buffer from cuPHY.
- num_ues (int) – Number of UEs.
- dmrs_ports (List[int]) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- scids (List[int]) – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- precoding_matrices (List[np.ndarray]) – Precoding matrices, one per UE. The shape of each precoding matrix is number of layers x number of Tx antennas. If set to None, precoding is disabled.

Returns

Transmitted OFDM symbols in a frequency x time x antenna tensor.

Return type

np.ndarray

3.4.1.3 PUSCH

This module contains classes related to the Physical Uplink Shared Channel, PUSCH.

class aerial.phy5g.pusch.pusch_rx.PuschRx

PUSCH receiver pipeline.

This class implements the whole PUSCH reception pipeline from the received OFDM post-FFT symbols to the received transport block (along with CRC check).

__init__ (cell_id, num_rx_ant, num_tx_ant, num_ul_bwp=273, num_dl_bwp=273, mu=1, enable_cfo_correction=0, enable_to_estimation=0, enable_pusch_tdi=0, eq_coeff_algo=1, ldpc_kernel_launch=aerial.pycuphy.PuschLdpcKernelLaunch.PUSCH_RX_ENABLE_DRIVER_LDPC_LAUNCH)

Initialize PuschRx.
Parameters

- **cell_id** *(int)* – Physical cell ID.
- **num_rx_ant** *(int)* – Number of receive antennas.
- **num_tx_ant** *(int)* – Number of transmit antennas.
- **num_ul_bwp** *(int)* – Number of PRBs in a uplink bandwidth part. Default: 273.
- **num_dl_bwp** *(int)* – Number of PRBs in a downlink bandwidth part. Default: 273.
- **mu** *(int)* – Numerology. Values in [0, 3]. Default: 1.
- **enable_cfo_correction** *(int)* – Enable/disable CFO correction:
  - 0: Disable (default).
  - 1: Enable.
- **enable_to_estimation** *(int)* – Enable/disable time offset estimation:
  - 0: Disable (default).
  - 1: Enable.
- **enable_pusch_tdi** *(int)* – Time domain interpolation on PUSCH.
  - 0: Disable (default).
  - 1: Enable.
- **eq_coeff_algo** *(int)* – Algorithm for equalizer coefficient computation.
  - 0 - ZF.
  - 1 - MMSE (default).
  - 2 - MMSE-IRC.
- **ldpc_kernel_launch** *(PuschLdpcKernelLaunch)* – LDPC kernel launch method.

Return type

None

run(rx_slot, num_ues, slot=0, num_dmrs_cdm_grps_no_data=2, dmrs_scrm_id=41, start_prb=0, num_prbs=273, dmrs_syms=None, dmrs_max_len=2, dmrs_add_ln_pos=1, start_sym=2, num_symbols=12, scids=None, layers=None, dmrs_ports=None, rntis=None, data_scids=None, mcs_tables=None, mcs_indices=None, code_rates=None, mod_orders=None, tb_sizes=None, rvs=None, ndis=None)

Run PUSCH Rx.

This runs the cuPHY PUSCH receiver pipeline for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols. Default values get filled for the parameters that are not given.

Parameters

- **rx_slot** *(numpy.ndarray)* – A tensor representing the receive slot buffer of the cell.
- **num_ues** *(int)* – Number of UEs in the UE group.
Aerial CUDA-Accelerated RAN, Release 24-2

- **slot (int)** – Slot number.
- **num_dmers_cdm_grps_no_data (int)** – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **dmrs_scrm_id (int)** – DMRS scrambling ID.
- **start_prb (int)** – Start PRB index of the UE group allocation.
- **num_prbs (int)** – Number of allocated PRBs for the UE group.
- **dmrs_syms (List[int])** – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol.
- **dmrs_max_len (int)** – The maxLength parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos (int)** – Number of additional DMRS positions.
- **start_sym (int)** – Start OFDM symbol index for the UE group allocation.
- **num_symbols (int)** – Number of symbols in the UE group allocation.
- **scids (List[int])** – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- **layers (List[int])** – Number of layers for each UE.
- **dmrs_ports (List[int])** – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- **rntis (List[int])** –
- **data_scids (List[int])** – Data scrambling IDs for each UE, more precisely dataScramblingIdentityPdsch [TS38.211, sec 7.3.1.1].
- **mcs_tables (List[int])** – MCS table to use for each UE (see TS 38.214).
- **mcs_indices (List[int])** – MCS indices for each UE.
- **code_rates (List[float])** – Code rate for each UE. This is the number of information bits per 1024 coded bits.
- **mod_orders (List[int])** – Modulation order for each UE.
- **tb_sizes (List[int])** – TB size in bytes for each UE.
- **rvs (List[int])** – Redundancy versions for each UE.
- **ndis (List[int])** – New data indicator per UE.

**Returns**

A tuple containing:
- `np.ndarray`: Transport block CRCs.
- `List[np.ndarray]`: Transport blocks, one per UE.

**Return type**

`np.ndarray, List[np.ndarray]`
3.4.1.4 LDPC 5G

This module contains the API for using the GPU-accelerated LDPC coding chain from the cuPHY library. This includes encoding and decoding as well as rate matching. Code block segmentation and concatenation are implemented in Python. Additionally, this module contains a number of utility functions for example for determining the LDPC base graph, transport block size, etc.

class aerial.phy5g.ldpc.decoder.LdpcDecoder
LdpcDecoder

This class supports decoding of LDPC code blocks encoded following TS 38.212. It uses cuPHY accelerated LDPC decoding routines under the hood.

__init__(num_iterations=10, throughput_mode=False, cuda_stream=None)
Initialize LdpcDecoder.

Parameters

▶ num_iterations (int) – Number of LDPC decoder iterations. Default: 10.
▶ throughput_mode (bool) – Enable throughput mode. Default: False.
▶ cuda_stream (int) – The CUDA stream. If not given, one will be created.

Return type
None

decode(input_llrs, tb_sizes, code_rates, redundancy_versions, rate_match_lengths)
Decode function for LDPC decoder.

The decoder outputs decoded code blocks which can be further concatenated into the received transport block using code_block_desegment().

Parameters

▶ input_llrs (List[np.ndarray]) – Input LLRs per UE, each array is a N x C array of 32-bit floats, N being the number of LLRs per code block and C being the number of code blocks.
▶ tb_sizes (List[int]) – Transport block size in bits, without CRC, per UE.
▶ code_rates (List[float]) – Target code rates per UE.
▶ redundancy_versions (List[int]) – Redundancy version, i.e. 0, 1, 2, or 3, per UE.
▶ rate_match_lengths (int) – Number of rate matching output bits of each UE. This is equal to N.

Returns
The decoded bits in a numpy array.

Return type
List[np.ndarray]

set_num_iterations(num_iterations)
Set a particular value for the number of iterations to be run.

Parameters

▶ num_iterations (int) – Value of the number of iterations.
Return type
None

set_throughput_mode(throughput_mode)
Enable throughput mode.

Parameters
throughput_mode (bool) – Enable (True) throughput mode.

Return type
None

get_soft_bits()
Get the soft bit output from the decoder.

Returns
The soft bits in a numpy array.

Return type
List[np.ndarray]

class aerial.phy5g.ldpc.encoder.LdpcEncoder
LDPC encoder.

This class provides encoding of transmitted transport block bits using LDPC coding following TS 38.212. The encoding process is GPU accelerated using cuPHY routines. As the input, the transport blocks are assumed to be attached with the CRC and segmented to code blocks (as per TS 38.212).

__init__(num_profiling_iterations=0, puncturing=True, max_num_code_blocks=152, cuda_stream=None)
Initialize LdpcEncoder.

Initialization does all the necessary memory allocations for cuPHY.

Parameters

▶ num_profiling_iterations (int) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0.
▶ puncturing (bool) – Whether to puncture the systematic bits (2Zc). Default: True.
▶ max_num_code_blocks (int) – Maximum number of code blocks. Memory is allocated based on this. Default: 152.
▶ cuda_stream (int) – The CUDA stream. If not given, one will be created.

Return type
None

encode(input_data, tb_size, code_rate, redundancy_version)
Encode function for LDPC encoder.

The input to this function is code blocks, meaning that the code block segmentation is expected to be done before calling this function. Code block segmentation can be done using code_block_segment().

Parameters
**input_data** (*np.ndarray*) – The input code blocks as a K x C array where K is the number of input bits per code block (including CRCs) and C is the number of code blocks. The dtype of the input array must be *np.float32*.

**tb_size** (*int*) – Transport block size in bits, without CRC.

**code_rate** (*float*) – Target code rate.

**redundancy_version** (*int*) – Redundancy version, 0, 1, 2, or 3.

Returns
Encoded bits as a N x C array where N is the number of encoded bits per code block.

Return type
*np.ndarray*

```python
def set_profiling_iterations(num_profiling_iterations)
    Set a particular value for the number of profiling iterations to be run.

    Parameters
    num_profiling_iterations (*int*) – Value of the number of profiling iterations.

    Return type
    None
```

```python
def set_puncturing(puncturing)
    Set puncturing flag.

    Parameters
    puncturing (*bool*) – Whether to puncture the systematic bits (2*Zc). Default: True.

    Return type
    None
```

```python
class aerial.phy5g.ldpc.rate_match.LdpcRateMatch
    LDPC rate matching.

    __init__(enable_scrambling=True, num_profiling_iterations=0, max_num_code_blocks=152, cuda_stream=None)
    Initialize LdpcRateMatch.

    Initialization does all the necessary memory allocations for cuPHY.

    Parameters

    enable_scrambling (*bool*) – Whether to enable scrambling after code block concatenation.

    num_profiling_iterations (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0 (no profiling).

    max_num_code_blocks (*int*) – Maximum number of code blocks. Memory will be allocated based on this number.

    cuda_stream (*int*) – The CUDA stream. If not given, one will be created.

    Return type
    None
```
**rate_match**(*input_data*, *tb_size*, *code_rate*, *rate_match_len*, *mod_order*, *num_layers*,
*redundancy_version*, *cinit*)

LDPC rate matching function.

This function does rate matching of LDPC code blocks following TS 38.212. If scrambling is enabled, it also scrambles the rate matched bits. In this case the *c_init* value needs to be set to an appropriate scrambling sequence initialization value.

**Parameters**

- **input_data** (*np.ndarray*) – Input bits as a N x C numpy array with dtype *np.float32*, where N is the number of bits per code block and C is the number of code blocks.
- **tb_size** (*int*) – Transport block size in bits without CRC.
- **code_rate** (*float*) – Code rate.
- **rate_match_len** (*int*) – Number of rate matching output bits.
- **mod_order** (*int*) – Modulation order.
- **num_layers** (*int*) – Number of layers.
- **redundancy_version** (*int*) – Redundancy version, i.e. 0, 1, 2, or 3.
- **cinit** (*int*) – The *c_init* value used for initializing scrambling.

**Returns**

Rate matched bits.

**Return type**

*np.ndarray*

**set_profiling_iterations**(*num_profiling_iterations*)

Set a particular value for the number of profiling iterations to be run.

**Parameters**

- **num_profiling_iterations** (*int*) – Value of the number of profiling iterations.

**Return type**

None

class aerial.phy5g.ldpc.derate_match.LdpcDeRateMatch

LDPC derate matching.

**__init__**(enable_scrambling=True, cuda_stream=None)

Initialize LdpcDeRateMatch.

Initialization does all the necessary memory allocations for cuPHY.

**Parameters**

- **enable_scrambling** (*bool*) – Whether to descramble the bits before derate matching. Default: True.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None
derate_match(input_llrs, tb_sizes, code_rates, rate_match_lengths, mod_orders, num_layers, redundancy_versions, ndis, cinits, ue_grp_idx=None)

LDPC derate matching function.

Parameters

- **input_llrs** (List[np.ndarray]) – Input LLRs as a N x 1 numpy array with dtype np.float32, where N is the number of LLRs coming from the equalizer. Ordering of this input data is bitsPerQam x numLayers x numSubcarriers x numDataSymbols. One entry per UE group.
- **tb_sizes** (List[int]) – Transport block sizes in bits without CRC, per UE.
- **code_rates** (List[float]) – Code rates per UE.
- **rate_match_lengths** (List[int]) – Number of rate matching output bits, the same as N, per UE.
- **mod_orders** (List[int]) – Modulation order per UE.
- **num_layers** (List[int]) – Number of layers per UE.
- **redundancy_versions** (List[int]) – Redundancy version, i.e. 0, 1, 2, or 3, per UE.
- **ndis** (List[int]) – New data indicator per UE.
- **cinits** (List[int]) – The c_init value used for initializing scrambling for each UE.
- **ue_grp_idx** (List[int]) – The UE group index for each UE. Default is one-to-one mapping.

Returns

Derate matched LLRs for each UE.

Return type

List[np.ndarray]

aerial.phy5g.ldpc.util.get_mcs(mcs, table_idx=2)

Get modulation order and code rate based on MCS index.

Parameters

- **mcs** (int) – MCS index pointing to the table indicated by table_idx.

Returns

A tuple containing:

- **int**: Modulation order.
- **float**: Code rate * 1024.

Return type

int, float
aerial.phy5g.ldpc.util.get_tb_size(mod_order, code_rate, dmrs_syms, num_prbs, start_sym, num_symbols, num_layers)

Get transport block size based on given parameters.
Determine transport block size as per TS 38.214 section 5.1.3.2.

Parameters

▶ mod_order (int) – Modulation order.
▶ code_rate (float) – Code rate * 1024 as in section 5.1.3.1 of TS 38.214.
▶ dmrs_syms (List[int]) – List of binary numbers indicating which symbols contain DMRS.
▶ num_prbs (int) – Number of PRBs.
▶ start_sym (int) – Starting symbol.
▶ num_symbols (int) – Number of symbols.
▶ num_layers (int) – Number of layers.

Returns
Transport block size in bits.

Return type
int

aerial.phy5g.ldpc.util.get_base_graph(tb_size, code_rate)
Get LDPC base graph.

Parameters

▶ tb_size (int) – Transport block size in bits, without CRC.
▶ code_rate (float) – Code rate.

Returns
Base graph, 1 or 2.

Return type
int

aerial.phy5g.ldpc.util.max_code_block_size(base_graph)
Get maximum LDPC code block size based on base graph.

Parameters

base_graph (int) – Base graph, 1 or 2.

Returns
Maximum code block size.

Return type
int

aerial.phy5g.ldpc.util.find_lifting_size(base_graph, tb_size)
Find lifting size for base graph.

Parameters

▶ base_graph (int) – Base graph, 1 or 2.
- **tb_size** (int) – Transport block size in bits without CRC.

**Returns**
Lifting size.

**Return type**
int

```c
aerial.phy5g.ldpc.util.get_num_info_nodes(base_graph, tb_size)
```
Get number of information nodes.

Note: This is the value $K_b$ in TS 38.212.

**Parameters**

- **base_graph** (int) – Base graph, 1 or 2.
- **tb_size** (int) – Transport block size without any CRCs.

**Returns**
The number of information nodes ($K_b$).

**Return type**
int

```c
aerial.phy5g.ldpc.util.get_code_block_num_info_bits(base_graph, tb_size)
```
Get number of information bits in a code block.
This is the number $K'$ in TS 38.212, i.e. the number of information bits without the filler bits.

**Parameters**

- **base_graph** (int) – Base graph, 1 or 2.
- **tb_size** (int) – Transport block size in bits, without CRC.

**Returns**
Number of information bits in a code block.

**Return type**
int

```c
aerial.phy5g.ldpc.util.get_code_block_size(tb_size, code_rate)
```
Get code block size.
This is the number K in TS 38.212, i.e. the number of information bits including filler bits.

**Parameters**

- **tb_size** (int) – Transport block size in bits, without CRC.
- **code_rate** (float) – Code rate.

**Returns**
Code block size.

**Return type**
int

```c
aerial.phy5g.ldpc.util.get_num_code_blocks(tb_size, code_rate)
```
Return the number of code blocks for a transport block.
Parameters

- **tb_size** *(int)* – Transport block size in bits, without CRC.
- **code_rate** *(float)* – Code rate.

Returns

The number of code blocks (C).

Return type

**int**

```python
aerial.phy5g.ldpc.util.code_block_segment(tb_size, transport_block, code_rate)
```

Do code block segmentation.

This function does code block segmentation as per TS 38.212 section 5.2.2. Randomly generated 24-bit string is attached to each code block to emulate code block CRC if there is more than one code block.

Parameters

- **tb_size** *(int)* – Transport block size in bits, without CRC.
- **transport_block** *(np.ndarray)* – Transport block in bits, CRC included.
- **code_rate** *(float)* – Code rate.

Returns

The code blocks.

Return type

**np.ndarray**

```python
aerial.phy5g.ldpc.util.code_block_desegment(code_blocks, tb_size, code_rate, return_bits=True)
```

Concatenate code blocks coming from LDPC decoding into a transport block.

This function desegments code blocks into a transport block as per TS 38.212, and removes the CRCs, i.e. does the opposite of `code_block_segment()`.

Parameters

- **code_blocks** *(np.ndarray)* – The code blocks coming out of the LDPC decoder as a N x C array.
- **tb_size** *(int)* – Transport block size, without CRC.
- **code_rate** *(float)* – Code rate.
- **return_bits** *(bool)* – If True (default), give the return value in bits. Otherwise convert to bytes.

Returns

The transport block with CRC, in bits or bytes depending on the value of `return_bits`.

Return type

**np.ndarray**

```python
aerial.phy5g.ldpc.util.add_crc_len(tb_size)
```

Append CRC length to transport block size.
Parameters
  \texttt{tb\_size (int)} – Transport block size in bits without CRC.

Returns
  Transport block size in bits with CRC.

Return type
  int

\begin{verbatim}
aerial.phy5g.ldpc.util.random_tb(mod\_order, code\_rate, dmrs\_syms, num\_prbs, start\_sym, num\_symbols, num\_layers, return\_bits=False)
\end{verbatim}

Generate a random transport block.

Generates random transport block according to given parameters. The transport block size is first determined as per TS 38.214 section 5.1.3.2.

Parameters
  \begin{itemize}
  \item \texttt{mod\_order (int)} – Modulation order.
  \item \texttt{code\_rate (float)} – Code rate * 1024 as in section 5.1.3.1 of TS 38.214.
  \item \texttt{dmrs\_syms (List [int])} – List of binary numbers indicating which symbols contain DMRS.
  \item \texttt{num\_prbs (int)} – Number of PRBs.
  \item \texttt{start\_sym (int)} – Starting symbol.
  \item \texttt{num\_symbols (int)} – Number of symbols.
  \item \texttt{num\_layers (int)} – Number of layers.
  \item \texttt{return\_bits (bool)} – Whether to return the transport block in bits (True) or bytes (False).
  \end{itemize}

Returns
  Random transport block payload.

Return type
  np.ndarray

\begin{verbatim}
aerial.phy5g.ldpc.util.get\_crc\_len(tb\_size)
\end{verbatim}

Return CRC length based on transport block size.

Parameters
  \texttt{tb\_size (int)} – Transport block size in bits without CRC.

Returns
  CRC length (either 16 or 24 bits).

Return type
  int
3.4.2. Utilities

3.4.2.1 FAPI and Matlab interface utilities

The FAPI module contains various utilities for handling the interface between the PUSCH database schema (SCF FAPI) and cuPHY.

*aerial.util.fapi.dmrs_fapi_to_bit_array(dmrs symbolism)*

Convert the DMRS symbol position decimal value to a bit array.

**Parameters**

- `dmrs_symbol_pos` (*np.uint16*) – DMRS symbol position decimal value as defined in SCF FAPI.

**Returns**

A bit array to be used for cuPHY interface, indicating the positions of DMRS symbols. The first bit corresponds to OFDM symbol 0.

**Return type**

list

*aerial.util.fapi.dmrs_bit_array_to_fapi(x)*

Convert a bit array to DMRS symbol position decimal value.

**Parameters**

- `x` (*list*) – A bit array to be used for cuPHY interface, indicating the positions of DMRS symbols. The first bit corresponds to OFDM symbol 0.

**Returns**

DMRS symbol position decimal value as defined in SCF FAPI.

**Return type**

`np.uint16`

*aerial.util.fapi.dmrs_fapi_to_sym(dmrs_symbol_pos)*

Convert the DMRS symbol position decimal value to a list of DMRS symbol indices.

**Parameters**

- `dmrs_symbol_pos` (*np.uint16*) – DMRS symbol position decimal value as defined in SCF FAPI.

**Returns**

A list of DMRS symbol indices.

**Return type**

list

*aerial.util.fapi.mac_pdu_to_bit_array(mac_pdu)*

Convert MAC PDU bytes to a bit array.

**Parameters**

- `mac_pdu` (*list*) – A list of bytes, the content of the MAC PDU.

**Returns**

The same MAC PDU as a bit array, i.e. the bytes are converted to a list of bits.

**Return type**

list
aerial.util.fapi.bit_array_to_mac_pdu(bits)
     Convert a bit array to MAC PDU bytes.

     Parameters
     bits (list) – A MAC PDU as a bit array.

     Returns
     A list of bytes corresponding to the above MAC PDU.

     Return type
     list

3.4.2.2 Data storing utilities

class aerial.util.data.PuschRecord
     Implements column schema of a PUSCH dataframe row.

     The PuschRecord includes fields collected from the data collection agent, and SCF FAPI message content for the PUSCH channels from UL_TTI.request, RxData.indication, and CRC.indication.

     Parameters

     ▶ Slot – Slot number. Value: 0 - 159.
     ▶ nPDUs – Number of PDUs that were included in the UL_TTI.request message.
     ▶ RachPresent – Indicates if a RACH PDU was included in the UL_TTI.request message.
         ▶ 0: No RACH in this slot.
         ▶ 1: RACH in this slot.
     ▶ nULSCH – Number of ULSCH PDUs that were included in the UL_TTI.request message. Value: 0 - 255.
     ▶ nULCCH – Number of ULCCH PDUs that were included in the UL_TTI.request message. Value: 0 - 255.
     ▶ nGroup – Number of UE Groups that were included in the UL_TTI.request message. Value: 0 - 8.
     ▶ PDUSize – Size of the PDU control information (in bytes). This length value includes the 4 bytes required for the PDU type and PDU size parameters. Value: 0 - 65535.
     ▶ nUE – Number of UEs in this group. For SU-MIMO, one group includes one UE only. For MU-MIMO, one group includes up to 12 UEs. Value: 1 - 6, None if nGroup = 0.
     ▶ pduIdx – This value is an index for number of PDU identified by nPDU in the UL_TTI.request message. Value: 0 - 255, None if nGroup = 0.
     ▶ pduBitmap – Bitmap indicating presence of optional PDUs.
         ▶ Bit 0: puschData (Indicates data is expected on the PUSCH).
         ▶ Bit 1: puschUci (Indicates UCI is expected on the PUSCH).
         ▶ Bit 2: puschPtrs (Indicates PTRS included (FR2)).
Bit 3: dftsOfdm (Indicates DFT S-OFDM transmission).

All other bits reserved.

RNTI – The RNTI used for identifying the UE when receiving the PDU. Value: 1 - 65535.

Handle – An opaque handling returned in the RxData.indication and/or UCI.indication message.

BWPSize – Bandwidth part size [TS38.213 sec12]. Number of contiguous PRBs allocated to the BWP. Value: 1 - 275.

BWPSStart – Bandwidth part start RB index from reference CRB [TS38.213 sec 12]. Value: 0 - 274.

SubcarrierSpacing – SubcarrierSpacing [TS38.211 sec 4.2]. Value: 0 - 4.

CyclicPrefix – Cyclic prefix type [TS38.211 sec 4.2].

0: Normal
1: Extended

TargetCodeRate – Target coding rate [TS38.214 sec 6.1.4.1]. This is the number of information bits per 1024 coded bits expressed in 0.1 bit units.

qamModOrder – QAM modulation [TS38.214 sec 6.1.4.1]. Values:

2,4,6,8 if transform precoding is disabled.
1,2,4,6,8 if transform precoding is enabled.

mcsIndex – MCS index [TS38.214, sec 6.1.4.1], should match value sent in DCI. Value: 0 - 31.

mcsTable – MCS-Table-PUSCH [TS38.214, sec 6.1.4.1]. Value:

0: notqam256 [TS38.214, table 5.1.3.1-1].
1: qam256 [TS38.214, table 5.1.3.1-2].
2: qam64LowSE [TS38.214, table 5.1.3.1-3].
3: notqam256-withTransformPrecoding [TS38.214, table 6.1.4.1-1].
4: qam64LowSE-withTransformPrecoding [TS38.214, table 6.1.4.1-2].

TransformPrecoding – Indicates if transform precoding is enabled or disabled [TS38.214, sec 6.1.4.1] [TS38.211 6.3.1.4].

0: Enabled
1: Disabled

dataScramblingId – dataScramblingIdentityPdsch [TS38.211, sec 6.3.1.1]. It equals the higher-layer parameter Data-scrambling-Identity if configured and the RNTI equals the C-RNTI, otherwise L2 needs to set it to physical cell ID. Value: 0 - 65535.

nrOfLayers – Number of layers [TS38.211, sec 6.3.1.3]. Value: 1 - 4.

ulDmrsSymbPos – DMRS symbol positions [TS38.211, sec 6.4.1.1.3 and Tables 6.4.1.1.3-3 and 6.4.1.1.3-4]. Bitmap occupying the 14 LSBs with bit 0 corresponding to the first symbol and for each bit, value 0 indicates no DMRS and value 1 indicates DMRS.
dmrsConfigType – UL DMRS config type [TS38.211, sec 6.4.1.1.3].
- 0: type 1
- 1: type 2

ulDmrsScramblingId – UL-DMRS-Scrambling-ID [TS38.211, sec 6.4.1.1.2]. If provided and the PUSCH is not a msg3 PUSCH, otherwise, L2 should set this to physical cell ID. Value: 0 - 65535.

puschIdentity – PUSCH-ID [TS38.211, sec 6.4.1.1.2]. If provided and the PUSCH is not a msg3 PUSCH, otherwise, L2 should set this to physical cell ID. Value: 0 - 1007.

SCID – DMRS sequence initialization [TS38.211, sec 6.4.1.1.1]. Should match what is sent in DCI 0_1, otherwise set to 0. Value: 0 - 1.

numDmrsCdmGrpsNoData – Number of DM-RS CDM groups without data [TS38.212 sec 7.3.1.1]. Value: 1 - 3.

dmrsPorts – DMRS ports. [TS38.212 7.3.1.1.2] provides description between DCI 0-1 content and DMRS ports. Bitmap occupying the 11 LSBs with bit 0 corresponding to antenna port 1000 and bit 11 corresponding to antenna port 1011 and for each bit:
- 0: DMRS port not used.
- 1: DMRS port used.

resourceAlloc – Resource Allocation Type [TS38.214, sec 6.1.2.2].
- 0: Type 0.
- 1: Type 1.

rbBitmap – For resource allocation type 0. [TS38.214, sec 6.1.2.2.1] [TS 38.212, 7.3.1.1.2] bitmap of RBs, 273 rounded up to multiple of 32. This bitmap is in units of VRBs. LSB of byte 0 of the bitmap represents the first RB of the BWP. Each element is of type numpy.uint8.

rbStart – For resource allocation type 1. [TS38.214, sec 6.1.2.2.2]. The starting resource block within the BWP for this PUSCH. Value: 0 - 274.

rbSize – For resource allocation type 1. [TS38.214, sec 6.1.2.2.2]. The number of resource block within for this PUSCH. Value: 1 - 275.

VRBtoPRBMapping – VRB to PRB mapping [TS38.211, sec 6.3.1.7].
- 0: Non-interleaved.
- 1: Interleaved.

FrequencyHopping – For resource allocation type 1, indicates if frequency hopping is enabled. [TS38.212, sec 7.3.1.1] [TS38.214, sec 6.3].
- 0: Disabled.
- 1: Enabled.

txDirectCurrentLocation – The uplink Tx Direct Current location for the carrier. Only values in the value range of this field between 0 and 3299, which indicate the subcarrier index within the carrier corresponding to the numerology of the corresponding uplink BWP and value 3300, which indicates “Outside the carrier” and value 3301, which indicates “Undetermined position within the carrier” are used. [TS38.331, UplinkTxDirectCurrentBWP IE]. Value: 0 - 4095.
- uplinkFrequencyShift7p5khz – Indicates whether there is 7.5 kHz shift or not. [TS38.331, UplinkTxDirectCurrentBWP IE].
  ▶ 0: False.
  ▶ 1: True.
- StartSymbolIndex – Start symbol index of PUSCH mapping from the start of the slot, S. [TS38.214, Table 6.1.2.1-1]. Value: 0 - 13.
- puschData – See SCF FAPI 10.02, Table 3-47.  
dict{'cbPresentAndPosition': array([], dtype=int32), 'harqProcessID': np.uint8, 'newDataIndicator': np.uint8, 'numCb': np.uint8, 'rvIndex': np.uint8, 'TBSize': np.uint32}
- puschUci – See SCF FAPI 10.02, Table 3-48.
- puschPtrs – See SCF FAPI 10.02, Table 3-49.
- dftsOfdm – See SCF FAPI 10.02, Table 3-50.
- Beamforming – See SCF FAPI 10.02, Table 3-53.
- PDULen – Length of PDU in bytes. A length of 0 indicates a CRC or decoding error.
- UL_CQI – SNR.
- RSSI – RSSI. See SCF FAPI 10.02 Table 3-16 for RSSI definition.
- macPdu – Contents of MAC PDU. Each element is of type numpy:uint8.
- TbCrcStatus – Indicates CRC result on TB data. Each element is of type numpy:uint8.
  ▶ 0: Pass.
  ▶ 1: Fail.
- NumCb – If CBG is not used this parameter can be set to zero. Otherwise the number of CBs in the TB. Value: 0 - 65535.
- CbCrcStatus – Byte-packed array where each bit indicates CRC result on CB data. Each element is of type numpy:uint8.
  ▶ 0: Pass.
  ▶ 1: Fail.
  ▶ None if NumCb = 0.
- rx_iq_data_filename – Filename of the received OFDM IQ data file. This file contains the complex OFDM slot data as a frequency x time x antenna numpy array.
- user_data_filename – Filename of the user data file. This file may contain for example ground truth data.
- errInd – Freeform error indication message.
Notes

The PDULen field is 32 bits whereas SCF FAPI 10.02 incorrectly uses 16 bits. Using 32 bits allows MAC PDUs larger than 65535 bytes.

static from_series(series)
Create a PuschRecord from a Pandas Series entry (e.g. a DataFrame row).

Parameters
series (pandas.Series) – The input dataframe row.

Returns
The PUSCH record built from the given Pandas Series.

Return type
PuschRecord

static columns()
Return the field names of PuschRecord.

Return type
Tuple

aerial.util.data.save_pickle(data, filename, s3=None)
Save the data in a pickle file either locally or on S3.

Parameters

▶ data (np.ndarray or dict) – The data to be saved.
▶ filename (str) – Full path of the file to be used.
▶ s3 (s3fs.S3FileSystem) – The S3 filesystem to be used. Set to None for local filesystem.

Return type
None

aerial.util.data.load_pickle(filename, s3=None)
Load data from a pickle file, either a local file or on S3.

Parameters

▶ filename (str) – Full path of the file to be used.
▶ s3 (s3fs.S3FileSystem) – The S3 filesystem to be used. Set to None for local filesystem.

Returns
The loaded data.

Return type
np.ndarray or dict
3.4.2.3 CUDA utilities

aerial.util.cuda.get_cuda_stream()

Return a CUDA stream.

Returns
A new CUDA stream.

Return type
cudart.cudaStream_t

aerial.util.cuda.check_cuda_errors(result)

Check CUDA errors.

Parameters
result (cudart.cudaError_t) – CUDA error value.

Return type
Any

Copyright

©2024, NVIDIA
Python Module Index

a
aerial.phy5g.algorithms.channel_equalizer, 518
aerial.phy5g.algorithms.channel_estimator, 515
aerial.phy5g.algorithms.demapper, 520
aerial.phy5g.algorithms.noise_intf_estimator, 517
aerial.phy5g.algorithms.srs_channel_estimator, 521
aerial.phy5g.algorithms.trt_engine, 520
aerial.phy5g.ldpc.decoder, 529
aerial.phy5g.ldpc.derate_match, 532
aerial.phy5g.ldpc.encoder, 530
aerial.phy5g.ldpc.rate_match, 531
aerial.phy5g.ldpc.util, 533
aerial.phy5g.pdsch.pdsch_tx, 524
aerial.phy5g.pusch.pusch_rx, 526
aerial.util.cuda, 544
aerial.util.data, 539
aerial.util.fapi, 538
Index

Non-alphabetical

__init__() (aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer method), 518
__init__() (aerial.phy5g.algorithms.channel_estimator.ChannelEstimator method), 515
__init__() (aerial.phy5g.algorithms.demapper.Demapper method), 520
__init__() (aerial.phy5g.algorithms.noise_intf_estimator.NoiseIntfEstimator method), 517
__init__() (aerial.phy5g.algorithms.srs_channel_estimator.SrsChannelEstimator method), 523
__init__() (aerial.phy5g.algorithms.trt_engine.TrtEngine method), 521
__init__() (aerial.phy5g.algorithms.trt_engine.TrtTensorPrms method), 520
__init__() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 529
__init__() (aerial.phy5g.ldpc.derate_match.LdpcDeRateMatch method), 532
__init__() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 530
__init__() (aerial.phy5g.ldpc.rate_match.LdpcRateMatch method), 531
__init__() (aerial.phy5g.pdsch.pdsch_tx.PdschTx method), 524
__init__() (aerial.phy5g.pusch.pusch_rx.PuschRx method), 526

A

add_crc_len() (in module aerial.phy5g.ldpc.util), 536
aerial.phy5g.algorithms.channel_equalizer module, 518
aerial.phy5g.algorithms.channel_estimator module, 515
aerial.phy5g.algorithms.demapper module, 520
aerial.phy5g.algorithms.noise_intf_estimator module, 517
aerial.phy5g.algorithms.srs_channel_estimator module, 521
aerial.phy5g.algorithms.trt_engine module, 520

B

bit_array_to_mac_pdu() (in module aerial.util.fapi), 538

C

ChannelEqualizer (class in aerial.phy5g.algorithms.channel_equalizer), 518
ChannelEstimator (class in aerial.phy5g.algorithms.channel_estimator), 515
check_cudastat() (in module aerial.util.cuda), 544
code_block_desegment() (in module aerial.phy5g.ldpc.util), 536
code_block_segment() (in module aerial.phy5g.ldpc.util), 536
columns() (aerial.util.data.PuschRecord static method), 543
cuphy_data_type (aerial.phy5g.algorithms.trt_engine.TrtTensorPrms property), 520
cuphy_to_tx() (aerial.phy5g.pdsch.pdsch_tx.PdschTx class method), 526

get_soft_bits() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 530
get_tb_size() (in module aerial.phy5g.ldpc.util), 533

de decode() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 529
demap() (aerial.phy5g.algorithms.demapper.Demapper method), 520
Demapper (class in aerial.phy5g.algorithms.demapper), 520
derate_match() (aerial.phy5g.ldpc.derate_match.LdpcDeRateMatch method), 532
dmrs_bit_array_to_fapi() (in module aerial.util.fapi), 538
dmrs_fapi_to_bit_array() (in module aerial.util.fapi), 538
dmrs_fapi_to_sym() (in module aerial.util.fapi), 538

encode() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 530
equalize() (aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer method), 519
estimate() (aerial.phy5g.algorithms.channel_estimator.ChannelEstimator method), 515
estimate() (aerial.phy5g.algorithms.noise_intf_estimator.NoiseIntfEstimator method), 517
estimate() (aerial.phy5g.algorithms.srs_channel_estimator.SrsChannelEstimator method), 521
find_lifting_size() (in module aerial.phy5g.ldpc.util), 534
from_series() (aerial.util.data.PuschRecord static method), 543

get_base_graph() (in module aerial.phy5g.ldpc.util), 534
get_code_block_num_info_bits() (in module aerial.phy5g.ldpc.util), 535
get_code_block_size() (in module aerial.phy5g.ldpc.util), 535
get_crc_len() (in module aerial.phy5g.ldpc.util), 537
get_cuda_stream() (in module aerial.util.cuda), 544
get_mcs() (in module aerial.phy5g.ldpc.util), 533
get_num_code_blocks() (in module aerial.phy5g.ldpc.util), 535
get_num_info_nodes() (in module aerial.phy5g.ldpc.util), 535

deencode() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 530

demapper (class in aerial.phy5g.algorithms.demapper), 520
derate_match (class in aerial.phy5g.ldpc.derate_match), 532
ldpc_decoder (class in aerial.phy5g.ldpc.decoder), 529
LdpcDeRateMatch (class in aerial.phy5g.ldpc.derate_match), 532
LdpcRateMatch (class in aerial.phy5g.ldpc.encoder), 530
LdpcEncoder (class in aerial.phy5g.ldpc.encoder), 530
LdpcRateMatch (class in aerial.phy5g.ldpc.rate_match), 531
load_pickle() (in module aerial.util.data), 543
max_code_block_size() (in module aerial.phy5g.ldpc.util), 534
mac_pdu_to_bit_array() (in module aerial.util.fapi), 538

NoiseIntfEstimator (class in aerial.phy5g.algorithms.noise_intf_estimator), 517
PdschTx (class in aerial.phy5g.pdsch.pdsch_tx), 524
PuschRecord (class in aerial.util.data), 539
PuschRx (class in aerial.phy5g.pusch.pusch_rx), 526

random_tb() (in module aerial.phy5g.ldpc.util), 537
rate_match() (aerial.phy5g.ldpc.rate_match.LdpcRateMatch method), 531
run() (aerial.phy5g.algorithms.trt_engine.TrtEngine method), 521
run() (aerial.phy5g.pdsch.pdsch_tx.PdschTx method), 524
run() (aerial.phy5g.pusch.pusch_rx.PuschRx method), 527

save_pickle() (in module aerial.util.data), 543
set_num_iterations()
(aerial.phy5g.ldpc.decoder.LdpcDecoder method), 529
set_profiling_iterations()
(aerial.phy5g.ldpc.encoder.LdpcEncoder method), 531
set_profiling_iterations()
(aerial.phy5g.ldpc.rate_match.LdpcRateMatch method), 532
set_puncturing() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 531
set_throughput_mode()
(aerial.phy5g.ldpc.decoder.LdpcDecoder method), 530
SrsCellPrms (class in aerial.phy5g.algorithms.srs_channel_estimator), 521
SrsChannelEstimator (class in aerial.phy5g.algorithms.srs_channel_estimator), 523
SrsReport (class in aerial.phy5g.algorithms.srs_channel_estimator), 523

TrtEngine (class in aerial.phy5g.algorithms.trt_engine), 521
TrtTensorPrms (class in aerial.phy5g.algorithms.trt_engine), 520

UeSrsPrms (class in aerial.phy5g.algorithms.srs_channel_estimator), 522