Running cuBB End-to-End

Table of contents

List of Figures

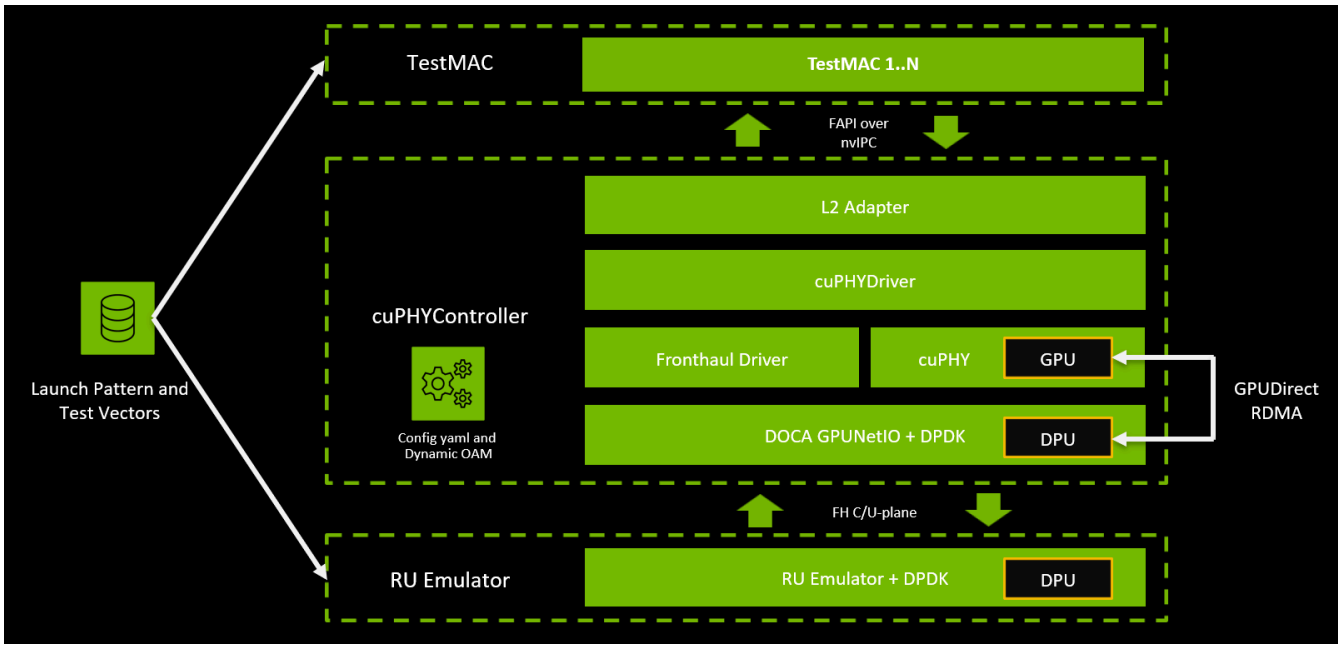
Beyond the cuPHY layer 1 PHY software and its standalone examples, this section describes how to build and run the cuBB software components shown in the block diagram below.

- 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.
    - It validates UL-C section information for PUCCH/PUSCH/PRACH/SRS and responds with corresponding UL U-plane.

- 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*.

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:

```
$ cd /opt/nvidia/cuBB $ export cuBB_SDK=$(pwd)
```

2. Create and navigate to the build directory:

```
$ mkdir build && cd build
```

> **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 enabled) if this field is not present in the l2_adapter yaml. It is no longer necessary to run cmake with the -*DDYNAMIC_SFN_SLOT=ON/OFF* flag. The same binaries can be used in RU emulator configuration and eLSU/O-RU configuration. The DYNAMIC_SFN_SLOT option has been removed entirely from CMakeLists.txt since Aerial 23-4 release.

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>
```

- To prepare the cmake build for Ninja compilation, you could add *-G Ninja* as part of the cmake command. All of the following *make* commands can be replaced with *ninja*

Here is the table of supported build variants:

RU Type \ Build Options	**FAPI 10.02: Default (no build flag)**	**Enable FAPI 10.04 fields: -DSCF_FAPI_10_04=ON**	**Enable TestMode: -DENABLE_CONFORMANCE_TM_PDSCH_PDC**
RU emulator: No build flag but set *enableTickDynamicSfnSlot: 0* in l2_adapter yaml	cmake ..	cmake .. -DSCF_FAPI_10_04=ON	cmake .. -DENABLE_CONFORMANCE_TM_PDSCH_PDC

Keysight eLSU: Default (no build flag)	cmake .. *	N/A	N/A

> **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).

Additional Build Options

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

```
$ cmake .. -DSCF_FAPI_10_04=ON -
DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON
```

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

```
$ cmake .. -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 .. -DENABLE_UL_HEAVY_TDD=ON
```

To build all Aerial cuBB components, use these commands:

```
$ cd ${cuBB_SDK}/build $ make -j $(nproc --all)
```

To build only the cuPHY, use these commands:

```
$ cd ${cuBB_SDK}/build/cuPHY $ make -j $(nproc --all)
```

To build only the Test MAC, use these commands:

```
$ cd ${cuBB_SDK}/build/cuPHY-CP/testMAC $ make -j $(nproc --all)
```

To build only the cuPHY controller, use these commands:

```
$ cd ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller $ make -j $(nproc --all)
```

To build only the cuPHY driver, use these commands:

```
$ cd ${cuBB_SDK}/build/cuPHY-CP/cuphydriver $ make -j $(nproc --all)
```

To build only the RU emulator, use these commands:

```
$ cd ${cuBB_SDK}/build/cuPHY-CP/ru-emulator $ make -j $(nproc --all)
```

To compile the Aerial code in the container on a devkit or DellR750 machine that has isolcpus restricting cores, you can override isolcpus using the following command:

```
$ sudo chrt -r 1 sudo -u aerial taskset -c 10-20 ninja # for ninja-based builds, or $ sudo chrt -r 1 sudo -u aerial taskset -c 10-20 make -j # for make-based builds
```

The example commands use cores 10-20.

Updating Configuration Files for End-to-End

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

Server #1 (to Run TestMAC and cuPHYController)

Check and edit the following parameters in the `.yaml` file:

- Edit the NIC PCIe address to match the NIC hardware PCIE address. The Aerial Devkit server uses address 0000:b5:00.0:

```
$ sed -i "s/ nic:.*/ nic: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08.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.

```
GPU0 mlx5_0 mlx5_1 CPU Affinity GPU0 X PIX PIX 0-23 mlx5_0 PIX X PIX mlx5_1 PIX PIX X
```

The meaning of PIX is:

```
X = Self SYS = Connection traversing PCIe and the SMP interconnect between NUMA nodes (e.g., QPI/UPI) 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.

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:

> *# 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:

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

Replace the MAC address with the MAC address of the NIC port used in Server#1:

```
$ 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.

```
$ sudo lshw -c network -businfo Bus info Device Class Description
==========================================================
pci@0000:05:00.0 eno1 network I210 Gigabit Network Connection pci@0000:06:00.0 enp6s0 network I210 Gigabit Network Connection pci@0000:b5:00.0 ens6f0 network MT2892 Family [ConnectX-6 Dx] pci@0000:b5:00.1 ens6f1 network MT2892 Family [ConnectX-6 Dx] vethdf87878 network Ethernet interface
```

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

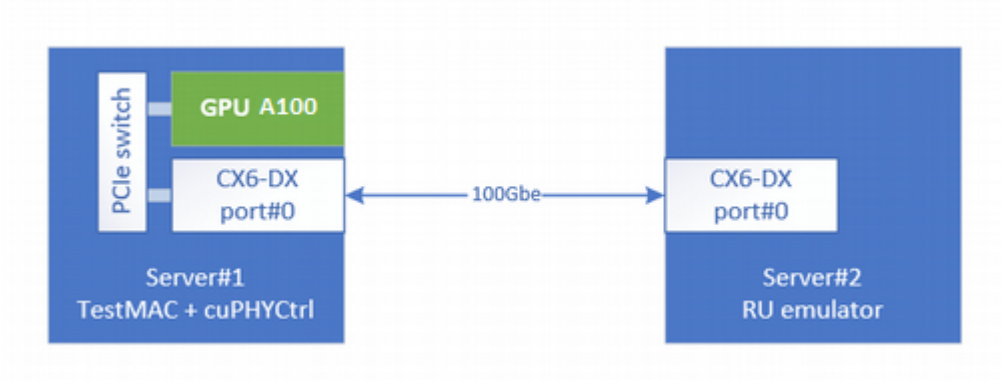
```
$ ifconfig -a ... 68: ens6f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1514 qdisc mq state UP group default qlen 1000 link/ether 1c:34:da:ff:ff:fe brd ff:ff:ff:ff:ff:ff inet6 fe80::bace:f6ff:fe33:fe16/64 scope link valid_lft forever preferred_lft forever 69: ens6f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000 link/ether 1c:34:da:ff:ff:ff brd ff:ff:ff:ff:ff:ff inet6 fe80::bace:f6ff:fe33:fe17/64 scope link valid_lft forever preferred_lft forever
```

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

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:

> $ 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:

> $ 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

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

Execute the following command before running `cuphycontroller`, `test_mac`, and `ru_emulator`.

```
export LD_LIBRARY_PATH=/opt/mellanox/dpdk/lib/x86_64-linux-gnu:\
/opt/mellanox/doca/lib/x86_64-linux-gnu
```

In some system environments, the export might not work; in this case, add the value before the command:

```
sudo -E LD_LIBRARY_PATH=/opt/mellanox/dpdk/lib/x86_64-linux-gnu:\
/opt/mellanox/doca/lib/x86_64-linux-gnu \ $cuBB_SDK/build/cuPHY-
CP/cuphycontroller/examples/cuphycontroller_scf
```

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:

```
sudo $cuBB_SDK/build/cuPHY-CP/scfl2adapter/scf_app/cuphycontroller\
/l2_adapter_cuphycontroller_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
```

Running testMAC + cuPHYController_SCF + RU Emulator

There are several common configurations.

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

- cuphycontroller_F08.yaml
- cuphycontroller_F08_R750.yaml
- cuphycontroller_F08_CG1.yaml
- cuphycontroller_nrSim_SCF.yaml

Server#1

Replace 0000:b5:00.0 with the PCIe address of NIC port for use to connect to Server#2 (RU emulator):

```
sed -i "s/ nic:.*/ nic: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_*.yaml
```

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

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

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

```
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/cuphycontroller/config/cuphycontroller_F08_R750.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/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/ channel_segment_timelines:.*/ channel_segment_timelines: 1/"${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.

Server#2

Replace 0000:b5:00.0 with the PCIe address of NIC for use. 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 sed -i "s/peerethaddr.*/peerethaddr: 1c:34:da:ff:ff:fe/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
```

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
```

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
```

Execute the following command to export the dpdk link path before running cuphycontroller and test_mac.

```
export LD_LIBRARY_PATH=$cuBB_SDK/gpu-dpdk/build/install/lib/x86_64-linux-gnu
```

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

```
sudo -E LD_LIBRARY_PATH=$cuBB_SDK/gpu-dpdk/build/install/lib/x86_64-linux-gnu
./cuphycontroller_scf
```

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.

```
15:15:56.251444 [FH.FLOW] [/opt/nvidia/cuBB/cuPHY-CP/aerial-fh-
driver/lib/flow.cpp:201] cuda failed with invalid argument
```

When running on an 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 -E numactl -N 1 -m 1 ./test_mac x
```

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 A100x R750 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:

```
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`:

```
pusch_tdi: 1 pusch_cfo: 1 pusch_to: 1 pusch_dftsofdm: 0 pusch_select_eqcoeffalgo: 1 puxch_polarDcdrListSz: 8
```

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

```
pusch_nMaxPrb: 36
```

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

```
# 4 UL Antenna config sed -i "s/eAxC_UL: \[8,0\]/eAxC_UL: \[8,0,1,2\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_UL: \[1,2\]/eAxC_UL: \[1,2,4,9\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_UL: \[0,1\]/eAxC_UL: \[0,1,2,3\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_id_pucch: \[8, 0\]/eAxC_id_pucch: \[8, 0, 1, 2\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pucch: \[1, 2\]/eAxC_id_pucch: \[1, 2, 4, 9\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pucch: \[0, 1\]/eAxC_id_pucch: \[0, 1, 2, 3\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pusch:
```

```
\[8, 0\]/eAxC_id_pusch: \[8, 0, 1, 2\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pusch: \[1, 2\]/eAxC_id_pusch: \[1, 2, 4, 9\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pusch: \[0, 1\]/eAxC_id_pusch: \[0, 1, 2, 3\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

23-4 supports early HARQ processing on AX800 setups. For the C and D variations of pattern 59 and 60, enable early HARQ processing with the following configurations:

```
# For early HARQ sed -i 's/uciIndPerSlot :.*/uciIndPerSlot : 2/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml sed -i "s/pusch_subSlotProcEn:.*/pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml # For early non HARQ sed -i 's/uciIndPerSlot :.*/uciIndPerSlot : 0/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml sed -i "s/pusch_subSlotProcEn:.*/pusch_subSlotProcEn: 0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

```
# For Enhanced L1 - L2 Interface sed -i 's/uciIndPerSlot :.*/uciIndPerSlot : 2/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml sed -i "s/pusch_subSlotProcEn:.*/pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/ mCh_segment_proc_enable:.*/ mCh_segment_proc_enable: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/ channel_segment_timelines:.*/ channel_segment_timelines: 1/"${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.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 RoyB 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:

 # 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 beamforming:

 # cuphycontroller configs for PUSCH conformance flags: sed -i "s/pusch_tdi:.*/pusch_tdi: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/pusch_cfo:.*/pusch_cfo: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/pusch_to:.*/pusch_to: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/puxch_polarDcdrListSz:.*/puxch_polarDcdrListSz: 8/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml *# RU emulator beamforming validation config* sed -i "s/enable_beam_forming:.*/enable_beam_forming: 1/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml

To test 4T 4R TDD 7 beams series 59 and 60 with 80 slot patterns have been generated:

- Series 59: 12C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, same as the old pattern 46, but slot 5 has 4 PRACH occasions:
    - PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
    - PUSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (37 PRBs / UEG when having 4 PRACH, 39 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: 24 UE 1 bit each (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 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
        - 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 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 60: 7 beams, 100 MHz (273 PRBs), 16C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, same as the old pattern 47, but slot 5 has 4 PRACH occasions:
    - PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)

- PUSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (14 PRBs / UEG when having 4 PRACH, 16 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: 24 UE 1 bit each (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 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

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.

```
sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu numactl -N 1 -m 1 ./cuphycontroller_scf F08_R750 sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu numactl -N 1 -m 1 ./test_mac F08 4C
```

```
59 sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu ./ru_emulator F08 4C 59
```

```
21:40:26.213585 WRN 2231 0 [RU] Cell 0 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.91% UL_C_ON 100.00% |Seconds 459 21:40:26.213591 WRN 2231 0 [RU] Cell 1 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.94% UL_C_ON 100.00% |Seconds 459 21:40:26.213595 WRN 2231 0 [RU] Cell 2 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.92% UL_C_ON 100.00% |Seconds 459 21:40:26.213599 WRN 2231 0 [RU] Cell 3 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.95% UL_C_ON 100.00% |Seconds 459
```

On R750 RoyB DU system F08 4C with pattern 60 (average pattern):

```
sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu numactl -N 1 -m 1 ./cuphycontroller_scf F08_R750 sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu numactl -N 1 -m 1 ./test_mac F08 4C 60 sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu ./ru_emulator F08 4C 60
```

```
22:01:12.039024 WRN 2375 0 [RU] Cell 0 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00% |Seconds 471 22:01:12.039030 WRN 2375 0 [RU] Cell 1 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00% |Seconds 471 22:01:12.039034 WRN 2375 0 [RU] Cell 2 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps 400 Slots | PBCH 200 |
```

```
PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00% |Seconds 471 22:01:12.039037 WRN 2375 0 [RU] Cell 3 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps 400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH 200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00% |Seconds 471
```

- Series 57: 100MHz (273 PRBs), 8C ave cells, 4 UL stream, OTA
    - The Pattern 57 is same as nrSim TC 90602. Because its 4 UL Streams 4 Antenna config must be used, the following are the configuration settings:

```
sed -i "s/eAxC_UL: \[8,0\]/eAxC_UL: \[8,0,1,2\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_UL: \[1,2\]/eAxC_UL: \[1,2,4,9\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_UL: \[0,1\]/eAxC_UL: \[0,1,2,3\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_id_pusch: \[8, 0\]/eAxC_id_pusch: \[8, 0, 1, 2\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pusch: \[1, 2\]/eAxC_id_pusch: \[1, 2, 4, 9\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pusch: \[0, 1\]/eAxC_id_pusch: \[0, 1, 2, 3\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pucch: \[8, 0\]/eAxC_id_pucch: \[8, 0, 1, 2\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pucch: \[1, 2\]/eAxC_id_pucch: \[1, 2, 4, 9\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sed -i "s/eAxC_id_pucch: \[0, 1\]/eAxC_id_pucch: \[0, 1, 2, 3\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml
```

On R750 A100x DU system F08 8C with pattern 57 (average pattern):

```
sudo -E ./cuphycontroller_scf F08_R750 sudo ./test_mac F08 8C 57 sudo ./ru_emulator F08 8C 57
```

This is the expected MAC throughput:

> 13:34:55.520460 WRN [MAC.FAPI] Cell 0 | DL 208.35 Mbps 1600 Slots | UL 2.25 Mbps 400 Slots | Prmb 400 | HARQ 10000 | SR 0 | CSI1 400 | CSI2 400 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520461 WRN [MAC.FAPI] Cell 1 | DL 416.69 Mbps 1600 Slots | UL 4.51 Mbps 400 Slots | Prmb 400 | HARQ 10400 | SR 0 | CSI1 800 | CSI2 800 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 2 | DL 625.04 Mbps 1600 Slots | UL 8.91 Mbps 400 Slots | Prmb 400 | HARQ 10800 | SR 0 | CSI1 1200 | CSI2 1200 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 3 | DL 833.38 Mbps 1600 Slots | UL 15.19 Mbps 400 Slots | Prmb 400 | HARQ 11200 | SR 0 | CSI1 1600 | CSI2 1600 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 4 | DL 1041.73 Mbps 1600 Slots | UL 21.46 Mbps 400 Slots | Prmb 400 | HARQ 11600 | SR 0 | CSI1 2000 | CSI2 2000 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 5 | DL 1250.07 Mbps 1600 Slots | UL 27.74 Mbps 400 Slots | Prmb 400 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 6 | DL 1458.42 Mbps 1600 Slots | UL 35.78 Mbps 400 Slots | Prmb 400 | HARQ 12400 | SR 0 | CSI1 2800 | CSI2 2800 | SRS 0 | ERR 0 | INV 0 | Slots 2000 13:34:55.520462 WRN [MAC.FAPI] Cell 7 | DL 1666.76 Mbps 1600 Slots | UL 43.82 Mbps 400 Slots | Prmb 400 | HARQ 12800 | SR 0 | CSI1 3200 | CSI2 3200 | SRS 0 | ERR 0 | INV 0 | Slots 2000

Simultaneous FH Port Test Configs with RU Emulator

The following TC needs both FH ports:

- BFP14 8C 59
- BFP14 16C 60
- BFP14 12C 50 (80MHz)

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 AX800 and 1 RU server
    - AX800 P0 <-> RU P0

    - AX800 P1 <-> RU P1
- 1 AX800 and 2 RU server
    - AX800 P0 <-> RU 1 P0
    - AX800 P1 <-> RU 2 P0

cuPHYController configuration:

```
nics: - nic: 0000:cc: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:cc: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:

```
cells: - name: O-RU 0 [...] nic: 0000:cc:00.0 - name: O-RU 1 [...] nic: 0000:cc:00.1 - name: O-RU 2 [...] nic: 0000:cc:00.0 - name: O-RU 3 [...] nic: 0000:cc: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:

```
nics: - nic_interface: 0000:b5:00.0 - nic_interface: 0000:b5:00.1 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:

```
cell_configs: - name: "Cell1" peer: 0 nic: 0 - name: "Cell2" peer: 1 nic: 1 - name: "Cell3" peer: 0 nic: 0 - name: "Cell4" peer: 1 nic: 1
```

For the second topology with 2 RU Emulator systems, an option is to run with the same configuration file, but half of the cells would not have traffic. Optionally you could remove the second nic_interface and peerethaddr definitions and only use index 0.

Running the nrSim Test Cases

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*

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*

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*

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_csip2_v3.*/enable_csip2_v3: 1/" ${cuBB_SDK}/cuPHY-

CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml 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 sudo ./test_mac nrSim 7599 --channels PUSCH sudo ./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 sudo ./test_mac nrSim 7600 --channels PUSCH sudo ./ru_emulator nrSim 7600 --channels PUSCH *# Expect testMAC to report 100 PUSCH and 100 CSIP2 per second*

PRACH

 # Restart MPS sudo -E ./cuphycontroller_scf nrSim_SCF sudo ./test_mac nrSim 5901 --channels PRACH sudo ./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 sudo ./test_mac nrSim 5013 --channels PRACH sudo ./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 15:57:41.161878 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 15:57:41.161904 I [DRV.PRACH] SFN 599.01 *#0 prmbIndex 14 prmbDelay 0.000000 prmbPower -4.193221* 15:57:41.161906 I [DRV.PRACH] SFN 599.01 *#1 prmbIndex 17 prmbDelay 0.000000 prmbPower -4.011869* 15:57:41.161909 I [DRV.PRACH] SFN 599.01 *#2 prmbIndex 20 prmbDelay 0.000000 prmbPower -3.471422* 15:57:41.161912 I

> [DRV.PRACH] SFN 599.01 *#3 prmbIndex 23 prmbDelay 0.000000 prmbPower -3.552692* 15:57:41.161924 I [DRV.PRACH] RO 2 SFN 599.01 Preambles num detected 4 15:57:41.161927 I [DRV.PRACH] SFN 599.01 *#0 prmbIndex 32 prmbDelay 0.000000 prmbPower -4.954414* 15:57:41.161930 I [DRV.PRACH] SFN 599.01 *#1 prmbIndex 35 prmbDelay 0.000000 prmbPower -3.706564* 15:57:41.161933 I [DRV.PRACH] SFN 599.01 *#2 prmbIndex 26 prmbDelay 0.000000 prmbPower -4.333083* 15:57:41.161935 I [DRV.PRACH] SFN 599.01 *#3 prmbIndex 29 prmbDelay 0.000000 prmbPower -3.994442* 15:57:41.161945 I [DRV.PRACH] RO 3 SFN 599.01 Preambles num detected 4 15:57:41.161947 I [DRV.PRACH] SFN 599.01 *#0 prmbIndex 38 prmbDelay 0.000000 prmbPower -3.341729* 15:57:41.161950 I [DRV.PRACH] SFN 599.01 *#1 prmbIndex 41 prmbDelay 0.000000 prmbPower -4.641103* 15:57:41.161952 I [DRV.PRACH] SFN 599.01 *#2 prmbIndex 44 prmbDelay 0.000000 prmbPower -4.189767* 15:57:41.161955 I [DRV.PRACH] SFN 599.01 *#3 prmbIndex 47 prmbDelay 0.000000 prmbPower -4.946166*

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*

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*

Precoding

> *# Below steps are applicable to precoding test for PDSCH, PDCCH, PBCH, and CSI_RS # In l2_adapter_config_nrSim_SCF.yaml, 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_SCF.yaml *# Restart MPS* sudo -E ./cuphycontroller_scf nrSim_SCF 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_SCF.yaml *# In ru-emulator/config/config.yaml, set dl_approx_validation to 1* sed -i -z "s/dl_approx_validation: 0/dl_approx_validation: 1/1" $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: 0/1" $cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml

PUCCH HARQ

 # Restart MPS sudo -E ./cuphycontroller_scf nrSim_SCF 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*

PUCCH Format 2

 # Restart MPS sudo -E ./cuphycontroller_scf nrSim_SCF 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*

PUCCH HARQ/SR

 # Restart MPS sudo -E ./cuphycontroller_scf nrSim_SCF 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*

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*

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*

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 -E ./ru_emulator nrSim 8301 --channels SRS or ./ru_emulator nrSim 8301 (default support all channels) sudo -E ./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. # 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 -E ./ru_emulator nrSim 8302 --channels SRS or ./ru_emulator nrSim 8302 (default support all channels) sudo -E ./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.*

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*

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*

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*

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] sed -i s/"eAxC_id_pusch: \\[8,0,1,2\\]/eAxC_id_pusch: \\[8,0,1,2,3,4,5,6\\]/1" $cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml *# 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" $cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

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 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/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/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.yaml cp
${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_900xx.yaml
${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml python3
auto_TestMacConfig.py -t ../../cuPHY-
CP/testMAC/testMAC/test_mac_config.yaml.orig -c 900xx -p devkit -o ../../cuPHY-
CP/testMAC/testMAC/test_mac_config.yaml # Restart MPS sudo -E
./cuphycontroller_scf nrSim_SCF sudo ./ru_emulator nrSim 900xx --channels 0x1ff
sudo ./test_mac nrSim 900xx --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
```

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 .. -DSCF_FAPI_10_04=ON make -j $(nproc --all)
```

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
- TVnr_3852_gNB_FAPI_s0.h5
- TVnr_7853_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/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/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.yaml cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_900xx.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml python3 auto_TestMacConfig.py -t ../../cuPHY-

```
CP/testMAC/testMAC/test_mac_config.yaml.orig -c 900xx -p devkit -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/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```

64T64R SRS + Dynamic Beamforming Weights Test Cases

Here are the steps to build and run the 64T6R SRS and dynamic beamforming weights related tests.

Build options:

```
cmake .. -DSCF_FAPI_10_04=ON make -j $(nproc --all)
```

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
- TVnr_3874_gNB_FAPI_s0.h5

100 MHz UL 2+2 layers (2 UE's) full allocation:

- TVnr_9245_gNB_FAPI_s0.h5
- TVnr_3874_gNB_FAPI_s0.h5

100 MHz DL 1+1+1+1 layers (4 UE's) full allocation:

- TVnr_9246_gNB_FAPI_s0.h5
- TVnr_3875_gNB_FAPI_s0.h5

100 MHz UL 1+1+1+1 layers (4 UE's) full allocation:

- TVnr_9247_gNB_FAPI_s0.h5
- TVnr_3875_gNB_FAPI_s0.h5

100 MHz DL 2+2+2+2 layers (4 UE's) full allocation:

- TVnr_9248_gNB_FAPI_s0.h5
- TVnr_3876_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/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/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.yaml cp
${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_900xx.yaml
${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml python3
auto_TestMacConfig.py -t ../../cuPHY-
CP/testMAC/testMAC/test_mac_config.yaml.orig -c 900xx -p devkit -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/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-
CP/testMAC/testMAC/test_mac_config.yaml
```

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:

> *# 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 canmay 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] 07:32:09.407979 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind

> PDU2 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.

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:

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/pcp.*/pcp: 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/l2_adapter_config_P5G.yaml sed -i "s/enableTickDynamicSfnSlot.*/enableTickDynamicSfnSlot: 0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/l2_adapter_config_P5G_R750.yaml
```

Server#2

Replace `0000:b5:00.0` with the PCIe address of the NIC fo 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 sed -i "s/eAxC_DL: \[8,0,1,2\]/eAxC_DL: \[0,8,1,9\]/1" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_UL: \[8,0,1,2\]/eAxC_UL: \[0,8,1,9\]/1" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml sed -i "s/eAxC_prach_list: \[15,7,0,1\]/eAxC_prach_list: \[7,15,6,14\]/1" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
```

Run the emulator:

```
sudo ./ru_emulator P5G PRACH --channels 0x1FF
```

Run the cuPHY controller and the testMAC:

```
sudo -E ./cuphycontroller_scf P5G sudo ./test_mac P5G PRACH --channels 0x1FF
```

Expected RU emulator console:

```
00:44:12.169849 Cell 0 DL 0.17 Mbps 100 Slots | UL 0.03 Mbps 50 Slots | PBCH 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 25 00:44:13.169848 Cell 0 DL 0.17 Mbps 100 Slots | UL 0.03 Mbps 50 Slots | PBCH 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 26 00:44:14.169849 Cell 0 DL 0.17 Mbps
```

> 100 Slots | UL 0.03 Mbps 50 Slots | PBCH 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 27

Expected testMAC console:

> 00:44:11.565232 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots | Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 00:44:12.565230 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots | Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 00:44:13.565230 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots | Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0

Expected cuPHYController logs to be flooded with preamble detection:

> 00:44:11.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots CRC 0 (0) | Tick 2000 00:44:12.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots CRC 0 (0) | Tick 4000 00:44:13.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots CRC 0 (0) | Tick 6000

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.

> **Note**
>
> You need to modify the above mentioned reference files based on to your End-to-End setup.

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 20GB, a new file gets created (e.g. `phy.log`, `phy.log.1`, `phy.log.2` ... `phy.log.7`).
    1. The test MAC logs are named as `testmac.log`, `testmac.log.1`, etc.
    2. 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:

```
# 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.

> *# Transport settings for nvIPC* transport: type: shm app_config: pcap_enable: 1 pcap_cpu_core: -1 *# CPU core of background pcap log save thread* 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".*

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.

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/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/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.yaml cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_90019.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig -c 90019 -p devkit -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml # Restart MPS sudo -E ./cuphycontroller_scf 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

Mixed BFP9/BFP14

 # 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/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/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 cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_90020.yaml ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig -c 90020 -p devkit -o ../../cuPHY-

```
CP/testMAC/testMAC/test_mac_config.yaml # Restart MPS sudo -E ./cuphycontroller_scf nrSim_SCF sudo ./ru_emulator nrSim 90020 --channels 0x1ff sudo ./test_mac nrSim 90020 --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 throughput and passing criteria ExpectedSlots: Cell=0 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0 CSI_RS=0 SRS=0 ExpectedData: Cell=0 DL=0.000000 UL=41.797600 Prmb=0 HARQ=0 SR=0 CSI1=0 CSI2=0 ERR=0 INV=0 ExpectedSlots: Cell=1 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0 CSI_RS=0 SRS=0 ExpectedData: 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 07:09:34.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:34.600015 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:35.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:35.600014 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:36.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:36.600013 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:37.600008 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:37.600017 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:38.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:38.600014 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 |
```

> CSI1 0 | CSI2 0 | ERR 0 | INV 0 07:09:39.600008 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0

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.

```
# 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
```

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
```

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.

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.

```
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_FAPI_10_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:

```
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.

Tag	Field	Type	Description
0xA012	PNMeasurement	uint8_t	Post equalisation noise variance measurement Value: 0: Do not report 1: dBm
0xA014	PF_234_Interference	uint8_t	Interference power per UE. Value: 0: Do not report 1: dBm

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/testMAC/test_mac_config.yaml` file:

```
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:

```
- 333: "SCF.PHY" shm_level: 6 # Example: overlay shm_log_level for a tag
```

The following example shows results in the `phy.log` log:

```
05:22:56.350648 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: PUCCH interference Raw=1520 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.

```
03:30:11.983670 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: HARQ detection status 4 03:30:11.983671 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: UCI on PUSCH HARQ bitlen 2 03:30:11.983671 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: PUCCH F234 CSI Part1 detection status 4 CSI P1 bit len 10
```

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.

```
% 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 HDF5 "TVnr_7427_gNB_FAPI_s0.h5_BFP9" { DATASET "IND1" { DATATYPE H5T_COMPOUND { H5T_STD_U32LE "idxPdu"; H5T_STD_U32LE "type"; H5T_STD_U32LE "TbCrcStatus"; H5T_STD_U32LE "NumCb"; H5T_STD_U32LE "UL_CQI"; H5T_STD_U32LE "TimingAdvance"; H5T_STD_U32LE "RSSI"; H5T_STD_U32LE "RSRP"; H5T_STD_I16LE "sinrdB"; H5T_STD_I16LE "postEqSinrdB"; H5T_STD_I16LE "noiseVardB"; H5T_STD_I16LE "postEqNoiseVardB"; } Compare RSSI, RSRP, sinrdB and noiseVar values against the FAPI values in logs - >>> 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.

> 23:18:48.950917 D [SCF.PHY] Raw RSSI=6.020887 db ul_configured_gain=48.680000 23:18:48.950919 D [SCF.PHY] Raw SINR=40.000000 23:18:48.950920 D [SCF.PHY] Raw RSRP =-0.045194 db ul_configured_gain =48.680000 23:18:48.950938 D [SCF.PHY] Raw PE Noise variance=-40.000000 ul_configured_gain=48.680000

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 - F0UcisOutRef. 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=10.804343
```

Format 1 - F1UcisOutRef. Compare “SinrDB”, “RSSI”, and “RSRP” values against the corresponding value in logs.

```
SCF_FAPI_UCI_INDICATION 10.04: PUCCH : Raw SINR=25.059706 RSSI=-3.927727 RSRP=-9.948327
```

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
```

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:

```
# 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*

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 throughput reduces to 0 for about 5 seconds, then the procedure repeats.

sudo ./cuPHY-CP/ru-emulator/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

Terminate cuphycontroller Using a gRPC Message

Run the F08 E2E test case as usual:

```
sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu/ ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08 sudo LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu/ ${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/aerial_terminate_cuphycontroller.py
```

Verify that the cuphycontroller stops running, and that `aerial_terminate_cuphycontroller.py` prints the following output:

```
12:23:32 Terminating cuphycontroller... 12:23:36 cuphycontroller terminated successfully!
```

Update M-plane Parameters Using gRPC Message

Dynamically changing M-plane parameters via gPRC 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:

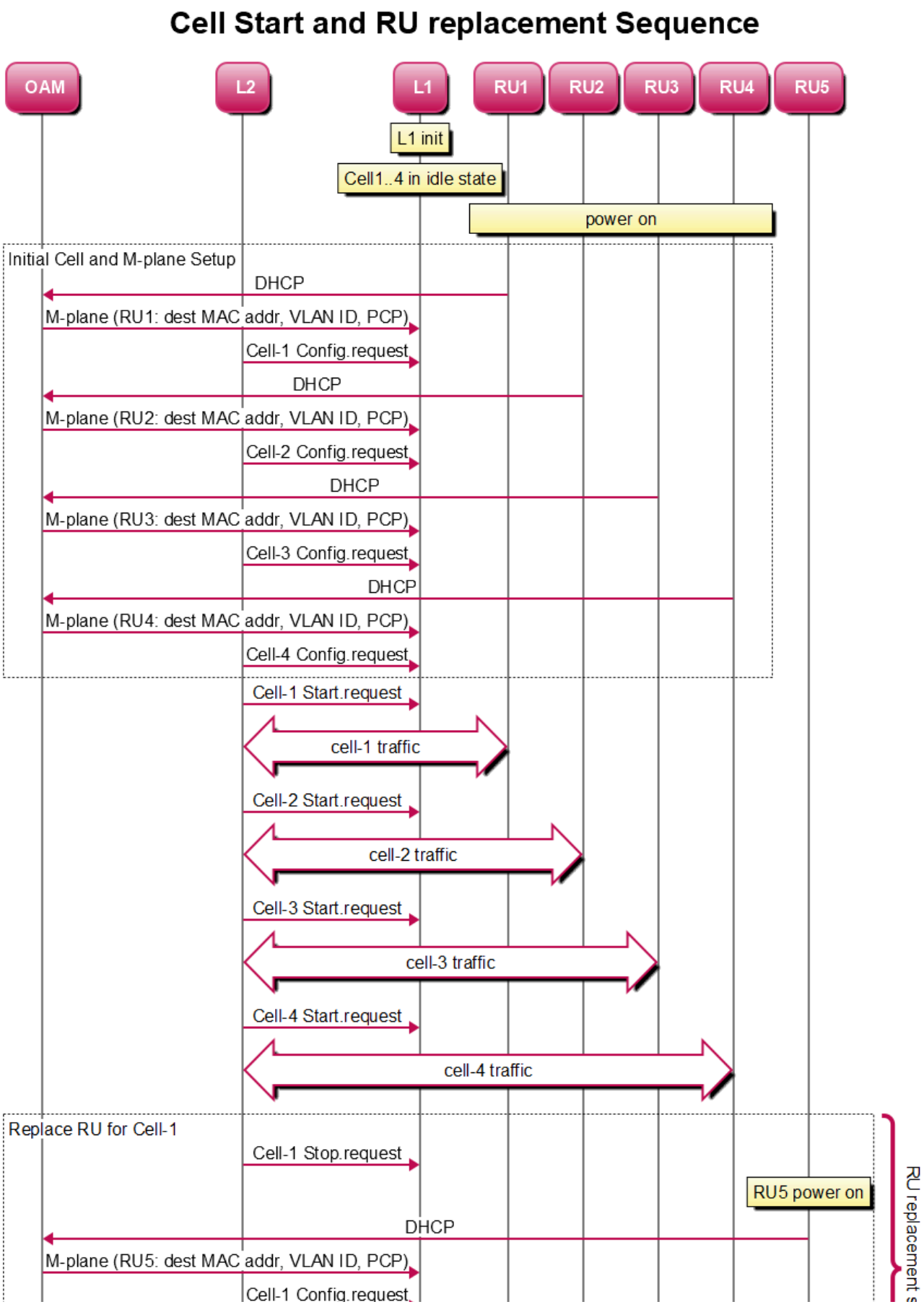
- **Initial cell and M-plane setup**: After launching cuphycontroller, L1 is initialized and all cells are in idle state to be configured. The max number of cells is defined by the `cell_group_num` parameter in the cuphycontroller YAML config. In the example sequence diagram, the OAM sends a gRPC message to update M-plane parameters

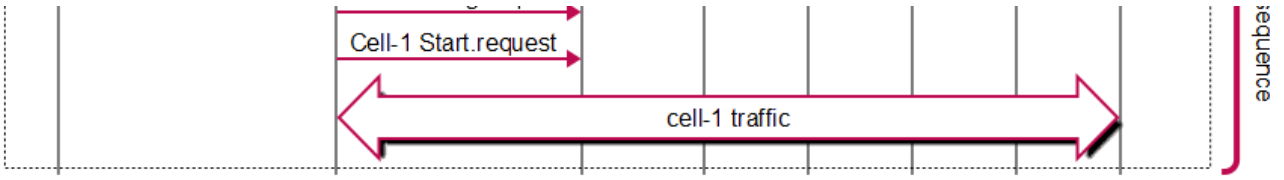
so that L1 gets the details to connect to the right RU for each cell. Then L2 sends `CONFIG.request` to configure the cell.

> **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.


Cell Start and RU replacement Sequence
OAM
L2
L1
RU1
RU2
RU3
RU4
RU5
L1 init
Cell1..4 in idle state
power on
Initial Cell and M-plane Setup
DHCP
M-plane (RU1: dest MAC addr, VLAN ID, PCP)
Cell-1 Config.request
DHCP
M-plane (RU2: dest MAC addr, VLAN ID, PCP)
Cell-2 Config.request
DHCP
M-plane (RU3: dest MAC addr, VLAN ID, PCP)
Cell-3 Config.request
DHCP
M-plane (RU4: dest MAC addr, VLAN ID, PCP)
Cell-4 Config.request
Cell-1 Start.request
cell-1 traffic
Cell-2 Start.request
cell-2 traffic
Cell-3 Start.request
cell-3 traffic
Cell-4 Start.request
cell-4 traffic
Replace RU for Cell-1
Cell-1 Stop.request
RU5 power on
DHCP
M-plane (RU5: dest MAC addr, VLAN ID, PCP)
Cell-1 Config.request
RU replacement s



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/'
```

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.

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 ./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 ./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:B3 E002 sudo ./ru_emulator F08 1C 59 sudo ./test_mac F08 1C 59

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 changedcell_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 ./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 ./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 E003 sudo ./ru_emulator F08 1C 59 sudo ./test_mac F08 1C 59
```

VLAN PCP 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 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.

```
sudo ./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.

```
sudo ./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
```

DST MAC + VLAN ID + PCP OAM Initial Update Test - Multi-Cells

```
# 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 dst_mac_addr: 20:20:20:20:20:A1 vlan: 3 pcp: 2 - cell_id: 2 dst_mac_addr: 20:20:20:20:20:A2 vlan: 4 pcp: 3 - cell_id: 3 dst_mac_addr: 20:20:20:20:20:A3 vlan: 5 pcp: 4 - cell_id: 4 dst_mac_addr: 20:20:20:20:20:A4 vlan: 6 pcp: 5
```

If you don’t perform the OAM update, the E2E test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo ./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.

```
sudo ./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
```

Dynamic OAM Update

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.

> 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] 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 ./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.

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.

> 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: 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)

> 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.

```
- 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:

```
sudo ./ru_emulator F08 1C_59_X2 sudo ./cuphycontroller_scf F08 sudo ./test_mac F08 1C 59
```

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.

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] 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

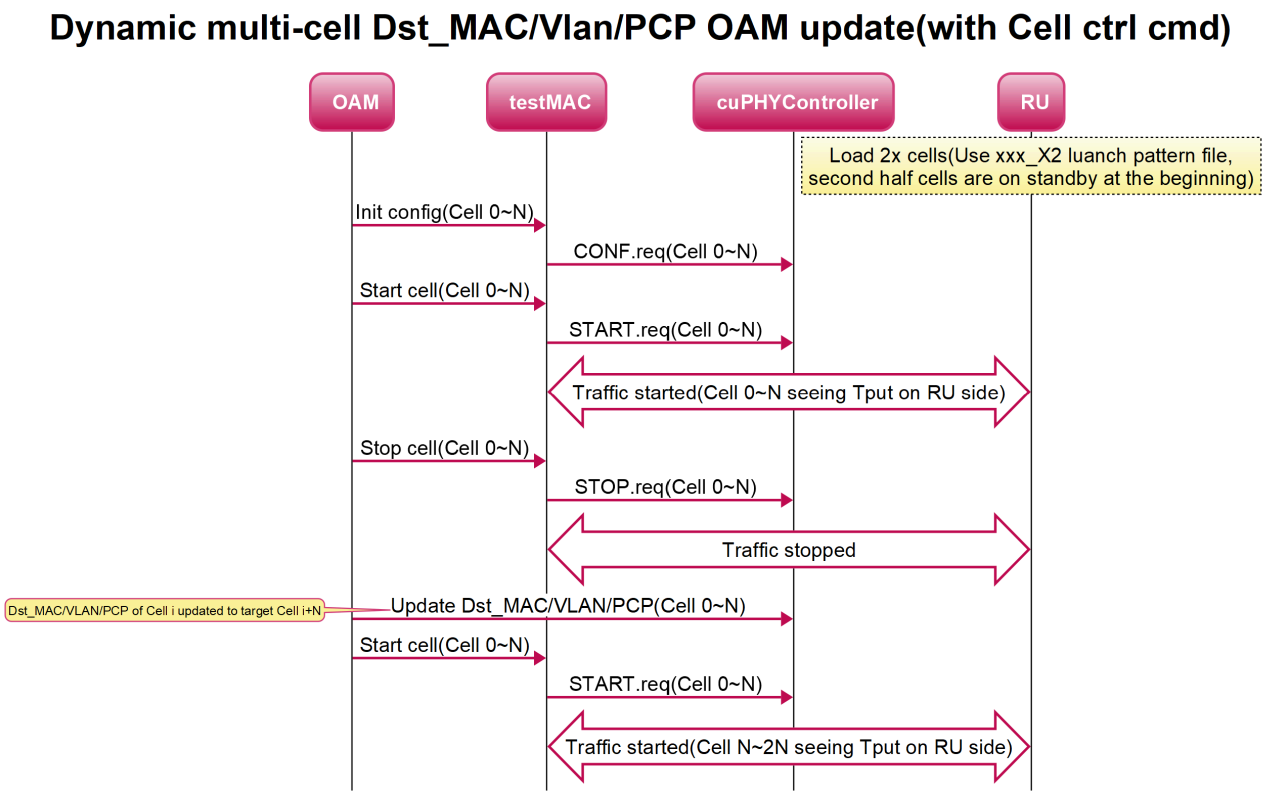
Run normal F08 Pattern 0 1C E2E test commands except change ru-emulator parameter "1C" to "1C_X2" The following example only shows the test case parameters, refer to F08 cases for full instructions:

> sudo ./ru_emulator F08 1C_59_X2 sudo ./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.

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.



Configuration update:

```
# Save original configuration before the test 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 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_.*/&\\[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 # Restore the
configuration after the test cp $cuBB_SDK/cuPHY-
CP/testMAC/testMAC/test_mac_config.yaml.orig $cuBB_SDK/cuPHY-
CP/testMAC/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:

```
sudo ./ru_emulator F08 4C_59_X2 sudo ./cuphycontroller_scf F08 sudo ./test_mac
F08 4C 59
```

Init CONF.req is sent to all cells (executed on DU server):

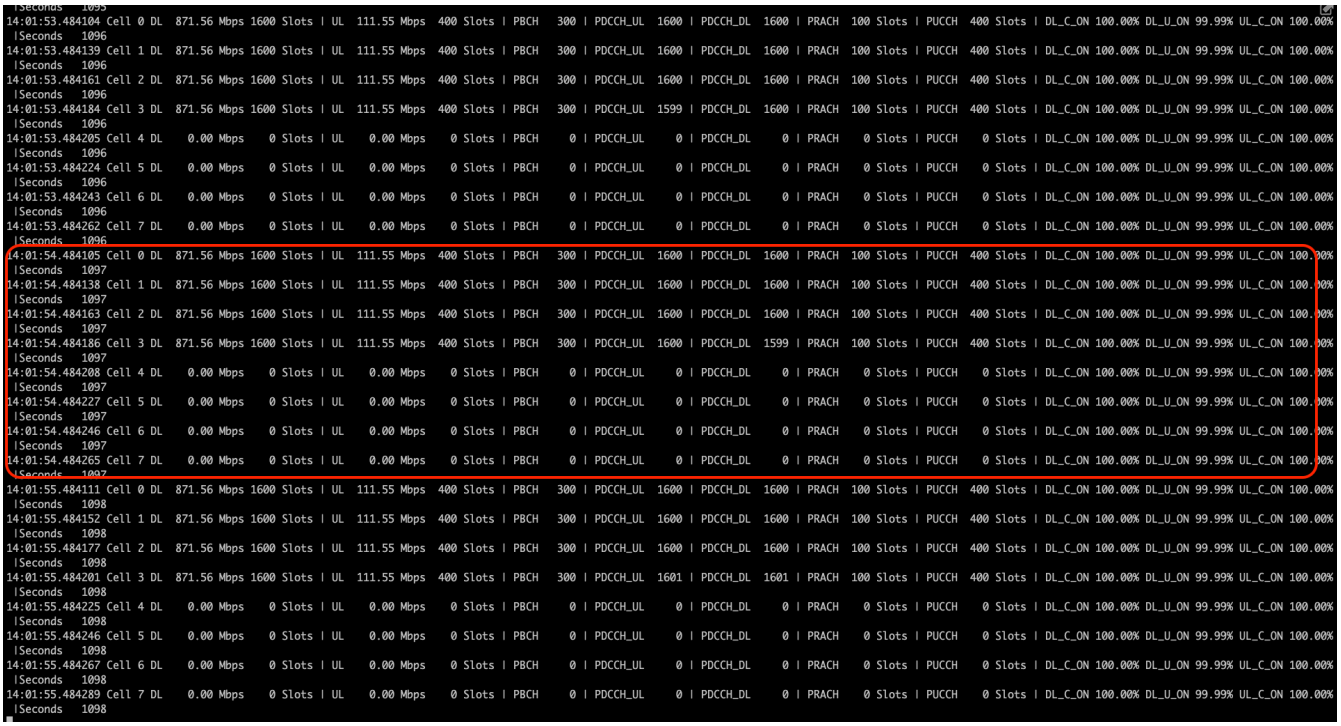
```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/ for i in {0..3}; do python3
$cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip
```

```
localhost --cell_id $i --cmd 3 && sleep 1; done;
```

START.req sent to all cells (executed on DU server):

```
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 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):

```
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 0 && sleep 1; done;
```

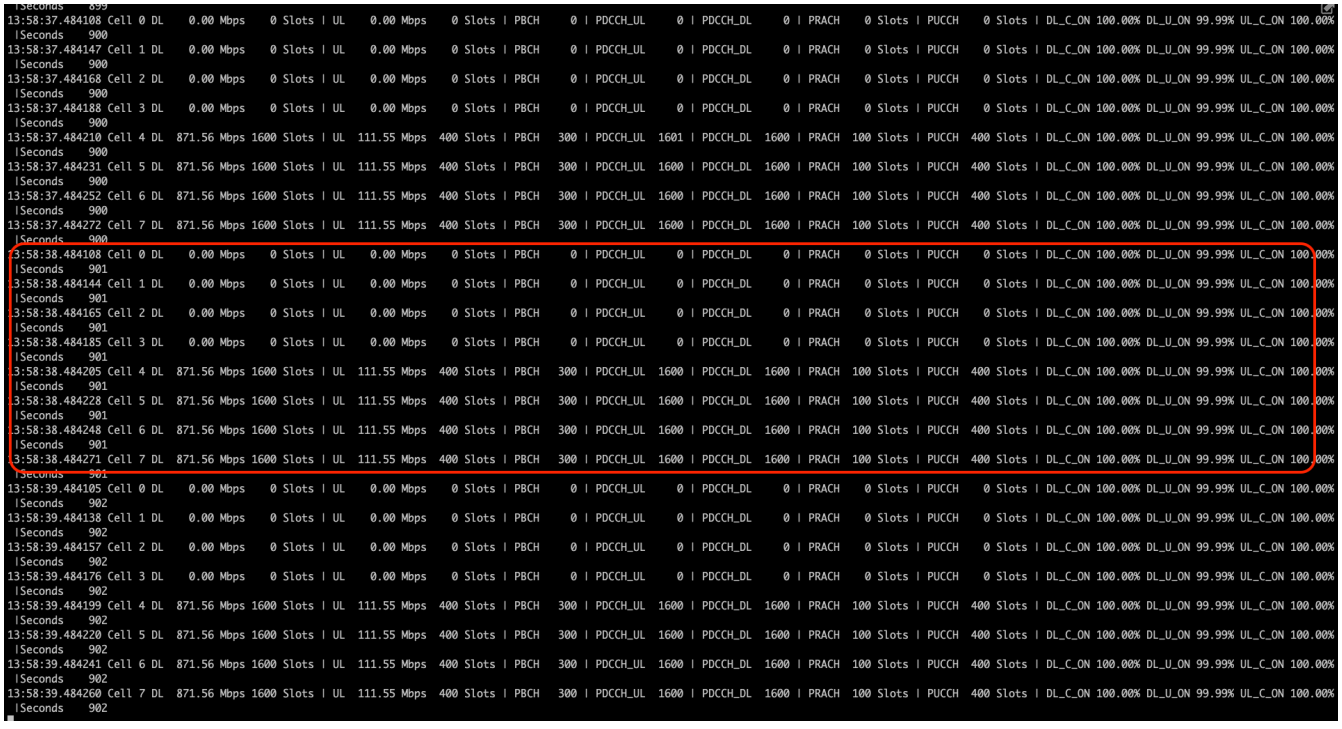
OAM update Cell i destination MAC updated to target cell i+4 on RU-Emulator side. That is: 0 4, 1 5, 2 6, 3->7) (executed on DU server):

```
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:27:05 E002 && sleep 1 python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 2 20:04:9B:9E:27:06 E002 && sleep 1 python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 3 20:04:9B:9E:27:07 E002 && sleep 1 python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 4 20:04:9B:9E:27:08 E002 && sleep 1
```

START.req sent to all cells (executed on DU server):

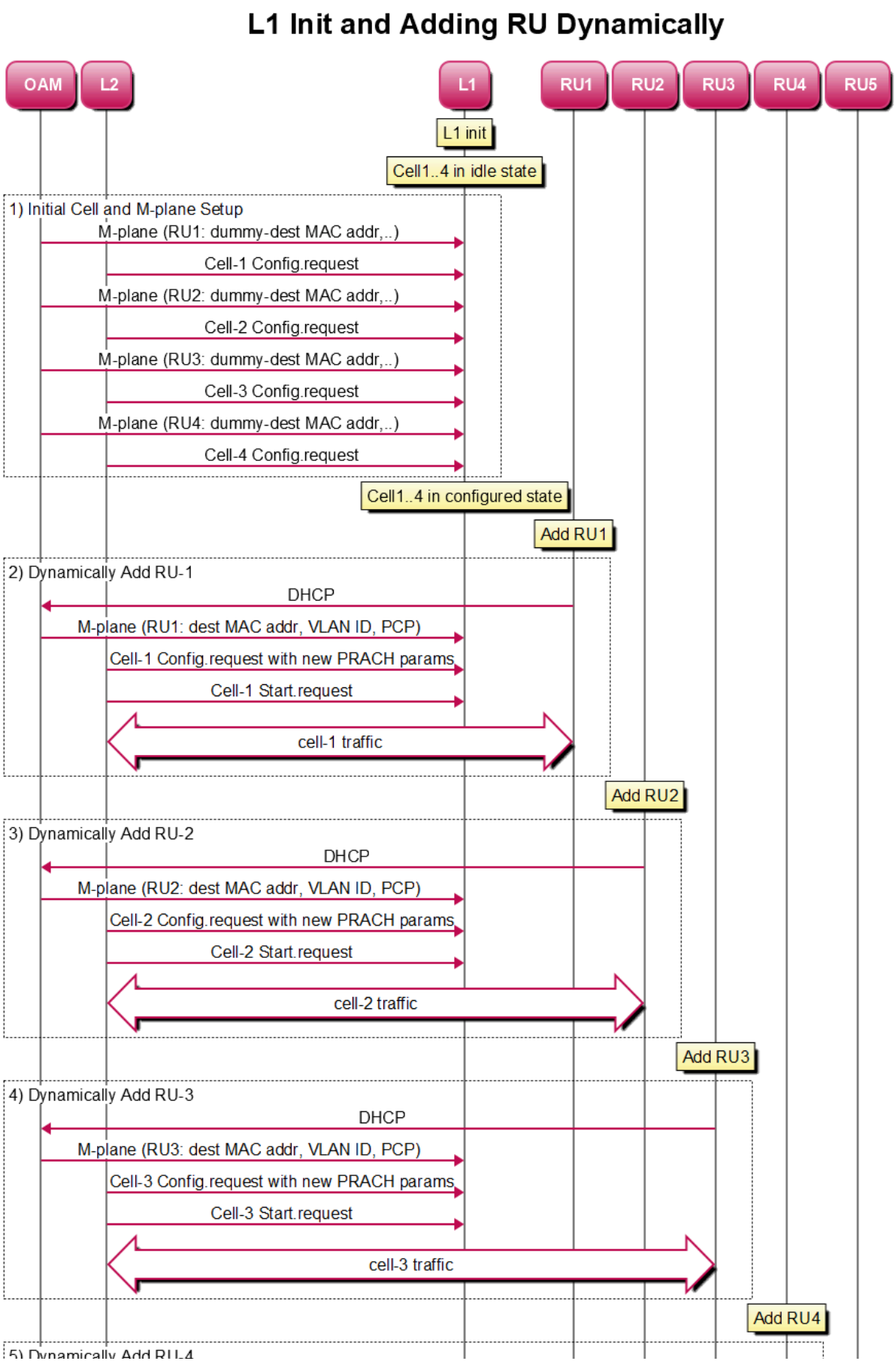
```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/ for i in {0..3}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --cmd 1 && sleep 1; done;
```

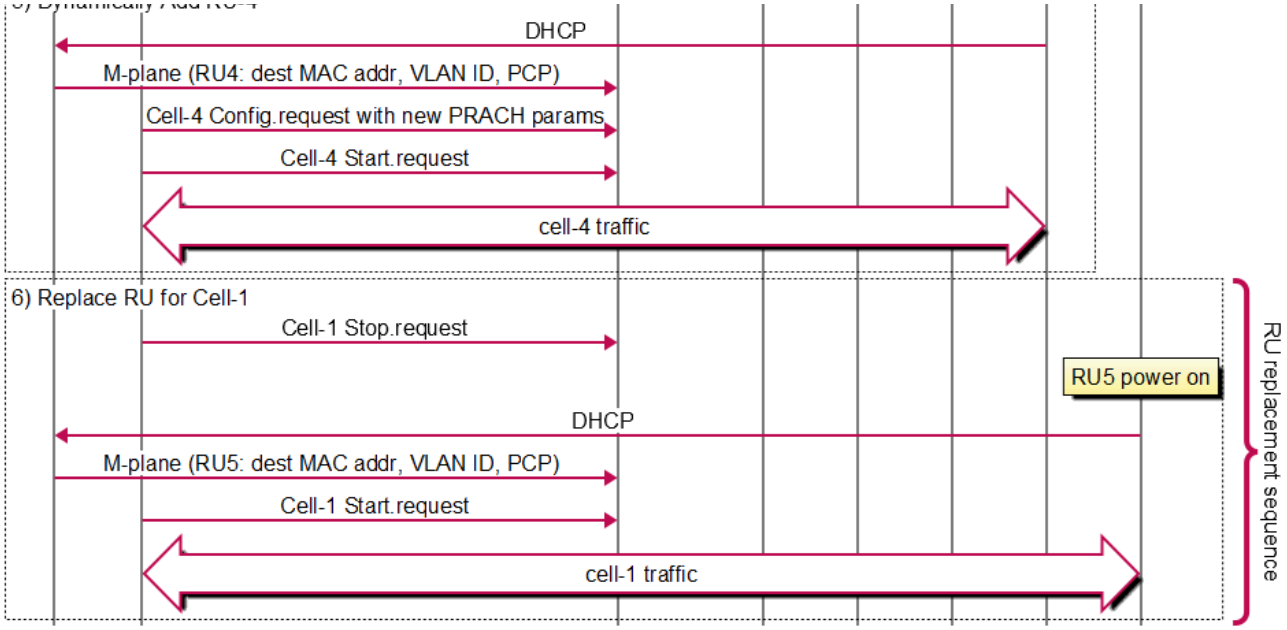
At this point, validate that the RU-Emulator sees cell 4~7 have tput:



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
OAM
L2
L1
RU1
RU2
RU3
RU4
RU5
L1 init
Cell1..4 in idle state
1) Initial Cell and M-plane Setup
M-plane (RU1: dummy-dest MAC addr,..)
Cell-1 Config.request
M-plane (RU2: dummy-dest MAC addr,..)
Cell-2 Config.request
M-plane (RU3: dummy-dest MAC addr,..)
Cell-3 Config.request
M-plane (RU4: dummy-dest MAC addr,..)
Cell-4 Config.request
Cell1..4 in configured state
Add RU1
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
Add RU2
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
Add RU3
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
Add RU4



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:

```
# 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.

```
cell_group: 1 cell_group_num: n fix_beta_dl: 0
```

For example, for 8C -

```
cell_group: 1 cell_group_num: 8 fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```
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:

```
sudo ./cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 9C 14 sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08 sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 9C 14
```

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.

> 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 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 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:

> 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 && sleep 1; done; //Send CONFIG.req for cell 0~7 python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id 0 --cmd 1 // Send START.req for cell-0

At this time you can see traffic running only for cell-0 on testMac, cuphycontroller and ru-emulator console:

> *# testMac console* 20:34:22.124683 C [MAC.SCF] cell_init: cell_id=0 fapi_type=SCF global_tick=-1 first_init=1 20:34:26.124793 C [MAC.SCF] cell_init: cell_id=1 fapi_type=SCF global_tick=-1 first_init=1 20:34:28.124858 C [MAC.SCF] cell_start: cell_id=0 fapi_type=SCF global_tick=-1 04:55:13.040024 Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 04:55:13.040037 Cell 1 | 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 04:55:13.040045 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 04:55:13.040051 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 04:55:13.040058 Cell 4 | 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 04:55:13.040065 Cell 5 | 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 04:55:13.040069 Cell 6 | 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 04:55:13.040074 Cell 7 | 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 04:55:13.040081 Cell 8 | 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 04:55:14.040025 Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 04:55:14.040037 Cell 1 | 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 04:55:14.040045 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 04:55:14.040049 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 04:55:14.040054 Cell 4 | 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 04:55:14.040061 Cell 5 | 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 04:55:14.040067 Cell 6 | 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 04:55:14.040071 Cell 7 | 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 04:55:14.040077 Cell 8 | 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* 04:55:13.040004 C [SCF.PHY] Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 (0) | Tick 142000 04:55:13.040018 C [SCF.PHY] Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040023 C [SCF.PHY] Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040027 C [SCF.PHY] Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040033 C [SCF.PHY] Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040037 C [SCF.PHY] Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040044 C [SCF.PHY] Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:13.040051 C [SCF.PHY] Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 142000 04:55:14.040005 C [SCF.PHY] Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 (0) | Tick 144000 04:55:14.040019 C [SCF.PHY] Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots

CRC 0 (0) | Tick 144000 04:55:14.040023 C [SCF.PHY] Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000 04:55:14.040028 C [SCF.PHY] Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000 04:55:14.040033 C [SCF.PHY] Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000 04:55:14.040040 C [SCF.PHY] Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000 04:55:14.040046 C [SCF.PHY] Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000 04:55:14.040050 C [SCF.PHY] Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots CRC 0 (0) | Tick 144000

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:

dst_mac_addr: 20:04:9B:9E:27:09

python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id 1 --cmd 2 --target_cell_id 8 //Send CONFIG.req for cell-1 with PRACH parameters read from TV for cell-8 and PCI of cell-8 python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 2 xx:xx:xx:xx:xx:xx E002 // Set VLAN-id and DST MAC address of cell-1 to point to VLAN-id & DST MAC address of cell-9 in cuphycontroller yaml file python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id 1 --cmd 1 // Send START.req for cell-1

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 12000 | SR 0 | CSI1 2400 | 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 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 20:35:01.560050 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400 20:35:01.560055 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 20:35:01.560060 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 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 20:35:02.560053 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400 20:35:02.560058 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 20:35:02.560063 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 20:35:03.560040 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 20:35:03.560051 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400 20:35:03.560056 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 20:35:03.560061 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 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 20:35:04.560054 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400 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* 20:35:00.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 62000 20:35:00.560014 C [SCF.PHY] Cell 1 | DL 752.17 Mbps 695 Slots | UL 104.81 Mbps

174 Slots CRC 0 (0) | Tick 62000 20:35:01.560004 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 64000 20:35:01.560012 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 64000 20:35:02.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 66000 20:35:02.560013 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 66000 20:35:03.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 68000 20:35:03.560012 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 68000 20:35:04.560006 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 70000 20:35:04.560013 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots CRC 0 (0) | Tick 70000 20:35:05.457529 C [SCF.PHY] Cell 0 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps 358 Slots CRC 0 (0) 20:35:05.457541 C [SCF.PHY] Cell 1 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps 358 Slots CRC 0 (0) 20:35:05.457676 C [SCF.PHY] Cell 0 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps 358 Slots CRC 0 (0) 20:35:05.457681 C [SCF.PHY] Cell 1 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps 358 Slots CRC 0 (0) *# ru-emulator console* 12:15:45.760099 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_C_ON 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_C_ON 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_C_ON 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_C_ON 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514 12:15:46.760060 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_C_ON 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_C_ON 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_C_ON 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514 12:15:46.760090 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_C_ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 514 12:15:47.760024 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_C_ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 515 12:15:47.760041 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760047 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760053 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760060 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760066 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760076 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760082 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_C_ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515 12:15:47.760089 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_C_ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 515 12:15:48.760023 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_C_ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 516

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.

```
sed -i "s/duplicate_config_all_cells.*/duplicate_config_all_cells: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08.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/test_mac_config.yaml sed -i "s/oam_cell_ctrl_cmd.*/oam_cell_ctrl_cmd: 1/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.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:

```
cell_group: 1 cell_group_num: 8 fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```
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:

```
sudo ./cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 8C 14 sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08 sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 8C 14
```

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:

```
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 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:

- CONFIG.req command for 1 cell at a time. cmd=3 is for CONFIG.req
- Start cell-0 (cmd=1)
- Repeat the above sequence for all cells

```
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
```

```
# testMac console 20:34:22.124683 C [MAC.SCF] cell_init: cell_id=0 fapi_type=SCF global_tick=-1 first_init=1 20:34:26.124793 C [MAC.SCF] cell_init: cell_id=1 fapi_type=SCF global_tick=-1 first_init=1 20:34:28.124858 C [MAC.SCF] cell_start:
```

```
cell_id=0 fapi_type=SCF global_tick=-1 04:55:13.040024 Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0 04:55:13.040037 Cell 1 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040045 Cell 2 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040051 Cell 3 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040058 Cell 4 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040065 Cell 5 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040069 Cell 6 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 04:55:13.040074 Cell 7 | DL 829.36 Mbps 1600 Slots| UL 122.92 Mbps 400 Slots | Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0 # cuphycontroller console 04:55:13.040004 C [SCF.PHY] Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040018 C [SCF.PHY] Cell 1 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040023 C [SCF.PHY] Cell 2 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040027 C [SCF.PHY] Cell 3 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040033 C [SCF.PHY] Cell 4 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040037 C [SCF.PHY] Cell 5 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040044 C [SCF.PHY] Cell 6 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000 04:55:13.040051 C [SCF.PHY] Cell 7 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots CRC 0 ( 0) | Tick 142000
```

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:

```
sed -i "s/prometheus_thread.*/prometheus_thread: 23/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_R750.yaml sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu numactl -N 1 -m 1 -- ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08_R750
```

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 LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu ${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/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml sudo -E LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mellanox/dpdk/lib/x86_64-linux-gnu:/opt/mellanox/doca/lib/x86_64-linux-gnu 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="6",type="UL"} 4000
> aerial_cuphycp_slots_total{cell="3",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="7",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="2",type="UL"} 4000
> aerial_cuphycp_slots_total{cell="3",type="UL"} 4000
> aerial_cuphycp_slots_total{cell="1",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="1",type="UL"} 4000
> aerial_cuphycp_slots_total{cell="2",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="6",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="4",type="UL"} 4000
> aerial_cuphycp_slots_total{cell="5",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="4",type="DL"} 16000
> aerial_cuphycp_slots_total{cell="5",type="UL"} 4000 *... # 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="1"} 1680000
> aerial_cuphycp_on_time_uplane_rx_packets_total{cell="8"} 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 # TYPE aerial_cuphycp_cplane_tx_bytes_total counter*
> aerial_cuphycp_cplane_tx_bytes_total{cell="7"} 101048000
> aerial_cuphycp_cplane_tx_bytes_total{cell="6"} 101048000

```
aerial_cuphycp_cplane_tx_bytes_total{cell="2"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="1"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="8"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="3"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="4"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="5"} 101048000
```

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.

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 sed -i 's/log_name:.*/log_name: testmac1.log/g' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml sed -i 's/oam_server_addr:.*/oam_server_addr: 0.0.0.0:50053/g' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml sed -i '/recv_thread_config/{ N; N; N; s/cpu_affinity:[^\n]*/cpu_affinity: 15/g}' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml sed -i '/builder_thread_config/{ N; N; N; s/cpu_affinity:[^\n]*/cpu_affinity: 16/g}' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml
```

2. Switch nvipc config to nvipc_multi_instances.yaml in L1.

```
# 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.

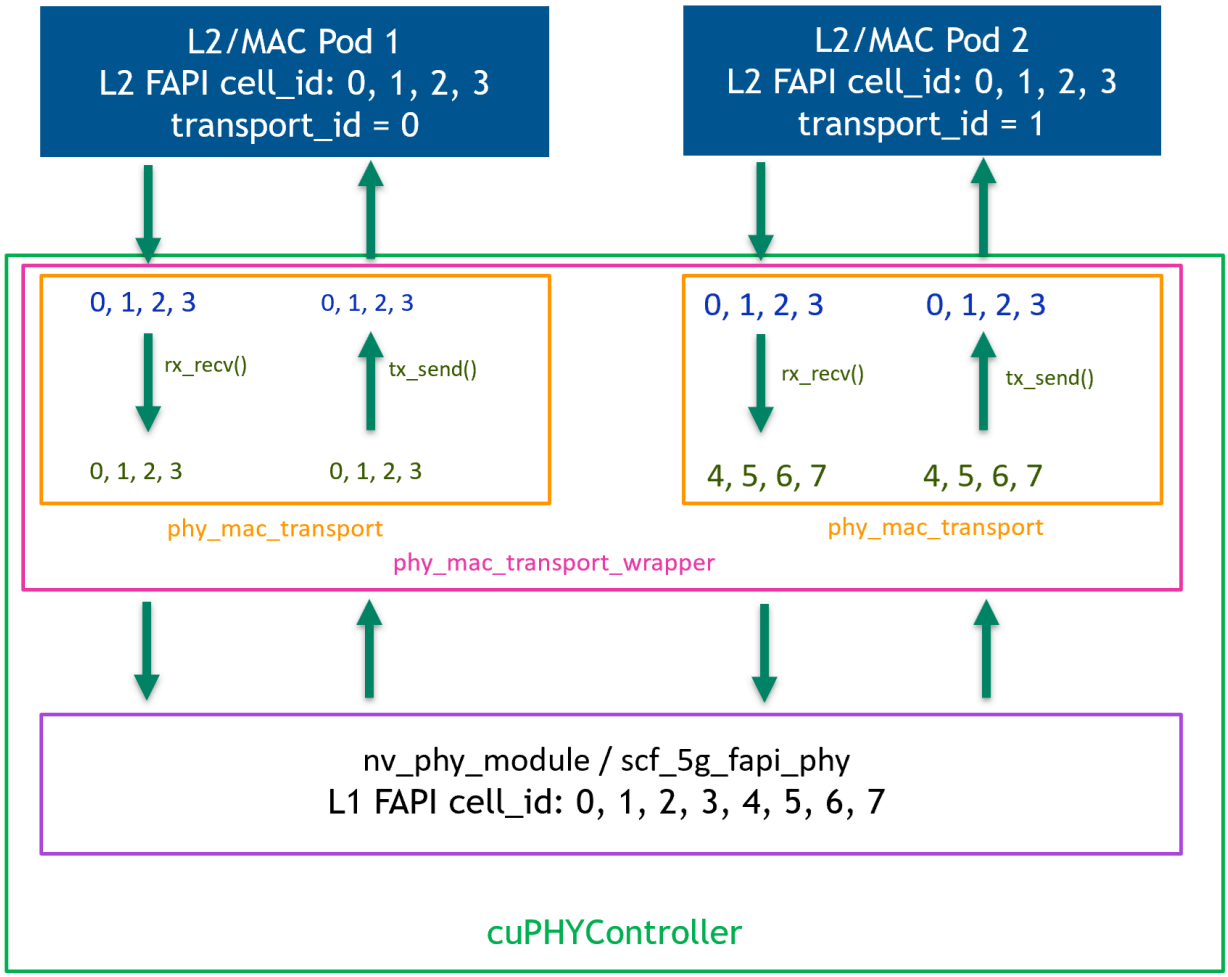
   Assume 8 cells are configured in cuphycontroller_XXX.yaml, the indexes for them are 0 ~ 7.

```
L1 cells: 0 ~ 7 The 1st L2 instance cells: 0 ~ 3 The 2nd L2 instance cells: 4 ~ 7
```

   Then configure:

```
# 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:

```
sudo ./ru_emulator F08 8C_60 sudo ./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.

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.

> *# 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.

```
# Usage: aerial_cell_param_net_update.py cell_id dst_mac_addr vlan_tci 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:B3 E002
```