xRAN Front Haul

Software Architecture Specification v21.03

March 2021

Revision 8.0

Intel Confidential

Document Number: 611268-8.0

Contents

Figures

Tables

Revision History

Revision Number	Description	Revision Date
8.0	Updated for the FlexRAN Software Release v21.03 with the following changes: • Revised Section 1.3 • Revised Table 8 in Section 4.2 • Revised Section 6 • Added Appendix C with eCPRI DDP for Columbiaville	March 2021
7.0	Updated for the FlexRAN Software Release v20.11 with the following changes: • Revised Section 1.4 • Revised Section 3.7 • Revised Table 8 in Section 4.2 • Revised Section 5.5.3, xRAN Ethernet • Revised Appendix A.2, A.3	November 2020
6.0	Updated for the FlexRAN Software Release v20.08 with the following changes: • Revised Section 2.1, software version number • Revised Section 2.4, Category B Support • Revised Section 4.1 Note • Revised Table 7 and Table 8 • Revised Section 4.5.5 Note • Revised Section 5.0 • Revised Sections 5.2.4, C-plane c • Revised Section 5.3.1, External Interface memory • Revised Section 5.4.2.1, Step 1, Step 2, and Step 3 • Revised Table 11 • Revised Section 5.4.3, U-plane • Revised Section 6.0, 1 Cell Sub6 100MHz TDD (Category B) • Replaced Figure 28 • Revised Appendix A.1, A.2, A3, A4	August 2020
5.0	• Updated for the FlexRAN software release v20.04 • Updated Table 1, Terminology Table and Table 2, Reference Documents • Revised Section 2.1, Assumptions • Revised Section 2.3, Dependencies DPDK v19.11 • Revised Section 2.3, Constraints • Revised Section 3.1, Introduction • Revised Section 4.1, Introduction • Revised Table 7 and Table 8 • Revised Section 4.3	April 2020

Revision Number	Description	Revision Date
	• Revised Section 4.4.2 and Section 4.4.3 • Revised Section 4.5, C-Plane • Revised Section 5.0, xRAN Library Design code • Revised Section 5.2.1, Configuration • Revised Section 5.2.4, C-Plane Information Settings code and text • Revised Section 5.4.1, External API • Revised Section 5.4.2.1.1, API and Data Structures • Revised Section 5.4.3, U-Plane • Revised Section 6.0, Sample Application • Revised Appendix A.2. Prerequisites • Revised Appendix A3, Configuration of System • Revised Appendix A4, Install and Configure Sample Application • Revised Appendix A5, Install and Configure FlexRAN 5G NR L1 Application	
4.0	Updated for the FlexRAN software release v20.02	February 2020
3.0	Updated for the FlexRAN software release v19.10	October 2019
2.0	Updated for the FlexRAN software release v19.06	July 2019
1.0	Initial release	April 2019

§

1.0 Introduction

This document describes the design of the Open Radio Access Network (O-RAN) Fronthaul interface system architecture supporting Fifth Generation New Radio (5G NR) and Long-Term Evolution (LTE). In this document, Extensible Radio Access Network (xRAN) refers to the software library implementing the O-RAN Fronthaul interface according to O-RAN Fronthaul Working Group Control, User and Synchronization Plane Specification (refer to Table 2).

1.1 Scope

This document provides a description of xRAN library design and implementation.

1.2 Intended Audience

The intended audience for this document is software engineers and system architects who design and develop 5G systems using the O-RAN Fronthaul interface, according to O-RAN Fronthaul Working Group Control, User, and Synchronization Plane Specification (refer to Table 2).

1.3 Terminology

Table 1. Terminology

Term	Description
5G NR	Fifth Generation New Radio
ACS	Access Control system
API	Application Programming Interface
BOM	Bill of Materials
CP	Cyclic Prefix
DDP	Dynamic Device Personalization
DPDK	Data Plane Development Kit
eAxC	Extended Antenna Carrier
eCPRI	Enhanced Common Public Radio Interface
eNB	Enode B
ETH	Ethernet
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FH	Front Haul
gNB	Next-generation NodeB also named as Base Station
GNSS	Global Navigation Satellite System

Term	Description
GPS	Global Positioning System
HARQ	Hybrid Automatic Repeat Request
HW	Hardware
IFG	Interframe Gap
IFFT	Inverse Fast Fourier Transform
IoT	Inter-Operability Testing
IQ	In-band and Quadrature
LAA	License Assisted Access
LTE	Long Term Evolution
MAC	Media Access Control
MEC	Mobile Edge Computing
M-Plane	Management Plane
mmWave	Millimeter Wave
NIC	Network Interface Controller
O-DU	O-RAN Distributed Unit: a logical node hosting RLC/MAC/High-PHY layers based on a lower layer functional split.
O-RU	O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split. This is similar to 3GPP's "TRP" or "RRH" but more specific in including the Low-PHY layer (FFT/IFFT, PRACH extraction).
OWD	One Way Delay
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PHC	Physical Hardware Clock
PHP	Hypetext Preprocessor
PMD	Poll Mode Driver
POSIX	Portable Operating System Interface
PRACH	Physical Random Access Channel
PRB	Physical Resource Block
PRTC	Protected Real Time Clock
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
PTP	Precision Time Protocol
RA	Random Access
RAN	Radio Access Network
RB	Resource Block
RE	Resource Element
RLC	Radio Link Control

Term	Description
RoE	Radio over Ethernet
RT	Real Time
RTE	Real Time Environment
RSS	Receive Side Scaling
RU	Radio Unit
SR-IOV	Single Root Input/Output Virtualization
SW	Software
SyncE	Synchronous Ethernet
TDD	Time Division Duplex
ToS	Top of the Second
TSC	Time Stamp Counter
TTI	Transmission Time Interval
UE	User Equipment
UL	Uplink
VF	Virtual Function
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
WLS	Wireless Subsystem Interface
xRAN	Extensible Radio Access Network

1.4 Reference Documents

Table 2. Reference Documents

Document	Document No./Location
FlexRAN Reference Solution Software Release Notes	575822
FlexRAN Reference Solution L1 XML Configuration User Guide	571741
FlexRAN Reference Solution LTE eNB L2-L1 Application Programming Interface [API] Specification	571742
FlexRAN Reference Solution L2-L1 nFAPI Specification	576423
FlexRAN and Mobile Edge Compute (MEC) Platform Setup Guide	575891
FlexRAN 5G NR Reference Solution RefPHY (Doxygen).	603577

Document	Document No./Location
Intel® Ethernet Controller E810 Dynamic Device Personalization (DDP) Technology Guide	617015
3GPP specification series*	https://www.3gpp.org/dynareport/SpecList.htm?release=Rel-15&tech=3&ts=1&tr=1
Wolf Pass Server Documentation	https://ark.intel.com/products/codename/80739/Wolf-Pass
Intel® C++ Compiler in Intel® Parallel Studio XE	https://software.intel.com/en-us/c-compilers/ipsxe
DPDK documentation	http://dpdk.org/doc/guides/
O-RAN Fronthaul Working Group Control, User and Synchronization Plane Specification (ORAN-WG4.CUS.0-v04.00)	https://www.o-ran.org/specifications
ORAN Specification	https://www.o-ran.org/adopter-license
IEEE-1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems	https://standards.ieee.org/standard/1588-2008.html
eCPRI Specification V2.0 Interface Specification	http://www.cpri.info/spec.html

§

2.0 Assumptions, Dependencies, and Constraints

This chapter contains limitations on the scope of the document.

2.1 Assumptions

Implementation and details of 5G NR gNodeB and LTE eNodeB software design are outside of the scope of this document.

It is expected that the user is familiar with the *O-RAN Fronthaul Working Group Control, User, and Synchronization Plane Specification* (refer to Table 2). Only implementation-specific details are provided in this document.

Unless otherwise specified, the information in this document is provided for the v21.03 version of the xRAN library.

2.2 Requirements

Usage of this library requires PTP for Linux* v2.0 (or later) to be installed to provide IEEE 1588 synchronization.

2.3 Dependencies

xRAN library implementation depends on the Data Plane Development Kit (DPDK v20.11).

DPDK v20.11 should be patched with corresponding DPDK patch provided with FlexRAN release (see Table 1, FlexRAN Reference Solution Software Release Notes)

Intel® C++ Compiler v19.0.3 is used.

NOTE: Optionally Octave v3.8.2 can be used to generate reference IQ samples (`octave-3.8.2-20.el7.x86_64`).

2.4 Constraints

This release has been designed and implemented to support the following numerologies defined in the 3GPP specifications for LTE and 5GNR (refer to Table 2):

5G NR

Category A support:

- Numerology 0 with bandwidth 5/10/20 MHz with up to 12 cells
- Numerology 1 with bandwidth 100 MHz with up to 1 cell
- Numerology 3 with bandwidth 100 MHz with up to 1 cell

Category B support:

Numerology 1 with bandwidth 100 MHz where the antenna panel is up to 64T64R with up to 3 cells.

LTE

Category A support:

Bandwidth 5/10/20 MHz with up to 12 cells

Category B support:

Bandwidth 5/10/20 MHz for 1 cell

The feature set of O-RAN protocol should be aligned with Radio Unit (O-RU) implementation. Inter-operability testing (IoT) is required to confirm the correctness of functionality on both sides. The exact feature set supported is described in Chapter 4.0 Transport Layer and O-RAN Fronthaul Protocol Implementation of this document.

§

3.0 Architecture Overview

This section provides an overview of the O-RAN architecture.

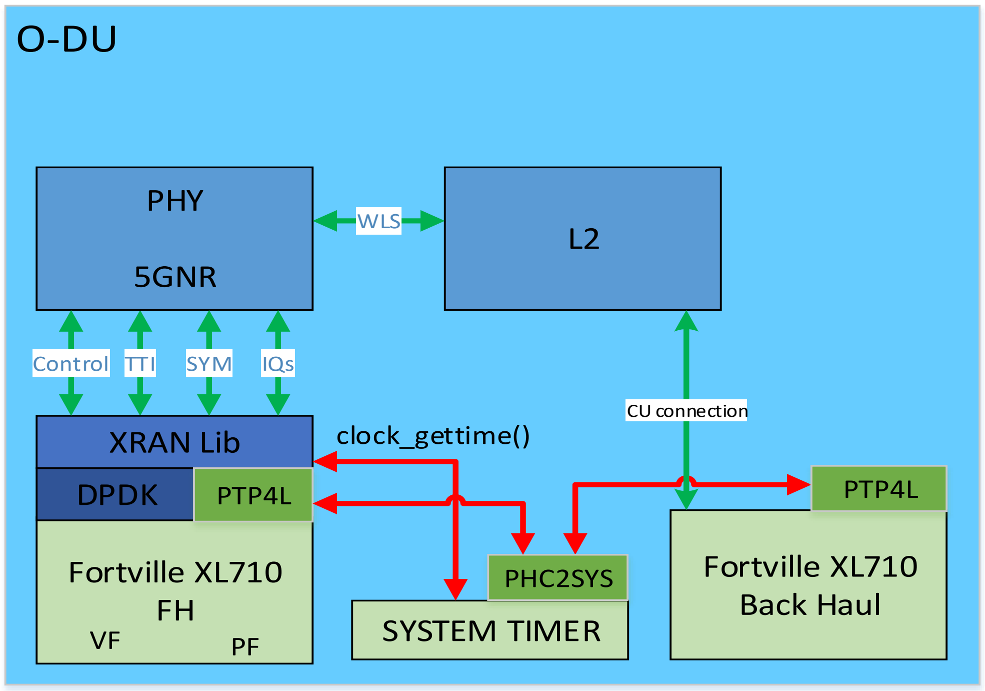
3.1 Introduction

The front haul interface, according to the O-RAN Fronthaul specification, is part of the 5G NR L1 reference implementation provided with the FlexRAN software package. It performs communication between O-RAN Distributed Unit (O-DU) and O-RAN Radio Unit (O-RU) and consists of multiple HW and SW components.

The logical representation of HW and SW components is shown in Figure 1.

The same architecture design is applicable for LTE; however, the FH library is not integrated with the PHY pipeline for FlexRAN LTE.

Figure 1. Architecture Block Diagram



From the hardware perspective, two networking ports are used to communicate to the Front Haul and Back (Mid) Haul network as well as to receive PTP synchronization. The system timer is used to provide a “sense” of time to the gNB application.

From the software perspective, the following components are used:

- Linux* PTP provides synchronization of system timer to GPS time:
  - `ptp4l` is used to synchronize oscillator on Network Interface Controller (NIC) to PTP GM.
  - `phc2sys` is used to synchronize system timer to oscillator on NIC.
- The DPDK to provide the interface to the Ethernet port.

- xRAN library is built on top of DPDK to perform U-plane and C-plane functionality according to the O-RAN Fronthaul specification.
- 5GNR reference PHY uses the xRAN library to access interface to O-RU. The interface between the library and PHY is defined to communicate TTI event, symbol time, C-plane information as well as IQ sample data.
- 5G NR PHY communicates with the L2 application using the set of MAC/PHY APIs and the shared memory interface defined as WLS.
- L2, in turn, can use Back (Mid) Haul networking port to connect to the CU unit in the context of 3GPP specification.

In this document, we focus on details of the design and implementation of the xRAN library for providing Front Haul functionality for both mmWave and Sub-6 scenarios as well as LTE.

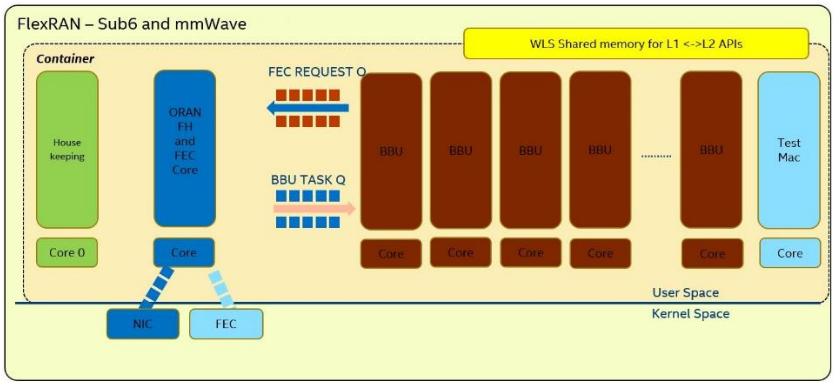
The xRAN M-plane is not implemented and is outside of the scope of this description. Configuration files are used to specify selected M-plane level parameters.

3.2 5G NR L1 Application Threads

The specifics of the L1 application design and configuration for the given scenario can be found in document 603577, *FlexRAN 5G NR Reference Solution RefPHY* (Doxygen) (refer to Table 2) Only information relevant to front haul is presented in this section.

Configuration of l1app with xRAN interface for Front Haul is illustrated acting as an O-DU in Figure 2.

Figure 2. 5G NR L1app Threads



In this configuration of L1app, the base architecture of 5G NR L1 is not changed. The original Front Haul FPGA interface was updated with the O-RAN fronthaul interface abstracted via the xRAN library.

O-RAN FH Thread Performs:

- Symbol base “time event” to the rest of the system based on System Clock synchronized to GPS time via PTP
- Baseline polling mode driver performing TX and RX of Ethernet packets
- Most of the packet processing such as Transport header, Application header, Data section header, and interactions with the rest of the PHY processing pipeline.
- Polling of BBDev for FEC on PAC N3000 acceleration card

  The other threads are standard for the L1app and created the independent usage of O-RAN as an interface to the Radio.

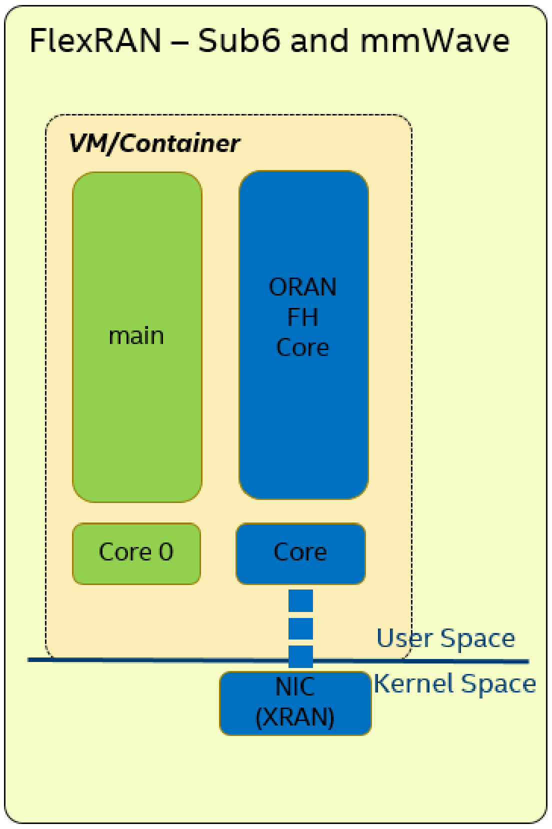
Communication between L1 and O-RAN layer is performed using a set of callback functions where L1 assigned callback and O-RAN layer executes those functions at a particular event or time moment. Detailed information on callback function options and setting, as well as design, can be found in the sections below.

Design and installation of the l1app do not depend on the Host, VM, or container environment and the same for all cases.

3.3 Sample Application Thread Model

Configuration of a sample application for both the O-DU and O-RU follows the model of 5G NR l1app application in Figure 2, but no BBU or FEC related threads are needed as minimal O-RAN FH functionality is used only.

Figure 3. Sample Application Threads

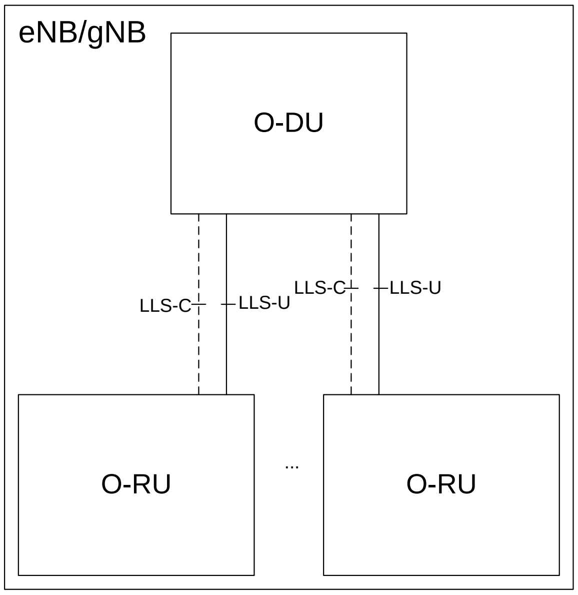


In this scenario, the main thread is used only for initializing and closing the application. No execution happens on core 0 during run time.

3.4 Functional Split

Figure 1 corresponds to the O-RU part of the O-RAN split. Implementation of the RU side of the O-RAN protocol is not covered in this document.

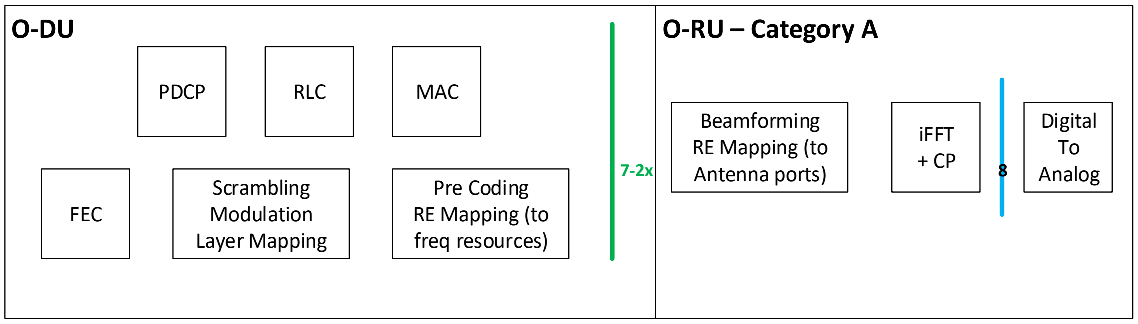
Figure 4. eNB/gNB Architecture with O-DU and RU



More than one RU can be supported with the same implementation of the xRAN library and depends on the configuration of gNB in general. In this document, we address details of implementation for a single O-DU – O-RU connection.

The O-RAN Fronthaul specification provides two categories of the split of Layer 1 functionality between O-DU and O-RU: Category A and Category B.

Figure 5. Functional Split


O-DU
PDCP
RLC
MAC
FEC
Scrambling Modulation Layer Mapping
Pre Coding RE Mapping (to freq resources)
7-2x
O-RU – Category A
Beamforming RE Mapping (to Antenna ports)
iFFT + CP
8
Digital To Analog

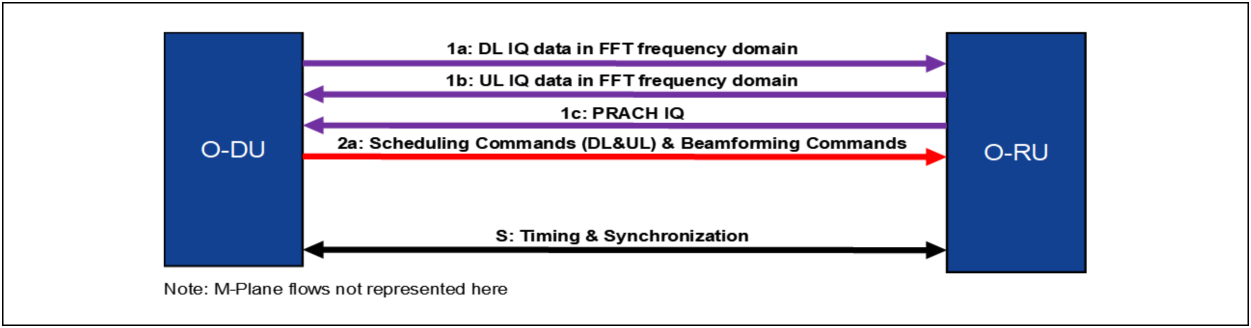
3.5 Data Flow

Table 3 lists the data flows supported for a single RU with a single Component Carrier.

Table 3. Supported Data Flow

Plane	ID	Name	Contents	Periodicity
U-Plane	1a	DL Frequency Domain IQ Data	DL user data (PDSCH), control channel data (PDCCH, etc.)	symbol
	1b	UL Frequency Domain IQ Data	UL user data (PUSCH), control channel data (PUCCH, etc.)	symbol
	1c	PRACH Frequency Domain IQ Data	UL PRACH data	slot or symbol
C-Plane	2a	Scheduling Commands (Beamforming is not supported)	Scheduling information, FFT size, CP length, Subcarrier spacing, UL PRACH scheduling	~ slot
S-Plane	S	Timing and Synchronization	IEEE 1588 PTP packets	-

Figure 6. Data Flows


O-DU
O-RU
1a: DL IQ data in FFT frequency domain
1b: UL IQ data in FFT frequency domain
1c: PRACH IQ
2a: Scheduling Commands (DL&UL) & Beamforming Commands
S: Timing & Synchronization
Note: M-Plane flows not represented here

Information on specific features of C-Plane and U-plane provided in Section 6.0,. Sample Application Configuration of S-plane used on test setup for simulation is provided in Appendix B, PTP Configuration.

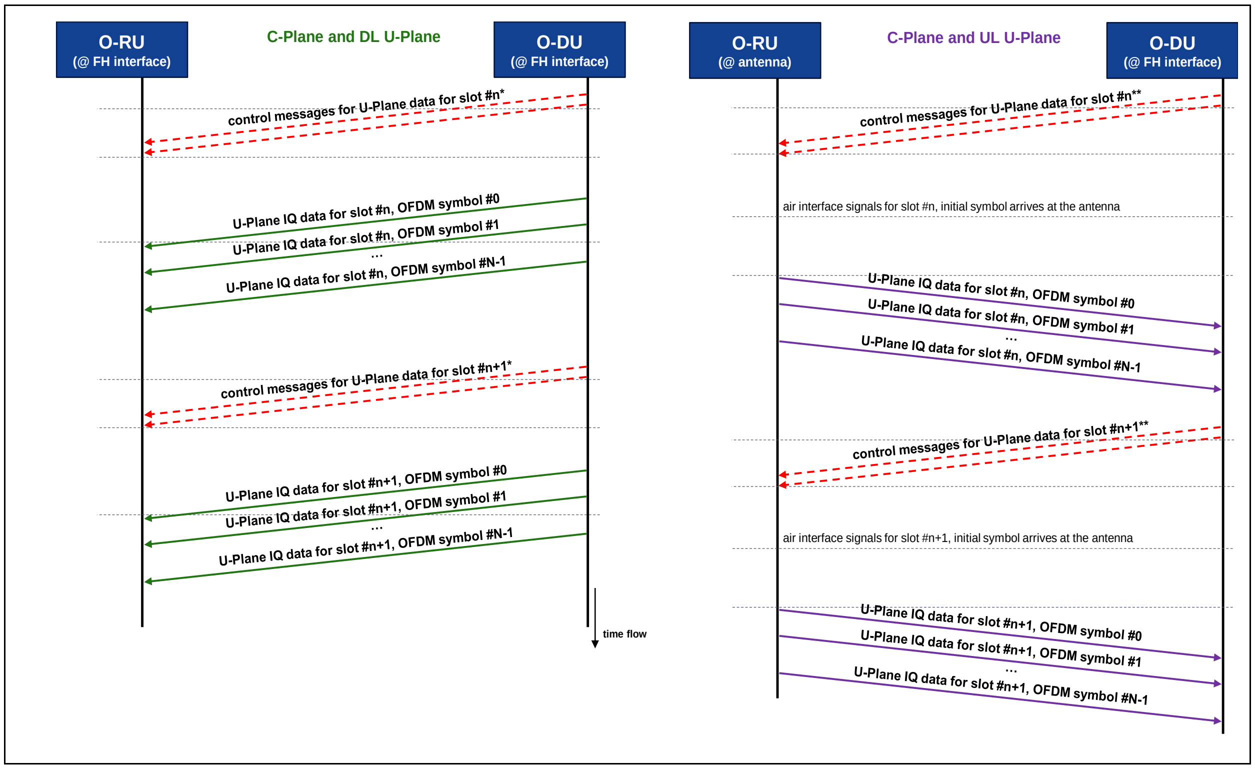
Data flow separation is based on VLAN (applicable when layer 2 or layer 3 is used for the C/U-plane transport.)

NOTE: The mechanism for assigning VLAN ID to U-Plane and C-Plane is assumed to be via the M-Plane.

NOTE: VLAN Tag is configurable via the standard Linux IP tool, refer to Appendix A, Setup Configuration.

NOTE: No Quality of Service (QoS) is implemented as part of xRAN library. Standard functionality of ETH port can be used to implement QoS.

Figure 7. C-plane and U-plane Packet Exchange


O-RU (@ FH interface)
C-Plane and DL U-Plane
O-DU (@ FH interface)
control messages for U-Plane data for slot #n*
U-Plane IQ data for slot #n, OFDM symbol #0
U-Plane IQ data for slot #n, OFDM symbol #1
...
U-Plane IQ data for slot #n, OFDM symbol #N-1
control messages for U-Plane data for slot #n+1*
U-Plane IQ data for slot #n+1, OFDM symbol #0
U-Plane IQ data for slot #n+1, OFDM symbol #1
...
U-Plane IQ data for slot #n+1, OFDM symbol #N-1
time flow
O-RU (@ antenna)
C-Plane and UL U-Plane
O-DU (@ FH interface)
control messages for U-Plane data for slot #n**
air interface signals for slot #n, initial symbol arrives at the antenna
U-Plane IQ data for slot #n, OFDM symbol #0
U-Plane IQ data for slot #n, OFDM symbol #1
...
U-Plane IQ data for slot #n, OFDM symbol #N-1
control messages for U-Plane data for slot #n+1**
air interface signals for slot #n+1, initial symbol arrives at the antenna
U-Plane IQ data for slot #n+1, OFDM symbol #0
U-Plane IQ data for slot #n+1, OFDM symbol #1
...
U-Plane IQ data for slot #n+1, OFDM symbol #N-1

3.6 Timing, Latency, and Synchronization to GPS

The O-RAN Fronthaul specification defines the latency model of the front haul interface and interaction between O-DU and 0-RU. This implementation of the xRAN library supports only the category with fixed timing advance and Defined Transport methods. It determines O-DU transmit and receive windows based on pre-defined transport network characteristics, and the delay characteristics of the RUs within the timing domain.

Table 4 below provides default values used for the implementation of O-DU – O-RU simulation with `mmWave` scenario. Table 5 and Table 6 below provide default values used for the implementation of O-DU – O-RU simulation with numerology 0 and numerology 1 for Sub6 scenarios. Configuration can be adjusted via configuration files for sample application and reference PHY.

NOTE: However, simulation of the different range of the settings was not performed, and additional implementation changes might be required as well as testing with actual O-RU. The parameters for the front haul network are out of scope as a direct connection between O-DU and 0-RU is used for simulation.

Table 4. Front Haul Interface Latency (numerology 3 - mmWave)

	Model Parameters	C-Plane		U-Plane	
		DL	UL	DL	UL
O-RU	T2amin	T2a_min_cp_dl=50	T2a_min_cp_ul=50	T2a_min_up=25	NA
	T2amax	T2a_max_cp_dl=140	T2a_max_cp_ul=140	T2a_max_up=140	NA
		Tadv_cp_dl	NA	NA	NA
	Ta3min	NA	NA	NA	Ta3_min=20

	Model Parameters	C-Plane		U-Plane	
		DL	UL	DL	UL
	Ta3max	NA	NA	NA	Ta3_max=32
O-DU	T1amin	T1a_min_cp_dl=70	T1a_min_cp_ul=60	T1a_min_up=35	NA
	T1amax	T1a_max_cp_dl=100	T1a_max_cp_ul=70	T1a_max_up=50	NA
	Ta4min	NA	NA	NA	Ta4_min=0
	Ta4max	NA	NA	NA	Ta4_max=45

Table 5. Front Haul Interface Latency (numerology 0 - Sub6)

	Model Parameters	C-Plane		U-Plane	
		DL	UL	DL	UL
O-RU	T2amin	T2a_min_cp_dl=400	T2a_min_cp_ul=400	T2a_min_up=200	NA
	T2amax	T2a_max_cp_dl=112 0	T2a_max_cp_ul=1120	T2a_max_up=1120	NA
		Tadv_cp_dl	NA	NA	NA
	Ta3min	NA	NA	NA	Ta3_min=160
	Ta3max	NA	NA	NA	Ta3_max=256
O-DU	T1amin	T1a_min_cp_dl=560	T1a_min_cp_ul=480	T1a_min_up=280	NA
	T1amax	T1a_max_cp_dl=800	T1a_max_cp_ul=560	T1a_max_up=400	NA
	Ta4min	NA	NA	NA	Ta4_min=0
	Ta4max	NA	NA	NA	Ta4_max=360

Table 6. Front Haul Interface Latency (numerology 1 - Sub6)

	Model Parameters	C-Plane		U-Plane	
		DL	UL	DL	UL
O-RU	T2amin	T2a_min_cp_dl=285	T2a_min_cp_ul=285	T2a_min_up=71	NA
	T2amax	T2a_max_cp_dl=429	T2a_max_cp_ul=429	T2a_max_up=428	NA
		Tadv_cp_dl	NA	NA	NA
	Ta3min	NA	NA	NA	Ta3_min=20
	Ta3max	NA	NA	NA	Ta3_max=32
O-DU	T1amin	T1a_min_cp_dl=285	T1a_min_cp_ul=285	T1a_min_up=96	NA
	T1amax	T1a_max_cp_dl=429	T1a_max_cp_ul=300	T1a_max_up=196	NA
	Ta4min	NA	NA	NA	Ta4_min=0
	Ta4max	NA	NA	NA	Ta4_max=75

IEEE 1588 protocol and PTP for Linux* implementations are used to synchronize local time to GPS time. Details of the configuration used are provided in Appendix B, PTP Configuration. Local time is used to get Top of the Second

(ToS) as a 1 PPS event for SW implementation. Timing event is obtained by performing polling of local time using `clock_gettime(CLOCK_REALTIME,..)`

All-time intervals are specified concerning the GPS time, which corresponds to OTA time.

3.7 Virtualization and Container-Based Usage

xRAN implementation is deployment agnostic and does not require special changes to be used in virtualized or container-based deployment options. The only requirement is to provide one SRIOV base virtual port for C-plane and one port for U-plane traffic per O-DU instance. This can be achieved with the default Virtual Infrastructure Manager (VIM) as well as using standard container networking.

To configure the networking ports, refer to the FlexRAN and Mobile Edge Compute (MEC) Platform Setup Guide (Table 2) and readme.md in xRAN library or Appendix A.

§

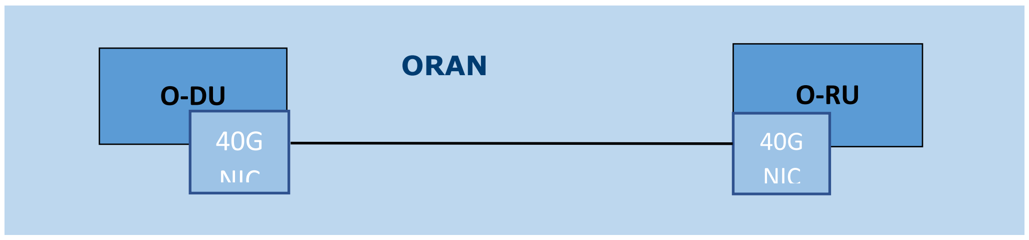
4.0 Transport Layer and O-RAN Fronthaul Protocol Implementation

This chapter describes how the transport layer and O-RAN Fronthaul protocol are implemented.

4.1 Introduction

Figure 8 presents an overview of the O-RAN Fronthaul process.

Figure 8. O-RAN Fronthaul Process



The xRAN library provides support for transporting In-band and Quadrature (IQ) samples between the O-DU and O-RU within the O-RAN architecture based on functional split 7.2x. The library defines the O-RAN packet formats to be used to transport radio samples within Front Haul according to the O-RAN Fronthaul specification, refer to Table 2. It provides functionality for generating O-RAN packets, appending IQ samples in the packet payload, and extracting IQ samples from O-RAN packets.

NOTE: Version 21.03 of the library supports U-plane and C-plane only. M-plane is not supported. It is ready to be used in the PTP synchronized environment.

NOTE: Regarding the clock model and synchronization topology, configurations C1 and C3 of the connection between O-DU and O-RU are the only configurations supported in this release of the xRAN implementation.

NOTE: Quality of PTP synchronization for the S-plane of O-RAN Fronthaul requirements as defined for O-RU is out of the scope of this document. PTP master and PTP slave configuration are expected to satisfy only the O-DU side of requirements and provide the “best-effort” PTP master for O-RU. This may or may not be sufficient for achieving the end to end system requirements of S-plane. Specialized dedicated NIC card with additional HW functionality might be required to achieve PTP master functionality to satisfy O-RU precision requirements for RAN deployments scenarios.

Figure 9. Configuration C1

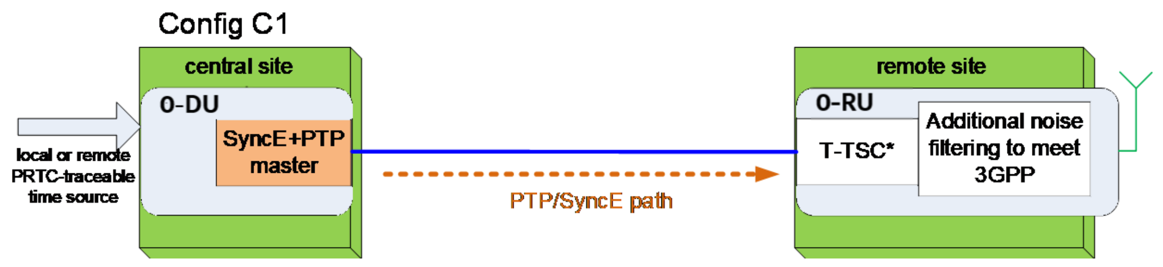
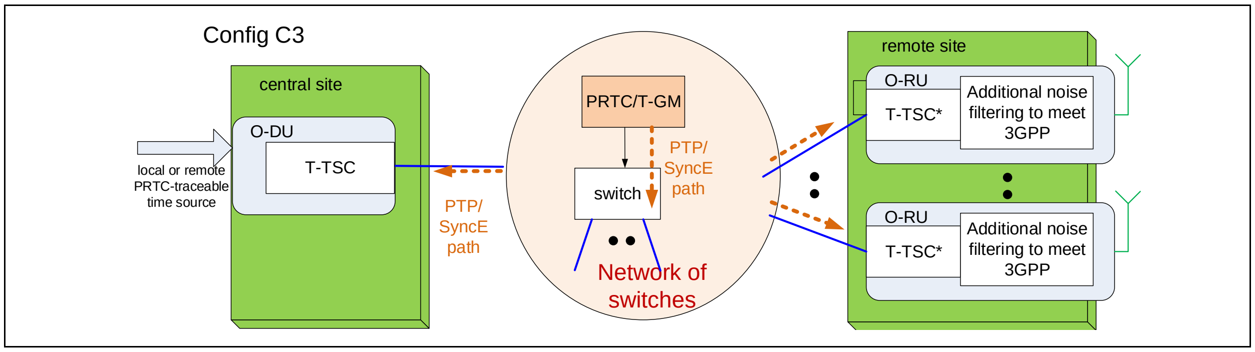

Config C1
central site
O-DU
SyncE+PTP master
local or remote PRTC-traceable time source
PTP/SyncE path
remote site
O-RU
T-TSC*
Additional noise filtering to meet 3GPP

Figure 10. Configuration C3



4.2 Supported Feature Set

The O-RAN Fronthaul specification defines a list of mandatory functionalities.

NOTE: Not all features defined as Mandatory for O-DU are currently supported to full extended.

The following tables contain information on what is available, and the level of validation performed for this release.

NOTE: Cells with a red background are listed as mandatory in the specification but not supported in this implementation of xRAN.

Table 7. O-RAN Mandatory and Optional Feature Support

Category	Feature	O-DU Support	Support
RU Category	Support for CAT-A RU (up to 8 spatial streams)	Mandatory	Y
	Support for CAT-A RU (> 8 spatial streams)		Y
	Support for CAT-B RU (precoding in RU)	Mandatory	Y
Beamforming	Beam Index based	Mandatory	Y
	Real-time BF Weights	Mandatory	Y
	Real-Time Beamforming Attributes		N
	UE Channel Info		N
Bandwidth Saving	Programmable static-bit-width Fixed Point IQ	Mandatory	Y
	Real-time variable-bit-width		Y
	Compressed IQ		Y
	Block floating point compression		Y
	Block scaling compression		N
	u-law compression		N
	modulation compression		Y
	beamspace compression		Y
	Variable Bit Width per Channel (per data section)		Y
	Static configuration of U-Plane IQ format and compression header		N

Category	Feature	O-DU Support	Support
	Use of "symInc" flag to allow multiple symbols in a C-Plane section		N
Energy Saving	Transmission blanking		N
O-DU - RU Timing	Pre-configured Transport Delay Method	Mandatory	Y
	Measured Transport Method (eCPRI Msg 5)		N
Synchronization	G.8275.1	Mandatory	Y* (*C3 only)
	G.8275.2		N
	GNSS based sync		N
	SyncE		N
Transport Features	L2: Ethernet	Mandatory	Y
	L3: IPv4, IPv6 (CUS Plane)		N
	QoS over Fronthaul	Mandatory	Y
	Prioritization of different U-plane traffic types		N
	Support of Jumbo Ethernet frames		N
	eCPRI	Mandatory	Y
	support of eCPRI concatenation		N
	IEEE 1914.3		N
	Application fragmentation	Mandatory	Y
	Transport fragmentation		N
Other	LAA LBT O-DU Congestion Window mgmt		N
	LAA LBT RU Congestion Window mgmt		N

Details on the subset of O-RAN functionality implemented are shown in Table 8.

Level of Validation Specified as:

- **C**: Completed code implementation for xRAN Library
- **I**: Integrated into Reference PHY
- **T**: Tested end to end with O-RU

Table 8. Levels of Validation

Category	Item	Status	C	I	T
General	Radio access technology (LTE/NR)	NR/LTE	N/A	N/A	N/A
	Nominal sub-carrier spacing	15/30/120KHz	Y	Y	N
	FFT size	512/1024/2048/4096	Y	Y	N
	Channel bandwidth	5/10/20/100Mhz	Y	Y	N

Category	Item	Status	C	I	T
	Number of the channel (Component Carrier)	12	Y	Y	N
	RU category	A, B	Y	Y	N
	TDD Config	Supported/Flexible	Y	Y	N
	FDD Support	Supported	Y	Y	N
	Tx/Rx switching based on 'dataDirection' field of C-plane message	Supported	Y	Y	N
	IP version for Management traffic at fronthaul network	N/A	N/A	N/A	N/A
PRACH	One Type 3 message for all repeated PRACH preambles	Supported	Y	Y	N
	Type 3 message per repeated PRACH preambles	1	Y	Y	N
	`timeOffset` including `cpLength`	Supported	Y	Y	N
	Supported	Supported	Y	Y	N
	PRACH preamble format/index number (number of the occasion)	Supported	Y	Y	N
Delay management	Network delay determination	Supported	Y	Y	N
	`lls-CU` timing advance type	Supported	Y	Y	N
	Non-delay managed U-plane traffic	Not supported	N	N	N
C/U-plane Transport	Transport encapsulation (Ethernet/IP)	Ethernet	Y	Y	N
	Jumbo frames	Supported	Y	Y	N
	Transport header (eCPRI/RoE)	eCPRI	Y	Y	N
	IP version when Transport header is IP/UDP	N/A	N/A	N/A	N/A
	eCPRI Concatenation when Transport header is eCPRI	Not supported	N	N	N
	eAxC ID `CU_Port_ID` bitwidth	4 *	Y	Y	N
	eAxC ID `BandSector_ID` bitwidth	4 *	Y	Y	N
	eAxC ID `CC_ID` bitwidth	4 *	Y	Y	N
	eAxC ID `RU_Port_ID` bitwidth	4 *	Y	Y	N
	Fragmentation	Supported	Y	Y	N
	Transport prioritization within U-plane	N/A	N	N	N
	Separation of C/U-plane and M-plane	Supported	Y	Y	N
	Separation of C-plane and U-plane	VLAN ID	Y	Y	N
	Max Number of VLAN per physical port	16	Y	Y	N
	`Rx_on_time`	Supported	Y	Y	N

Category	Item		Status	C	I	T
Reception Window Monitoring (Counters)	`Rx_early`		Supported	N	N	N
	`Rx_late`		Supported	N	N	N
	`Rx_corrupt`		Supported	N	N	N
	`Rx_pkt_dupl`		Supported	N	N	N
	`Total_msgs_rcvd`		Supported	Y	N	N
Beamforming	RU beamforming type		Index and weights	Y	Y	N
	Beamforming control method		C-plane	Y	N	N
	Number of beams		No-restrictions	Y	Y	N
IQ compression	U-plane data compression method		Supported	Y	Y	Y
	U-plane data IQ bitwidth (Before/After compression)		BFP: 8,9,12,14 bits Modulation compression: 1,2,3,4 bits	Y	Y	Y
	Static configuration of U-plane IQ format and compression header		Supported	N	N	N
eCPRI Header Format	`ecpriVersion`		001b	Y	Y	Y
	`ecpriReserved`		Supported	Y	Y	Y
	`ecpriConcatenation`		Not supported	N	N	N
	`ecpriMessage`	U-plane	Supported	Y	Y	Y
		C-plane	Supported	Y	Y	Y
		Delay measurement	Supported	Y	Y	Y
	`ecpriPayload` (payload size in bytes)		Supported	Y	Y	Y
	`ecpriRtcid/ecpriPcid`		Supported	Y	Y	Y
	`ecpriSeqid: Sequence ID`		Supported	Y	Y	Y
	`ecpriSeqid: E bit`		Supported	Y	Y	Y
	`ecpriSeqid: Subsequence ID`		Not supported	N	N	N
C-plane Type	Section Type 0		Not supported	N	N	N
	Section Type 1		Supported	Y	Y	Y
	Section Type 3		Supported	Y	Y	Y
	Section Type 5		Not supported	N	N	N
	Section Type 6		Not supported	N	N	N
	Section Type 7		Not supported	N	N	N
C-plane Packet Format	*Coding of Information Elements – Application Layer, Common*	`dataDirection` (data direction (gNB Tx/Rx))	Supported	Y	Y	N
		`payloadVersion` (payload version)	001b	Y	Y	N

Category	Item		Status	C	I	T
		`filterIndex` (filter index)	Supported	Y	Y	N
		`frameId` (frame identifier)	Supported	Y	Y	N
		`subframeId` (subframe identifier)	Supported	Y	Y	N
		`slotId` (slot identifier)	Supported	Y	Y	N
		`startSymbolid` (start symbol identifier)	Supported	Y	Y	N
		`numberOfsections` (number of sections)	up to the maximum number of PRBs	Y	Y	N
		`sectionType` (section type)	1 and 3	Y	Y	N
		`udCompHdr` (user data compression header)	Supported	Y	Y	N
		`numberOfUEs` (number Of UEs)	Not supported	N	N	N
		`timeOffset` (time offset)	Supported	Y	Y	N
		`frameStructure` (frame structure)	mu=0,1,3	Y	Y	N
		`cpLength` (cyclic prefix length)	Supported	Y	Y	N
	Coding of Information Elements – Application Layer, Sections	`sectionId` (section identifier)	Supported	Y	Y	N
		`rb` (resource block indicator)	0	Y	Y	N
		`symInc` (symbol number increment command)	0 or 1	Y	Y	N
		`startPrbc` (starting PRB of control section)	Supported	Y	Y	N
		`reMask` (resource element mask)	Supported	Y	Y	N
		`numPrbc` (number of contiguous PRBs per control section)	Supported	Y	Y	N
		`numSymbol` (number of symbols)	Supported	Y	Y	N

Category	Item		Status	C	I	T
		`ef` (extension flag)	Supported	Y	Y	N
		`beamId` (beam identifier)	Support	Y	Y	N
		`ueId` (UE identifier)	Not supported	N	N	N
		`freqOffset` (frequency offset)	Supported	Y	Y	N
		`regularizationFactor` (regularization Factor)	Not supported	N	N	N
		`ciIsample, ciQsample` (channel information I and Q values)	Not supported	N	N	N
		`laaMsgType` (LAA message type)	Not supported	N	N	N
		`laaMsgLen` (LAA message length)	Not supported	N	N	N
		`lbtHandle`	Not supported	N	N	N
		`lbtDeferFactor` (listen-before-talk defer factor)	Not supported	N	N	N
		`lbtBackoffCounter` (listen-before-talk backoff counter)	Not supported	N	N	N
		`lbtOffset` (listen-before-talk offset)	Not supported	N	N	N
		`MCOT` (maximum channel occupancy time)	Not supported	N	N	N
		`lbtMode` (LBT Mode)	Not supported	N	N	N
		`lbtPdschRes` (LBT PDSCH Result)	Not supported	N	N	N
		`sfStatus` (subframe status)	Not supported	N	N	N
		`lbtDrsRes` (LBT DRS Result)	Not supported	N	N	N
		`initialPartialSF` (Initial partial SF)	Not supported	N	N	N
		`lbtBufErr` (LBT Buffer Error)	Not supported	N	N	N
		`sfnSf` (SFN/SF End)	Not supported	N	N	N

Category	Item		Status	C	I	T
		lbtCWConfig_H (HARQ Parameters for Congestion Window management)	Not supported	N	N	N
		lbtCWConfig_T (TB Parameters for Congestion Window management)	Not supported	N	N	N
		lbtTrafficClass (Traffic class priority for Congestion Window management)	Not supported	N	N	N
		lbtCWR_Rst (Notification about packet reception successful or not)	Not supported	N	N	N
		reserved (reserved for future use)	0	N	N	N
		Section Extension Commands				
		extType (extension type)	Supported	Y	Y	N
		ef (extension flag)	Supported	Y	Y	N
		extLen (extension length)	Supported	Y	Y	N
	Coding of Information Elements – Application Layer, Section Extensions					
	ExtType=1: Beamforming Weights Extension Type	bfwCompHdr (beamforming weight compression header)	Supported	Y	Y	N
		bfwCompParam (beamforming weight compression parameter)	Supported	Y	Y	N
		bfwI (beamforming weight in-phase value)	Supported	Y	Y	N
		bfwQ (beamforming weight quadrature value)	Supported	Y	Y	N
	ExtType=2: Beamforming Attributes Extension Type	bfaCompHdr (beamforming attributes compression header)	Supported	Y	N	N

Category	Item		Status	C	I	T
		bfAzPt (beamforming azimuth pointing parameter)	Supported	Y	N	N
		bfZePt (beamforming zenith pointing parameter)	Supported	Y	N	N
		bfAz3dd (beamforming azimuth beamwidth parameter)	Supported	Y	N	N
		bfZe3dd (beamforming zenith beamwidth parameter)	Supported	Y	N	N
		bfAzSl (beamforming azimuth sidelobe parameter)	Supported	Y	N	N
		bfZeSl (beamforming zenith sidelobe parameter)	Supported	Y	N	N
		zero-padding	Supported	Y	N	N
	ExtType=3: DL Precoding Extension Type	codebookIndex (precoder codebook used for transmission)	Supported	Y	N	N
		layerID (Layer ID for DL transmission)	Supported	Y	N	N
		txScheme (transmission scheme)	Supported	Y	N	N
		numLayers (number of layers used for DL transmission)	Supported	Y	N	N
		crsReMask (CRS resource element mask)	Supported	Y	N	N
		crsSyumINum (CRS symbol number indication)	Supported	Y	N	N
		crsShift (crsShift used for DL transmission)	Supported	Y	N	N
		beamIdAP1 (beam id to be used for antenna port 1)	Supported	Y	N	N

Category	Item		Status	C	I	T
		`beamIdAP2` (beam id to be used for natenna port 2)	Supported	Y	N	N
		`beamIdAP3` (beam id to be used for natenna port 3)	Supported	Y	N	N
	ExtType=4: Modulation Compression Parameters Extension Type	`csf` (constellation shift flag)	Supported	Y	Y	N
		`modCompScaler` (modulation compression scaler value)	Supported	Y	Y	N
	ExtType=5: Modulation Compression Additional Parameters Extension Type	`mcScaleReMask` (modulation compression power scale RE mask)	Supported	Y	N	N
		`csf` (constellation shift flag)	Supported	Y	N	N
		`mcScaleOffset` (scaling value for modulation compression)	Supported	Y	N	N
	ExtType=6: Non-contiguous PRB allocation in time and frequency domain	`rbgSize` (resource block group size)	Supported	Y	N	N
		`rbgMask` (resource block group bit mask)	Supported	Y	N	N
		`symbolMask` (symbol bit mask)	Supported	Y	N	N
	ExtType=10: Section description for group configuration of multiple ports	`beamGroupType`	Supported	Y	N	N
		`numPortc`	Supported	Y	N	N
	ExtType=11: Flexible Beamforming Weights Extension Type	`bfwCompHdr` (beamforming weight compression header)	Supported	Y	Y	N
		`bfwCompParam` for PRB bundle x (beamforming weight compression parameter)	Supported	Y	Y	N

Category	Item		Status	C	I	T
		numBundPrb (Number of bundled PRBs per beamforming weights)	Supported	Y	Y	N
		bfwI (beamforming weight in-phase value)	Supported	Y	Y	N
		bfwQ (beamforming weight quadrature value)	Supported	Y	Y	N
		disableBFWs (disable beamforming weights)	Supported	Y	Y	N
		RAD (Reset After PRB Discontinuity)	Supported	Y	Y	N
U-plane Packet Format	dataDirection (gNB Tx/Rx)		Supported	Y	Y	Y
	payloadVersion (payload version)		001b	Y	Y	Y
	filterIndex (filter index)		Supported	Y	Y	Y
	frameId (frame identifier)		Supported	Y	Y	Y
	subframeId (subframe identifier)		Supported	Y	Y	Y
	slotId (slot identifier)		Supported	Y	Y	Y
	symbolId (symbol identifier)		Supported	Y	Y	Y
	sectionId (section identifier)		Supported	Y	Y	Y
	rb (resource block indicator)		0	Y	Y	Y
	symInc (symbol number increment command)		0	Y	Y	Y
	startPrbu (startingPRB of user plane section)		Supported	Y	Y	Y
	numPrbu (number of PRBs per user plane section)		Supported	Y	Y	Y
	udCompHdr (user data compression header)		Supported	Y	Y	Y
	reserved (reserved for future use)		0	Y	Y	Y
	udCompParam (user data compression parameter)		Supported	Y	Y	Y
	iSample (in-phase sample)		16	Y	Y	Y
	qSample (quadrature sample)		16	Y	Y	Y

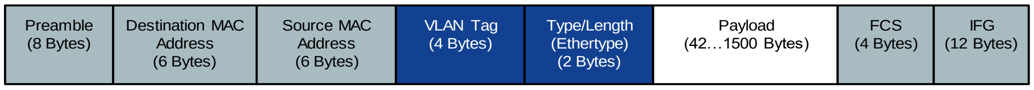
Category	Item		Status	C	I	T
S-plane	Topology configuration: C1		Supported	N	N	N
	Topology configuration: C2		Supported	N	N	N
	Topology configuration: C3		Supported	Y	Y	Y
	Topology configuration: C4		Supported	N	N	N
	PTP	Full Timing Support (G.8275.1)	Supported	Y	Y	N
M-plane			Not supported	N	N	N

NOTE: The bit width of each component in `eAxC` ID can be configurable.

4.3 Transport Layer

O-RAN Fronthaul data can be transported over Ethernet or IPv4/IPv6. In the current implementation, the xRAN library supports only Ethernet with VLAN.

Figure 11. Native Ethernet Frame with VLAN

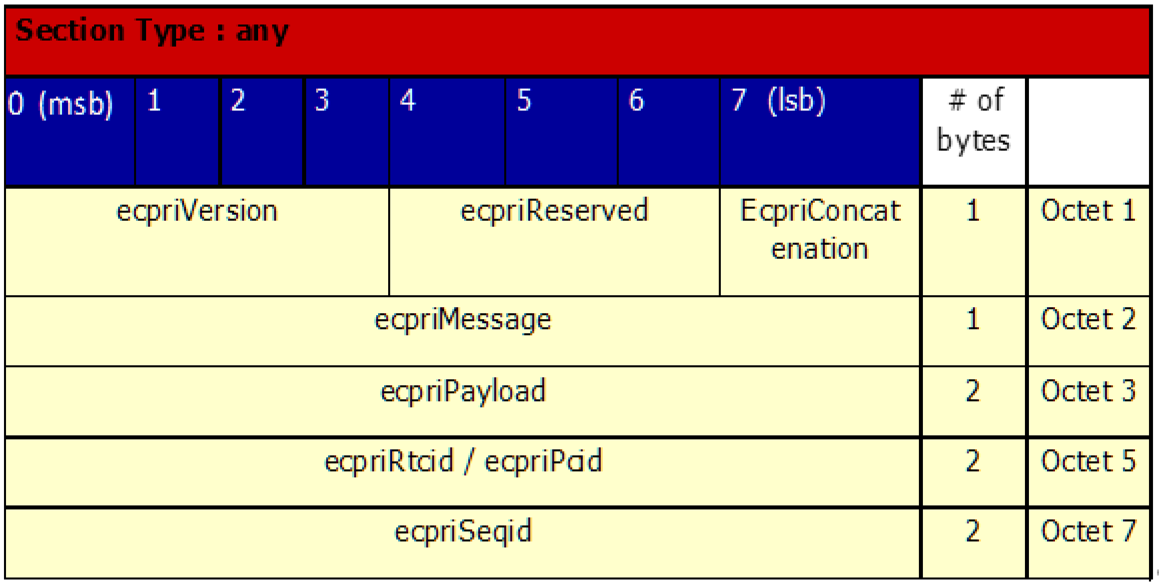

Preamble (8 Bytes)
Destination MAC Address (6 Bytes)
Source MAC Address (6 Bytes)
VLAN Tag (4 Bytes)
Type/Length (Ethertype) (2 Bytes)
Payload (42…1500 Bytes)
FCS (4 Bytes)
IFG (12 Bytes)

Standard DPDK routines are used to perform Transport Layer functionality.

VLAN tag functionality is offloaded to NIC as per the configuration of VF (refer to Appendix A, Setup Configuration).

The transport header is defined in the O-RAN Fronthaul specification based on the eCPRI specification, Refer to Table 2.

Figure 12. eCPRI Header Field Definitions

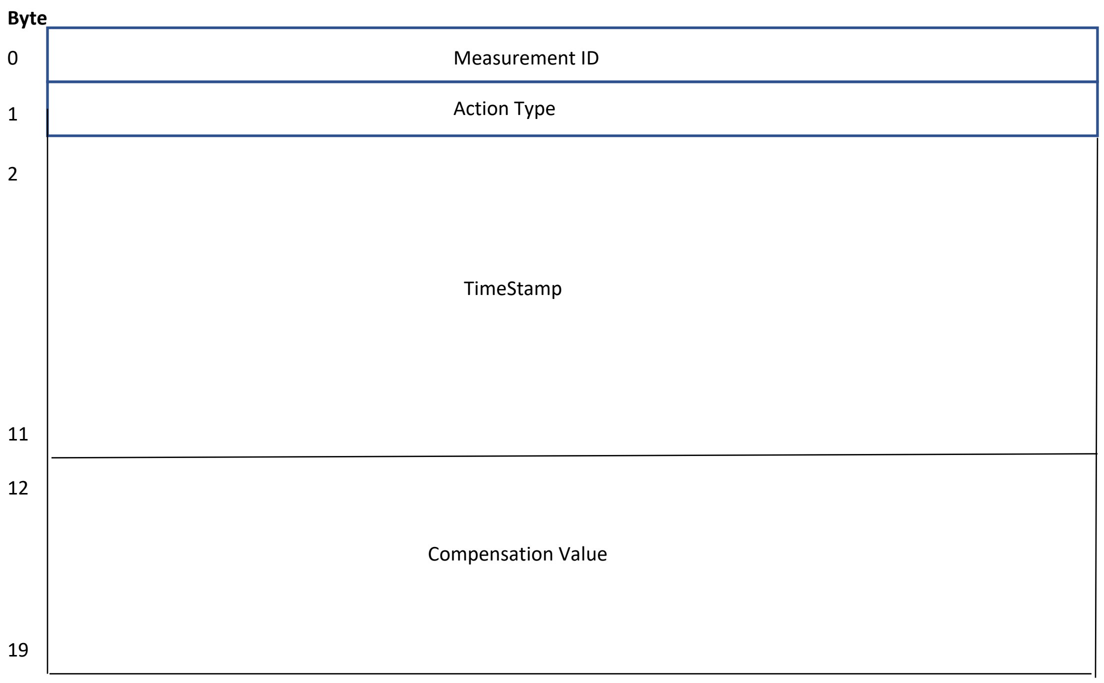


Section Type : any									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
ecpriVersion				ecpriReserved			EcpriConcatenation	1	Octet 1
ecpriMessage								1	Octet 2
ecpriPayload								2	Octet 3
ecpriRtcid / ecpriPcid								2	Octet 5
ecpriSeqid								2	Octet 7

Only `ECPRI_IQ_DATA = 0x00`, `ECPRI_RT_CONTROL_DATA= 0x02 and ECPRI_DELAY_MEASUREMENT` message types are supported.

For one-way delay measurements the eCPRI Header Field Definitions are the same as above until the ecpriPayload. The one-delay measurement message format is shown in the next figure.

Figure 13. ecpri one-way delay measurement message



In addition, for the eCPRI one-delay measurement message there is a requirement of dummy bytes insertion so the overall ethernet frame has at least 64 bytes.

The measurement ID is a one-byte value used by the sender of the request to distinguish the response received between different measurements.

The action type is a one-byte value defined in Table 8 of the eCPRI Specification V2.0.

Action Type 0x00 corresponds to a Request

Action Type 0x01 corresponds to a Request with Follow Up

Both values are used by an eCPRI node to initiate a one-way delay measurement in the direction of its own node to another node.

Action Type 0x02 corresponds to a Response

Action Type 0x03 is a Remote Request

Action Type 0x04 is a Remote Request with Follow Up

Values 0x03 and 0x04 are used when an eCPRI node needs to know the one-way delay from another node to itself.

Action Type 0x05 is the Follow_Up message.

The timestamp uses the IEEE-1588 Timestamp format with 8 bytes for the seconds part and 4 bytes for the nanoseconds part. The timestamp is a positive time with respect to the epoch.

The compensation value is used with Action Types 0x00 (Request), 0x02 (Response) or 0x05 (Follow_up) for all others this field contains zeros. This value is the compensation time measured in nanoseconds and multiplied by 2^{16} and follows the format for the correctionField in the common message header specified by the IEE 1588-2008 clause 13.3.

Handling of `ecpriRtcid`/`ecpriPcid` Bit field size is configurable and can be defined on the initialization stage of the xRAN library.

Figure 14. Bit Allocations of `ecpriRtcid/ecpriPcid`

0 (msb)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 (lsb)	Number of Octets
CU_Port_ID				BandSector_ID				CC_ID				RU_Port_ID				2

For `ecpriSeqid` only, the support for a sequence number is implemented. The following number is not supported.

Comments in the source code can be used to see more information on the implementation specifics of handling this field.

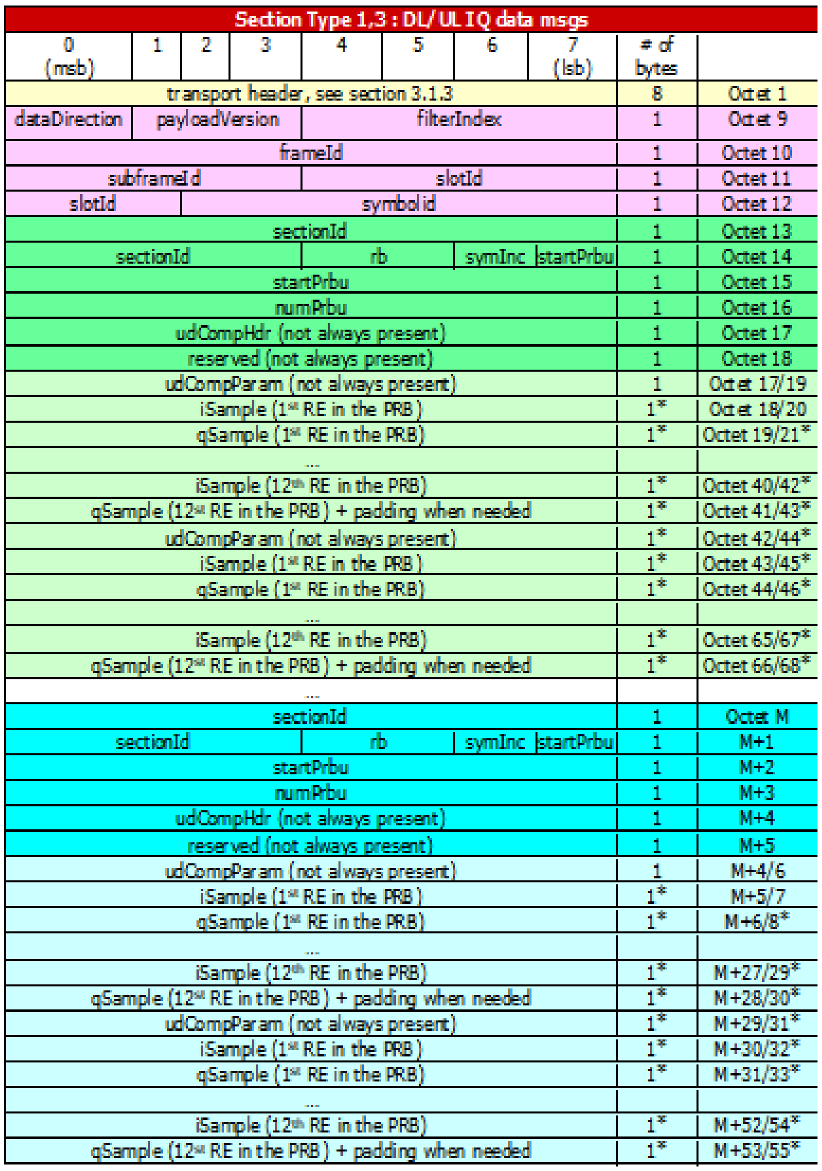
4.4 U-plane

The following diagrams show O-RAN packet protocols' headers and data arrangement with and without compression support.

XRAN packet meant for traffic with compression enabled has the Compression Header added after each Application Header. According to *O-RAN Fronthaul's specification* (Refer to Table 2), the Compression Header is part of a repeated Section Application Header. In the xRAN library implementation, the header is implemented as a separate structure, following the Application Section Header. As a result, the Compression Header is not included in the O-RAN packet, if compression is not used.

Figure 15 shows the components of an O-RAN packet.

Figure 15. O-RAN Packet Components

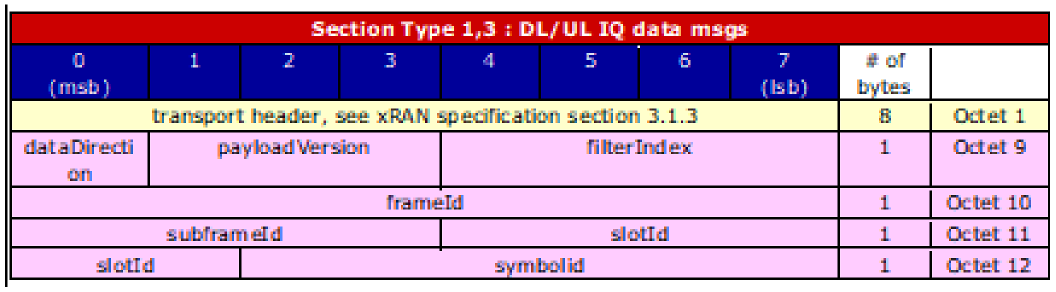


Section Type 1,3 : DL/ UL IQ data msgs									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		symbolid						1	Octet 12
sectionId								1	Octet 13
sectionId				rb		symInc	startPrbu	1	Octet 14
startPrbu								1	Octet 15
numPrbu								1	Octet 16
udCompHdr (not always present)								1	Octet 17
reserved (not always present)								1	Octet 18
udCompParam (not always present)								1	Octet 17/19
iSample (1st RE in the PRB)								1*	Octet 18/20
qSample (1st RE in the PRB)								1*	Octet 19/21*
...									
iSample (12th RE in the PRB)								1*	Octet 40/42*
qSample (12st RE in the PRB) + padding when needed								1*	Octet 41/43*
udCompParam (not always present)								1*	Octet 42/44*
iSample (1st RE in the PRB)								1*	Octet 43/45*
qSample (1st RE in the PRB)								1*	Octet 44/46*
...									
iSample (12th RE in the PRB)								1*	Octet 65/67*
qSample (12st RE in the PRB) + padding when needed								1*	Octet 66/68*
...									
sectionId								1	Octet M
sectionId				rb		symInc	startPrbu	1	M+1
startPrbu								1	M+2
numPrbu								1	M+3
udCompHdr (not always present)								1	M+4
reserved (not always present)								1	M+5
udCompParam (not always present)								1	M+4/6
iSample (1st RE in the PRB)								1*	M+5/7
qSample (1st RE in the PRB)								1*	M+6/8*
...									
iSample (12th RE in the PRB)								1*	M+27/29*
qSample (12st RE in the PRB) + padding when needed								1*	M+28/30*
udCompParam (not always present)								1*	M+29/31*
iSample (1st RE in the PRB)								1*	M+30/32*
qSample (1st RE in the PRB)								1*	M+31/33*
...									
iSample (12th RE in the PRB)								1*	M+52/54*
qSample (12st RE in the PRB) + padding when needed								1*	M+53/55*

4.4.1 Radio Application Header

The next header is a common header used for time reference.

Figure 16. Radio Application Header



Section Type 1,3 : DL/UL IQ data msgs									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see xRAN specification section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		symbolid						1	Octet 12

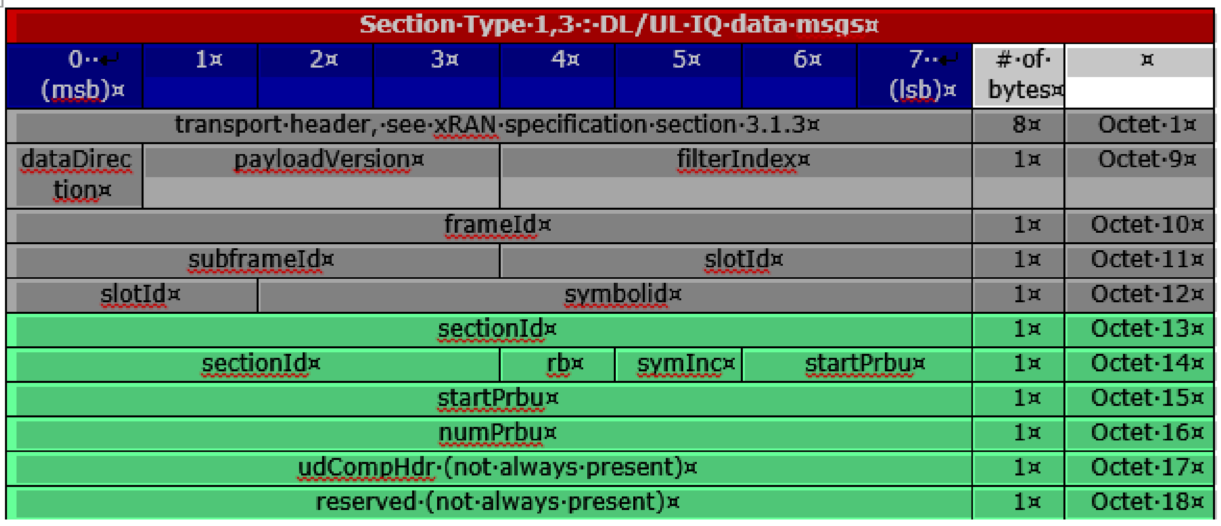
The radio application header specific field values are implemented as follows:

- `filterIndex = 0`
- `frameId = [0:99]`
- `subframeId = [0:9]`
- `slotId = [0:7]`
- `symbolId = [0:13]`

4.4.2 Data Section Application Data Header

The Common Radio Application Header is followed by the Application Header that is repeated for each Data Section within the eCPRI message. The relevant section of the O-RAN packet is shown in color.

Figure 17. Data Section Application Data Header



Section Type 1,3 : DL/UL IQ data msgs									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see xRAN specification section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		symbolid						1	Octet 12
sectionId								1	Octet 13
sectionId				rb	symInc	startPrbu		1	Octet 14
startPrbu								1	Octet 15
numPrbu								1	Octet 16
udCompHdr (not always present)								1	Octet 17
reserved (not always present)								1	Octet 18

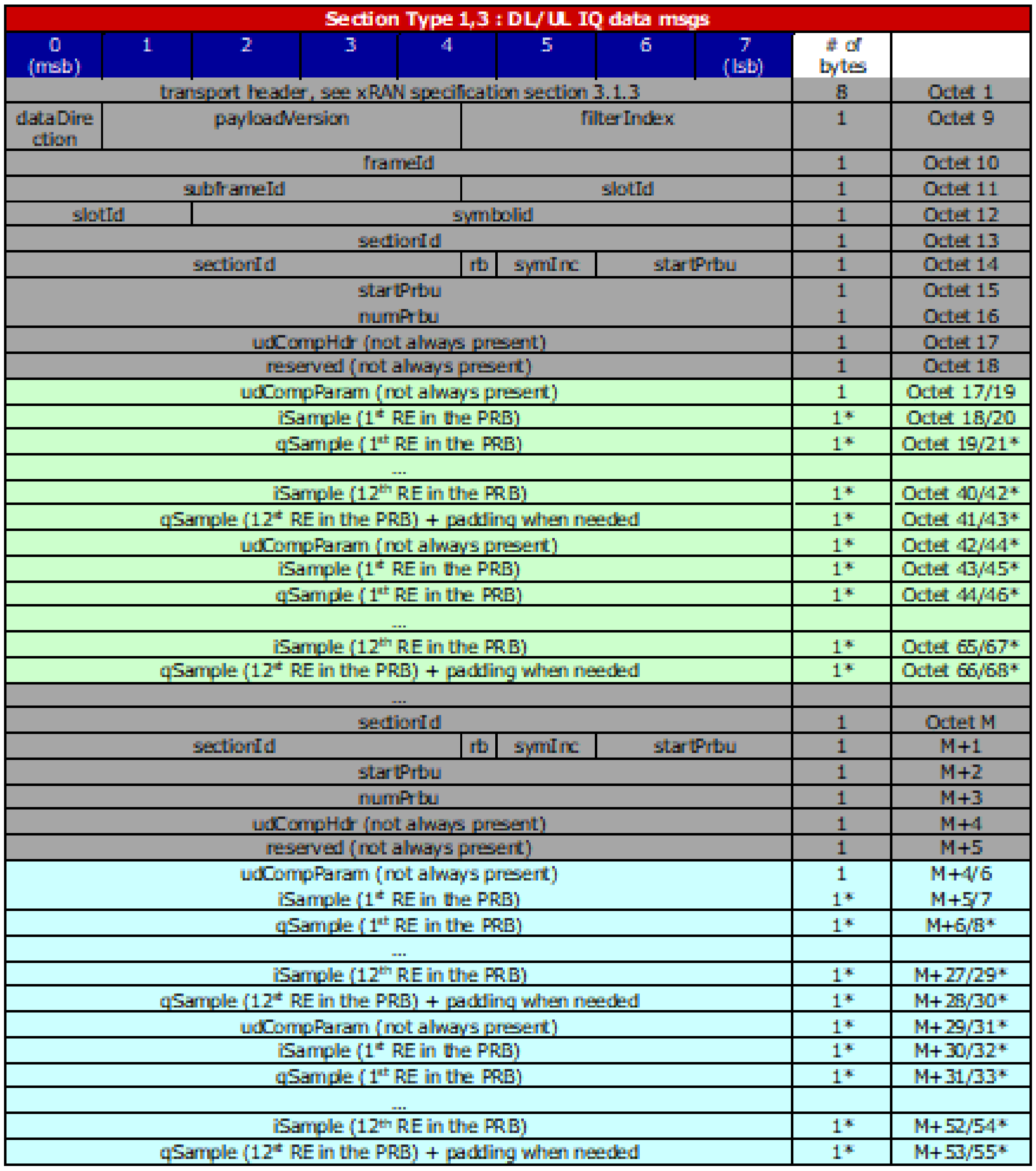
A single section is used per one Ethernet packet with IQ samples `startPrbu` is equal to 0, and `numPrbu` is `wqual` to the number of RBs used:

- `rb` field is not used (value 0).
- `symInc` is not used (value 0)

4.4.3 Data Payload

An O-RAN packet data payload contains several PRBs. Each PRB is built of 12 IQ samples. Flexible IQ bit width is supported. If compression is enabled, `udCompParam` is included in the data payload. The data section is shown in color.

Figure 18. Data Payload



Section Type 1,3 : DL/UL IQ data msgs									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see xRAN specification section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		symbolid						1	Octet 12
sectionId								1	Octet 13
sectionId				rb	symInc	startPrbu		1	Octet 14
startPrbu								1	Octet 15
numPrbu								1	Octet 16
udCompHdr (not always present)								1	Octet 17
reserved (not always present)								1	Octet 18
udCompParam (not always present)								1	Octet 17/19
iSample (1st RE in the PRB)								1*	Octet 18/20
qSample (1st RE in the PRB)								1*	Octet 19/21*
...									
iSample (12th RE in the PRB)								1*	Octet 40/42*
qSample (12th RE in the PRB) + padding when needed								1*	Octet 41/43*
udCompParam (not always present)								1*	Octet 42/44*
iSample (1st RE in the PRB)								1*	Octet 43/45*
qSample (1st RE in the PRB)								1*	Octet 44/46*
...									
iSample (12th RE in the PRB)								1*	Octet 65/67*
qSample (12th RE in the PRB) + padding when needed								1*	Octet 66/68*
...									
sectionId								1	Octet M
sectionId				rb	symInc	startPrbu		1	M+1
startPrbu								1	M+2
numPrbu								1	M+3
udCompHdr (not always present)								1	M+4
reserved (not always present)								1	M+5
udCompParam (not always present)								1	M+4/6
iSample (1st RE in the PRB)								1*	M+5/7
qSample (1st RE in the PRB)								1*	M+6/8*
...									
iSample (12th RE in the PRB)								1*	M+27/29*
qSample (12th RE in the PRB) + padding when needed								1*	M+28/30*
udCompParam (not always present)								1*	M+29/31*
iSample (1st RE in the PRB)								1*	M+30/32*
qSample (1st RE in the PRB)								1*	M+31/33*
...									
iSample (12th RE in the PRB)								1*	M+52/54*
qSample (12th RE in the PRB) + padding when needed								1*	M+53/55*

4.5 C-plane

C-Plane messages are encapsulated using a two-layered header approach. The first layer consists of an eCPRI standard header, including corresponding fields used to indicate the message type, while the second layer is an application layer, including necessary fields for control and synchronization. Within the application layer, a "section" defines the characteristics of U-plane data to be transferred or received from a beam with one pattern id. In general, the transport header, application header, and sections are all intended to be aligned on 4-byte boundaries and are transmitted in "network byte order" meaning the most significant byte of a multi-byte parameter is transmitted first.

Table 9 is a list of sections currently supported.

Table 9. Section Types

Section Type	Target Scenario	Remarks
0	Unused Resource Blocks or symbols in Downlink or Uplink	Not supported
1	Most DL/UL radio channels	Supported
2	reserved for future use	N/A
3	PRACH and mixed-numerology channels	Only PRACH is supported. Mixed numerology is not supported.
4	Reserved for future use	Not supported
5	UE scheduling information (UE-ID assignment to section)	Not supported
6	Channel information	Not supported
7	LAA	Not supported
8-255	Reserved for future use	N/A

Section extensions are not supported in this release.

The definition of the C-Plane packet can be found `lib/api/xran_pkt_cp.h`, and the fields are appropriately re-ordered to apply the conversion of network byte order after setting values.

The comments in the source code of xRAN lib can be used to see more information on implementation specifics of handling sections as well as particular fields. Additional changes may be needed on the C-plane to perform IOT with O-RU depending on the scenario

4.5.1 Ethernet Header

Refer to Figure 11.

4.5.2 eCPRI Header

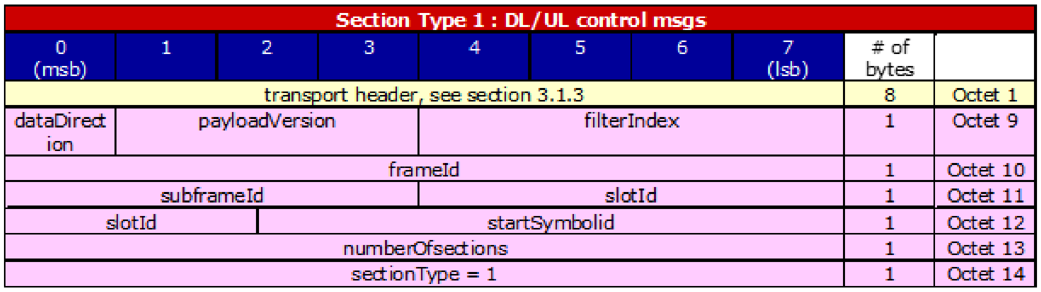
Refer to Figure 12.

This header is defined as the structure of `xran_ecpri_hdr` in `lib/api/xran_pkt.h`.

4.5.3 Radio Application Common Header

The Radio Application Common Header is used for time reference. Its structure is shown in Figure 19.

Figure 19. Radio Application Common Header



Section Type 1 : DL/UL control msgs

0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		startSymbolid						1	Octet 12
numberOfsections								1	Octet 13
sectionType = 1								1	Octet 14

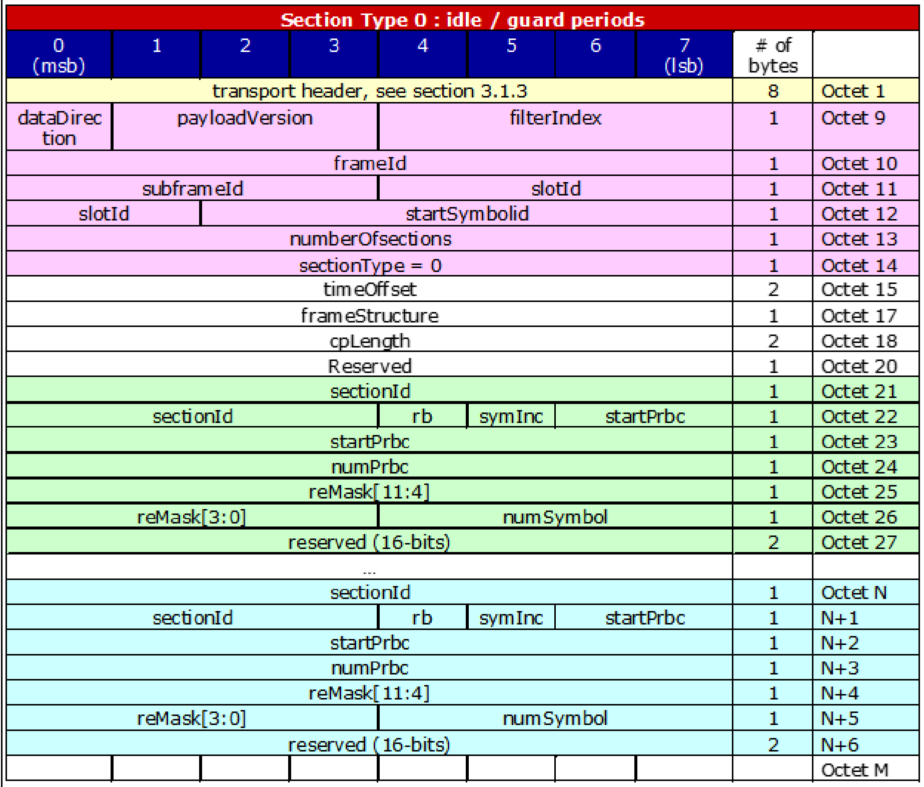
This header is defined as the structure of `xran_cp_radioapp_common_header` in `lib/api/xran_pkt_cp.h`.

NOTE: The payload version in this header is fixed to `XRAN_PAYLOAD_VER` (defined as 1) in this release.

4.5.4 Section Type 0 Structure

Figure 20 describes the structure of Section Type 0.

Figure 20. Section Type 0 Structure



Section Type 0 : idle / guard periods

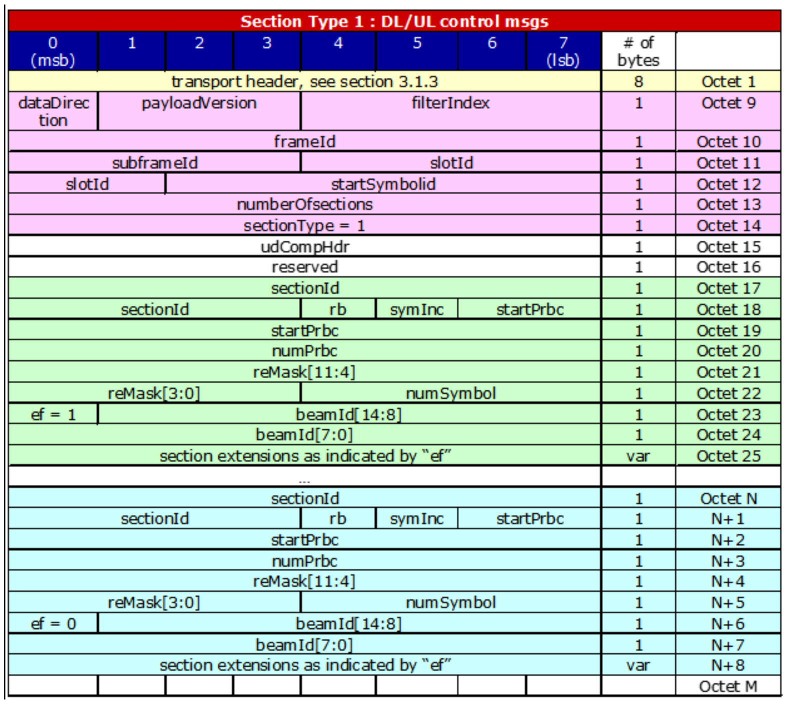
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		startSymbolid						1	Octet 12
numberOfsections								1	Octet 13
sectionType = 0								1	Octet 14
timeOffset								2	Octet 15
frameStructure								1	Octet 17
cpLength								2	Octet 18
Reserved								1	Octet 20
sectionId								1	Octet 21
sectionId				rb	symInc	startPrbc		1	Octet 22
startPrbc								1	Octet 23
numPrbc								1	Octet 24
reMask[11:4]								1	Octet 25
reMask[3:0]				numSymbol				1	Octet 26
reserved (16-bits)								2	Octet 27
...									
sectionId								1	Octet N
sectionId				rb	symInc	startPrbc		1	N+1
startPrbc								1	N+2
numPrbc								1	N+3
reMask[11:4]								1	N+4
reMask[3:0]				numSymbol				1	N+5
reserved (16-bits)								2	N+6
									Octet M

NOTE: In Figure 19 through Figure 23, the color yellow means it is a transport header; the color pink is the radio application header; others are repeated sections.

4.5.5 Section Type 1 Structure

Figure 21 describes the structure of Section Type 1.

Figure 21. Section Type 1 Structure



Section Type 1 : DL/UL control msgs									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		startSymbolid						1	Octet 12
numberOfsections								1	Octet 13
sectionType = 1								1	Octet 14
udCompHdr								1	Octet 15
reserved								1	Octet 16
sectionId								1	Octet 17
sectionId				rb	symInc	startPrbc		1	Octet 18
startPrbc								1	Octet 19
numPrbc								1	Octet 20
reMask[11:4]								1	Octet 21
reMask[3:0]				numSymbol				1	Octet 22
ef = 1	beamId[14:8]							1	Octet 23
beamId[7:0]								1	Octet 24
section extensions as indicated by "ef"								var	Octet 25
...									
sectionId								1	Octet N
sectionId				rb	symInc	startPrbc		1	N+1
startPrbc								1	N+2
numPrbc								1	N+3
reMask[11:4]								1	N+4
reMask[3:0]				numSymbol				1	N+5
ef = 0	beamId[14:8]							1	N+6
beamId[7:0]								1	N+7
section extensions as indicated by "ef"								var	N+8
									Octet M

Section Type 1 message has two additional parameters in addition to radio application common header:

- udCompHdr : defined as the structure of xran_radioapp_udComp_header
- reserved : fixed by zero

Section type 1 is defined as the structure of xran_cp_radioapp_section1, and this part can be repeated to have multiple sections.

Whole section type 1 message can be described in this summary:

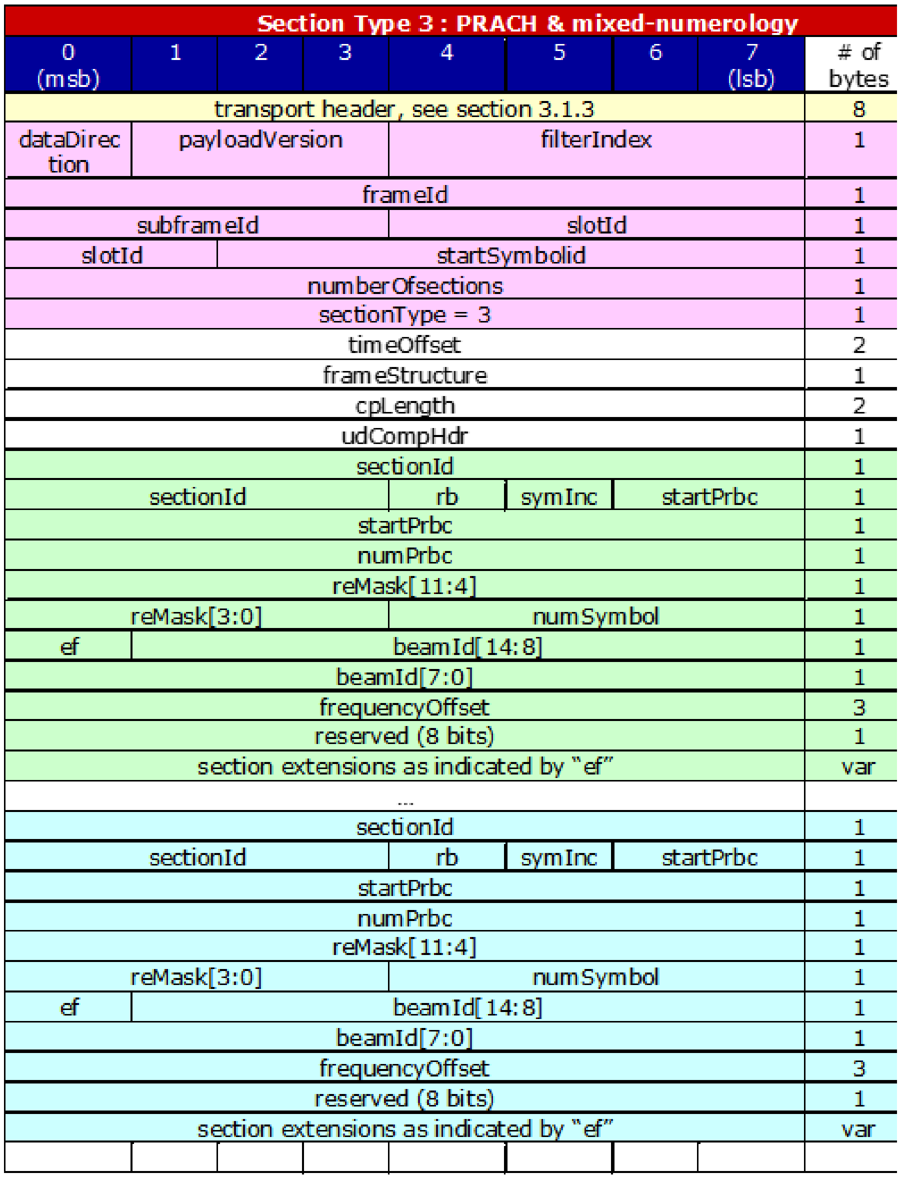
xran_cp_radioapp_common_header
xran_cp_radioapp_section1_header
xran_cp_radioapp_section1
……
xran_cp_radioapp_section1

NOTE: Even though API function can support to compose multiple sections in a C-Plane message, current implementation is to compose single section per a C-Plane message.

4.5.6 Section Type 3 Structure

Figure 22 describes the structure of Section Type 3.

Figure 22. Section Type 3 Structure



Section Type 3 : PRACH & mixed-numerology								
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes
transport header, see section 3.1.3								8
dataDirection	payloadVersion			filterIndex				1
frameId								1
subframeId				slotId				1
slotId		startSymbolid						1
numberOfsections								1
sectionType = 3								1
timeOffset								2
frameStructure								1
cpLength								2
udCompHdr								1
sectionId								1
sectionId				rb	symInc	startPrbc		1
startPrbc								1
numPrbc								1
reMask[11:4]								1
reMask[3:0]				numSymbol				1
ef	beamId[14:8]							1
beamId[7:0]								1
frequencyOffset								3
reserved (8 bits)								1
section extensions as indicated by "ef"								var
...								
sectionId								1
sectionId				rb	symInc	startPrbc		1
startPrbc								1
numPrbc								1
reMask[11:4]								1
reMask[3:0]				numSymbol				1
ef	beamId[14:8]							1
beamId[7:0]								1
frequencyOffset								3
reserved (8 bits)								1
section extensions as indicated by "ef"								var

Section Type 3 message has below four additional parameters in addition to radio application common header.

- `timeOffset`
- `frameStrucutre`: defined as the structure of `xran_cp_radioapp_frameStructure`
- `cpLength`
- `udCompHd`r: defined as the structure of `xran_radioapp_udComp_header`

Section Type 3 is defined as the structure of `xran_cp_radioapp_section3` and this part can be repeated to have multiple sections.

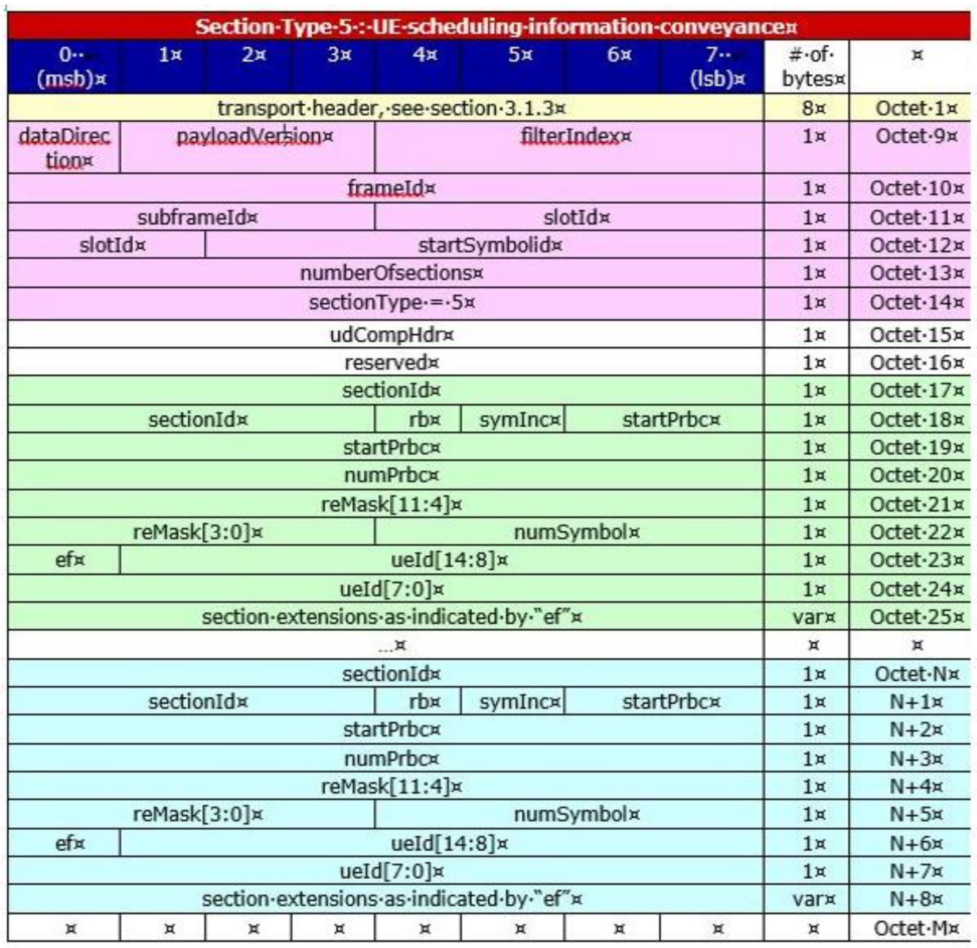
Whole section type 3 message can be described in this summary:

`xran_cp_radioapp_common_header`
`xran_cp_radioapp_section3_header`
`xran_cp_radioapp_section3`
……
`xran_cp_radioapp_section3`

4.5.7 Section Type 5 Structure

Figure 23 describes the structure of Section Type 5.

Figure 23. Section Type 5 Structure

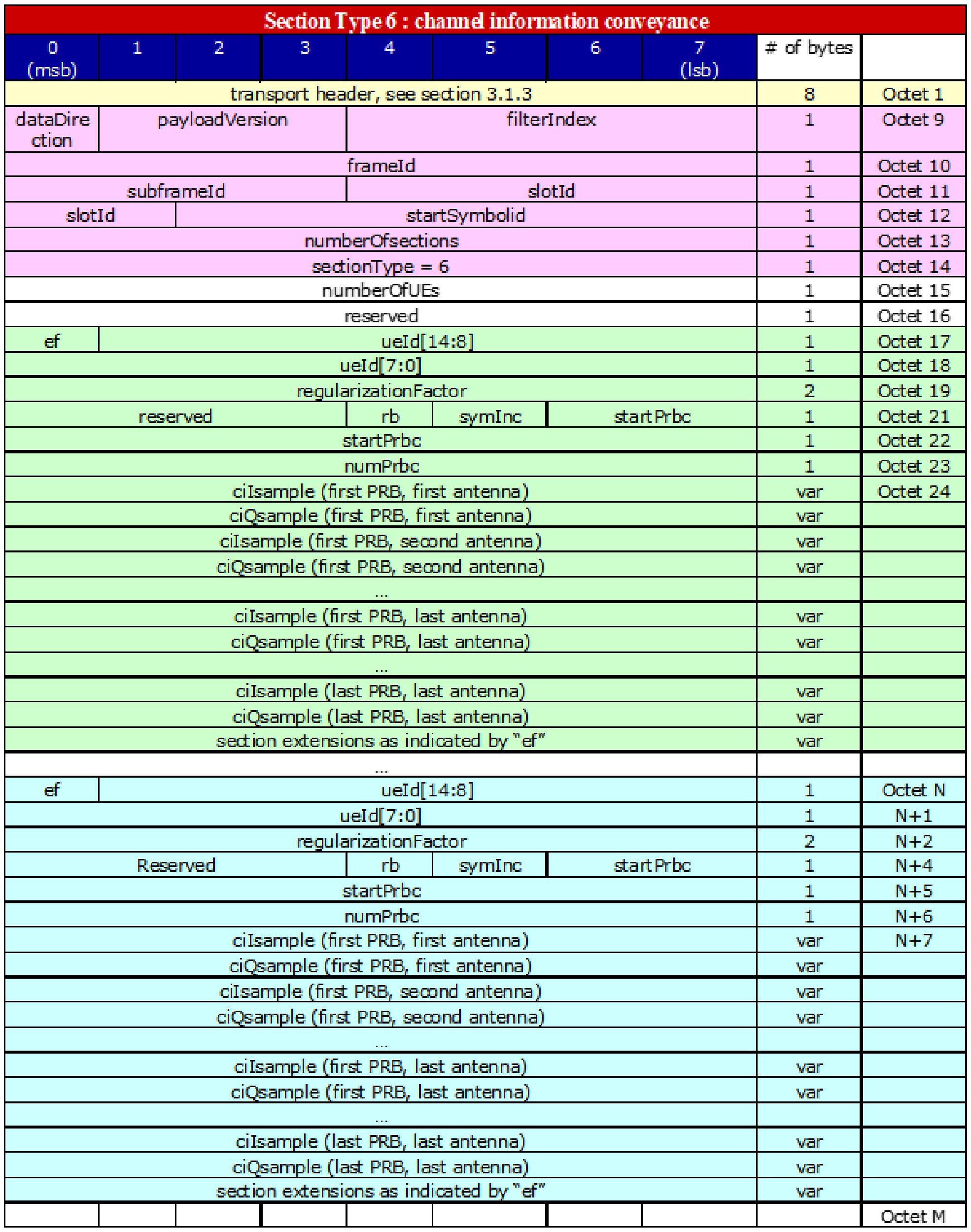


Section Type 5 : UE scheduling information conveyance									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		startSymbolid						1	Octet 12
numberOfsections								1	Octet 13
sectionType = 5								1	Octet 14
udCompHdr								1	Octet 15
reserved								1	Octet 16
sectionId								1	Octet 17
sectionId				rb	symInc	startPrbc		1	Octet 18
startPrbc								1	Octet 19
numPrbc								1	Octet 20
reMask[11:4]								1	Octet 21
reMask[3:0]				numSymbol				1	Octet 22
ef	ueId[14:8]							1	Octet 23
ueId[7:0]								1	Octet 24
section extensions as indicated by "ef"								var	Octet 25
…									
sectionId								1	Octet N
sectionId				rb	symInc	startPrbc		1	N+1
startPrbc								1	N+2
numPrbc								1	N+3
reMask[11:4]								1	N+4
reMask[3:0]				numSymbol				1	N+5
ef	ueId[14:8]							1	N+6
ueId[7:0]								1	N+7
section extensions as indicated by "ef"								var	N+8
									Octet M

4.5.8 Section Type 6 Structure

Figure 24 describes the structure of Section Type 6.

Figure 24. Section Type 6 Structure



Section Type 6 : channel information conveyance									
0 (msb)	1	2	3	4	5	6	7 (lsb)	# of bytes	
transport header, see section 3.1.3								8	Octet 1
dataDirection	payloadVersion			filterIndex				1	Octet 9
frameId								1	Octet 10
subframeId				slotId				1	Octet 11
slotId		startSymbolid						1	Octet 12
numberOfsections								1	Octet 13
sectionType = 6								1	Octet 14
numberOfUEs								1	Octet 15
reserved								1	Octet 16
ef	ueId[14:8]							1	Octet 17
ueId[7:0]								1	Octet 18
regularizationFactor								2	Octet 19
reserved				rb	symInc	startPrbc		1	Octet 21
startPrbc								1	Octet 22
numPrbc								1	Octet 23
ciIsample (first PRB, first antenna)								var	Octet 24
ciQsample (first PRB, first antenna)								var	
ciIsample (first PRB, second antenna)								var	
ciQsample (first PRB, second antenna)								var	
…									
ciIsample (first PRB, last antenna)								var	
ciQsample (first PRB, last antenna)								var	
…									
ciIsample (last PRB, last antenna)								var	
ciQsample (last PRB, last antenna)								var	
section extensions as indicated by "ef"								var	
…									
ef	ueId[14:8]							1	Octet N
ueId[7:0]								1	N+1
regularizationFactor								2	N+2
Reserved				rb	symInc	startPrbc		1	N+4
startPrbc								1	N+5
numPrbc								1	N+6
ciIsample (first PRB, first antenna)								var	N+7
ciQsample (first PRB, first antenna)								var	
ciIsample (first PRB, second antenna)								var	
ciQsample (first PRB, second antenna)								var	
…									
ciIsample (first PRB, last antenna)								var	
ciQsample (first PRB, last antenna)								var	
…									
ciIsample (last PRB, last antenna)								var	
ciQsample (last PRB, last antenna)								var	
section extensions as indicated by "ef"								var	
									Octet M

§

5.0 xRAN Library Design

The xRAN Library consists of multiple modules where different functionality is encapsulated. The complete list of all `*.c` and `*.h` files, as well as Makefile for xRAN v21.03 release, is:

```
├── app
│   ├── dpdk.sh
│   ├── gen_test.m
│   ├── ifft_in.txt
│   ├── Makefile
│   ├── run_o_du.sh
│   ├── run_o_ru.sh
│   ├── src
│   │   ├── app_io_fh_xran.c
│   │   ├── app_io_fh_xran.h
│   │   ├── common.c
│   │   ├── common.h
│   │   ├── config.c
│   │   ├── config.h
│   │   ├── debug.h
│   │   ├── sample-app.c
│   │   └── xran_mlog_task_id.h
│   └── usecase
│       ├── cat_a
│       ├── cat_b
│       ├── lte_a
│       ├── lte_b
├── banner.txt
├── build.sh
├── lib
│   ├── api
│   │   ├── xran_compression.h
│   │   ├── xran_compression.hpp
│   │   ├── xran_cp_api.h
│   │   ├── xran_fh_o_du.h
│   │   ├── xran_mlog_lnx.h
│   │   ├── xran_pkt_cp.h
│   │   ├── xran_pkt.h
│   │   ├── xran_pkt_up.h
│   │   ├── xran_sync_api.h
│   │   ├── xran_timer.h
│   │   ├── xran_transport.h
│   │   └── xran_up_api.h
│   ├── ethernet
│   │   ├── ethdi.c
│   │   ├── ethdi.h
│   │   ├── ethernet.c
│   │   └── ethernet.h
│   ├── Makefile
│   └── src
│       ├── xran_app_frag.c
│       ├── xran_app_frag.h
│       ├── xran_bfp_byte_packing_utils.hpp
│       ├── xran_bfp_cplane16.cpp
│       ├── xran_bfp_cplane16_snc.cpp
│       ├── xran_bfp_cplane32.cpp
│       ├── xran_bfp_cplane32_snc.cpp
│       ├── xran_bfp_cplane64.cpp
│       ├── xran_bfp_cplane64_snc.cpp
│       ├── xran_bfp_cplane8.cpp
│       ├── xran_bfp_cplane8_snc.cpp
│       ├── xran_bfp_ref.cpp
│       ├── xran_bfp_uplane_9b16rb.cpp
│       ├── xran_bfp_uplane.cpp
│       ├── xran_bfp_uplane_snc.cpp
│       ├── xran_bfp_utils.hpp
│       ├── xran_cb_proc.c
│       ├── xran_cb_proc.h
│       ├── xran_common.c
│       ├── xran_common.h
│       ├── xran_compression.cpp
│       ├── xran_compression_snc.cpp
│       ├── xran_cp_api.c
│       ├── xran_cp_proc.c
│       ├── xran_cp_proc.h
│       ├── xran_delay_measurement.c
│       ├── xran_dev.c
│       ├── xran_dev.h
│       ├── xran_ecpri_owd_measurements.h
│       ├── xran_frame_struct.c
│       ├── xran_frame_struct.h
│       ├── xran_lib_mlog_tasks_id.h
│       ├── xran_main.c
│       ├── xran_main.h
│       ├── xran_mod_compression.cpp
│       ├── xran_mod_compression.h
│       ├── xran_prach_cfg.h
│       ├── xran_printf.h
│       ├── xran_rx_proc.c
│       ├── xran_rx_proc.h
│       ├── xran_sync_api.c
│       ├── xran_timer.c
│       ├── xran_transport.c
│       ├── xran_tx_proc.c
│       ├── xran_tx_proc.h
│       ├── xran_ul_tables.c
│       └── xran_up_api.c
├── Licenses.txt
├── misc
│   └── icx-sp-cvl.sh
├── readme.md
└── test
    ├── common
    │   ├── common.cpp
    │   ├── common.hpp
    │   ├── common_typedef_xran.h
    │   ├── json.hpp
    │   ├── MIT_License.txt
    │   ├── xranlib_unit_test_main.cc
    │   └── xran_lib_wrap.hpp
    ├── master.py
    ├── readme.txt
    └── test_xran
        ├── chain_tests.cc
        ├── compander_functional.cc
        ├── conf.json
        ├── c_plane_tests.cc
        ├── init_sys_functional.cc
        ├── Makefile
        ├── mod_compression_unit_test.cc
        ├── prach_functional.cc
        ├── prach_performance.cc
        ├── unittests.cc
        ├── u_plane_functional.cc
        └── u_plane_performance.cc
```

5.1 General Introduction

The xRAN Library functionality is broken down into two main sections:

- O-RAN specific packet handling (src)
- Ethernet and supporting functionality (ethernet)

External functions and structures are available via a set of header files in the API folder.

This library depends on DPDK primitives to perform Ethernet networking in user space, including initialization and control of Ethernet ports. Ethernet ports are expected to be SRIOV virtual functions (VF) but also can be physical functions (PF) as well

This library is expected to be included in the project via `xran_fh_o_du.h`, statically compiled and linked with the L1 application as well as DPDK libraries. The O-RAN packet processing-specific functionality is encapsulated into this library and not exposed to the rest of the 5G NR pipeline.

This way, xRAN specific changes are decoupled from the L1 pipeline. As a result, the design and implementation of the 5G L1 pipeline code and xRAN library can be done in parallel, provided the defined interface is not modified.

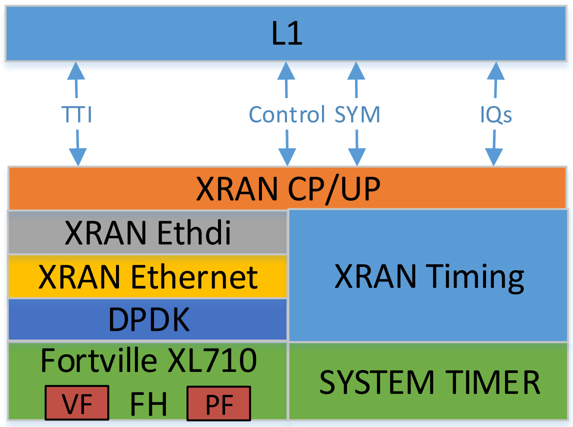
Ethernet consists of two modules:

- Ethernet implements xRAN specific HW Ethernet initialization, close, send and receive
- `ethdi` provides Ethernet level software primitives to handle O-RAN packet exchange

The xRAN layer implements the next set of functionalities:

- Common code specific for both C-plane and U-plane as well as TX and RX
- Implementation of C-plane API available within the library and externally
- The primary function where general library initialization and configuration performed
- Module to provide the status of PTP synchronization
- Timing module where system time is polled
- eCPRI specific transport layer functions
- APIs to handle U-plane packets
- A set of utility modules for debugging (`printf`) and data tables are included as well.

Figure 25. Illustration of xRAN Sublayers



A detailed description of functions and input/output arguments, as well as key data structures, can be found in the Doxygen file for the FlexRAN 5G NR release, Refer to Table 2. In this document, supplemental information is provided for the overall design and implementation assumptions.

5.2 Initialization and Close

An example of the initialization sequence can be found in the sample application code. It consists of the following steps:

1. Setup structure struct `xran_fh_init` according to configuration.
2. Call `xran_init()` to instantiate the xRAN lib memory model and threads. The function returns a pointer to xRAN handle, which is used for consecutive configuration functions.
3. Initialize memory buffers used for L1 and xRAN exchange of information.
4. Assign callback functions for (one) TTI event and the reception of half of the slot of symbols (7 symbols) and Full slot of symbols (14 symbols).
5. Call `xran_open()` to initialize PRACH configuration, initialize DPDK, and launch xRAN timing thread,
6. Call `xran_start()` to start processing O-RAN packets for DL and UL

After this is complete 5G L1 runs with xRAN Front haul interface. During run time for every TTI event, the corresponding call back is called. For packet reception on UL direction, the corresponding call back is called. OTA time information such as frame id, subframe id, and slot id can be obtained as result synchronization of the L1 pipeline to GPS time is performed.

To stop and close the interface, perform this sequence of steps:

1. Call `xran_stop()` to stop the processing of DL and UL
2. Call `xran_close()` to remove usage of xRAN resources
3. Call `xran_mm_destroy()` to destroy memory management subsystem

After this session is complete, a restart of the full L1 application is required. The current version of the library does not support multiple sessions without a restart of the full L1 application.

5.2.1 Configuration

The xRAN library configuration is provided in the set of structures, such as struct `xran_fh_init` and struct `xran_fh_config`. The sample application gives an example of a test configuration used for LTE and 5GNR mmWave, and Sub 6. Sample application folder `app/usecase/contains` set of examples for different Radio Access technology (LTE|5G NR) deferent category (A|B) and list of numerologies (0,1,3) and list of bandwidth (5,10,20,100 MHz)

NOTE: Some configuration options are not used in v21.03 and are reserved for future use.

These options are available: (refer to *FlexRAN 5G NR Reference Solution RefPHY* (Doxygen); Table 2):

Structure struct `xran_fh_init`:

- Number of CC and corresponding settings for each
- Core allocation for xRAN
- Ethernet port allocation
- O-DU and RU Ethernet Mac address
- Timing constraints of O-DU and 0-RU
- Debug features

Structure struct `xran_fh_config`:

- Number of `eAxC`
- TTI Callback function and parameters
- PRACH 5G NR specific settings
- TDD frame configuration
- BBU specific configuration
- RU specific configuration

From an implementation perspective:

The `xran_init()` performs `init` of the xRAN library and interface according to struct `xran_fh_init` information as per the start of application configuration:

- Init DPDK with corresponding networking ports and core assignment
- Init `mbuf` pools
- Init DPDK timers and DPDK rings for internal packet processing
- Instantiate ORAH FH thread doing
  - Timing processing (`xran_timing_source_thread()`)
  - ETH PMD (`process_dpdk_io()`)
  - IO XRAN-PHY exchange (`ring_processing_func()`)

The `xran_open()` performs additional configuration as per run scenario:

- PRACH configuration
- C-plane initialization

The function xran_close() performs free of resources and allows the potential restart of the front haul interface with a different scenario.

5.2.2 Start/Stop

Functions xran_start()/xran_stop() enable/disable packet processing for both the DL and UL. This triggers the execution of callbacks into the L1 application.

5.2.3 Data Exchange

Exchange of IQ samples, as well as C-plane specific information, is performed using a set of buffers allocated by xRAN library from DPDK memory and shared with the l1 application. Buffers are allocated as a standard mbuf structure, and DPDK pools are used to manage the allocation and free resources. Shared buffers are allocated at the init stage and are expected to be reused within 80 TTIs (10 ms).

The xRAN protocol requires U-plane IQ data to be transferred in network byte order, and the L1 application handles IQ sample data in CPU byte order, requiring a swap. The PHY BBU pooling tasks perform copy and byte order swap during packet processing.

5.2.4 C-plane Information Settings

The interface between the xRAN library and PHY is defined via struct xran_prb_map and similar to the data plane. The same mbuf memory is used to allocate memory map of PRBs for each TTI.

```
/** Beamforming waights for single stream for each PRBs  given number of Antenna
elements */
struct xran_cp_bf_weight{
    int16_t nAntElmTRx;         /**< num TRX for this allocation */
    int16_t  ext_section_sz;    /**< extType section size */
    int8_t*  p_ext_start;       /**< pointer to start of buffer for full C-plane packet
*/
    int8_t*  p_ext_section;     /**< pointer to form extType */

    /* For ext 11 */
    uint8_t     bfwCompMeth;     /* Compression Method for BFW */
    uint8_t     bfwIqWidth;      /* Bitwidth of BFW */
    uint8_t     numSetBFWs;      /* Total number of beam forming weights set (L) */
    uint8_t     numBundPrb;      /* The number of bundled PRBs, 0 means to use ext1 */
    uint8_t     RAD;
    uint8_t     disableBFWs;
    int16_t     maxExtBufSize;   /* Maximum space of external buffer */
    struct xran_ext11_bfw_info bfw[XRAN_MAX_SET_BFWS]
};

/** PRB element structure */
struct xran_prb_elm {
    int16_t nRBStart;     /**< start RB of RB allocation */
    int16_t nRBSize;      /**< number of RBs used */
    int16_t nStartSymb;   /**< start symbol ID */
    int16_t numSymb;      /**< number of symbols */
    int16_t nBeamIndex;   /**< beam index for given PRB */
    int16_t bf_weight_update; /** need to update beam weights or not */
    int16_t compMethod;   /**< compression index for given PRB */
    int16_t iqWidth;      /**< compression bit width for given PRB */
    uint16_t ScaleFactor;  /**< scale factor for modulation compression */
    int16_t reMask;    /**< 12-bit RE Mask for modulation compression */
    int16_t BeamFormingType; /**< index based, weights based or attribute based beam
forming*/
    int16_t nSecDesc[XRAN_NUM_OF_SYMBOL_PER_SLOT]; /**<  number of section descriptors
per symbol */
    struct xran_section_desc *
p_sec_desc[XRAN_NUM_OF_SYMBOL_PER_SLOT][XRAN_MAX_FRAGMENT]; /**< section desctiptors
to U-plane data given RBs */
    struct xran_cp_bf_weight   bf_weight; /**< beam forming information relevant for
given RBs */
    union {
        struct xran_cp_bf_attribute bf_attribute;
        struct xran_cp_bf_precoding bf_precoding;
    };
};
/** PRB map structure */
struct xran_prb_map {
    uint8_t   dir;         /**< DL or UL direction */
    uint8_t   xran_port;   /**< xran id of given RU [0-(XRAN_PORTS_NUM-1)] */
    uint16_t  band_id;     /**< xran band id */
    uint16_t  cc_id;       /**< component carrier id [0 - (XRAN_MAX_SECTOR_NR-1)] */
    uint16_t  ru_port_id;  /**< RU device antenna port id [0 - (XRAN_MAX_ANTENNA_NR-1)
*/
    uint16_t  tti_id;      /**< xRAN slot id [0 - (max tti-1)] */
    uint8_t   start_sym_id;      /**< start symbol Id [0-13] */
    uint32_t  nPrbElm;     /**< total number of PRB elements for given map [0-
(XRAN_MAX_SECTIONS_PER_SLOT-1)] */
    struct xran_prb_elm prbMap[XRAN_MAX_SECTIONS_PER_SLOT];
};
```

C-plane sections are expected to be provided by L1 pipeline. If 100% of RBs always allocated single element of RB map is expected to be allocated across all symbols. Dynamic RB allocation is performed base on C-plane configuration.

The xRAN library will require that the content of the PRB map should be sorted in increasing order of PRB first and then symbols.

5.3 Memory Management

Memory used for the exchange of IQ data as well as control information, is controlled by the xRAN library. L1 application at the init stage performs:

- init memory management subsystem
- init buffer management subsystem (via DPDK pools)
- allocate buffers (mbuf) for each CC, antenna, symbol, and direction (DL, UL, PRACH) for `XRAN_N_FE_BUF_LEN` TTIs.
- buffers are reused for every `XRAN_N_FE_BUF_LEN` TTIs

After the session is completed, the application can free buffers and destroy the memory management subsystem.

From an implementation perspective, the xRAN library uses a standard mbuf primitive and allocates a pool of buffers for each sector. This function is performed using `rte_pktmbuf_pool_create()`, `rte_pktmbuf_alloc()`, and `rte_pktmbuf_append()` to allocate one buffer per symbol for the mmWave case. More information on mbuf and DPDK pools can be found in the DPDK documentation.

In the current implementation, mbuf, is the number of buffers shared with the L1 application is the same number of buffers used to send to and receive from the Ethernet port. Memory copy operations are not required if the packet size is smaller than or equal to MTU. Future versions of the xRAN library are required to remove the memory copy requirement for packets where the size larger than MTU.

5.3.1 External Interface Memory

The xRAN library header file defines a set of structures to simplify access to memory buffers used for IQ data.

```
struct xran_flat_buffer {
    uint32_t nElementLenInBytes;
    uint32_t nNumberOfElements;
    uint32_t nOffsetInBytes;
    uint32_t nIsPhyAddr;
    uint8_t *pData;
    void *pCtrl;
};
struct xran_buffer_list {
    uint32_t nNumBuffers;
    struct xran_flat_buffer *pBuffers;
    void *pUserData;
    void *pPrivateMetaData;
};
struct xran_io_buf_ctrl {
    /* -1-this subframe is not used in current frame format
         0-this subframe can be transmitted, i.e., data is ready
          1-this subframe is waiting transmission, i.e., data is not ready
         10 - DL transmission missing deadline. When FE needs this subframe data but
bValid is still 1,
        set bValid to 10.
    */
    int32_t bValid ; // when UL rx, it is subframe index.
    int32_t nSegToBeGen;
    int32_t nSegGenerated; // how many date segment are generated by DL LTE processing
or received from FE
                       // -1 means that DL packet to be transmitted is not ready in BS
    int32_t nSegTransferred; // number of data segments has been transmitted or
received
    struct rte_mbuf *pData[N_MAX_BUFFER_SEGMENT]; // point to DPDK allocated memory
pool
    struct xran_buffer_list sBufferList;
};
```

There is no explicit requirement for user to organize a set of buffers in this particular way. From a compatibility perspective it is useful to follow the existing design of the 5G NR l1app used for Fronthaul FPGA and define structures shared between l1 and xRAN lib as shown:

```
struct bbu_xran_io_if {
    void*    nInstanceHandle[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR]; /**< instance per O-
RAN port per CC */
    uint32_t
nBufPoolIndex[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR][MAX_SW_XRAN_INTERFACE_NUM];  /**<
unique buffer pool */
    uint16_t nInstanceNum[XRAN_PORTS_NUM]; /**< instance is equivalent to CC */

    uint16_t DynamicSectionEna;
    uint32_t nPhaseCompFlag;

    int32_t num_o_ru;
    int32_t num_cc_per_port[XRAN_PORTS_NUM];
    int32_t map_cell_id2port[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR];
    struct xran_io_shared_ctrl ioCtrl[XRAN_PORTS_NUM]; /**< for each O-RU port */

    struct xran_cb_tag  RxCbTag[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR];
    struct xran_cb_tag  PrachCbTag[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR];
    struct xran_cb_tag  SrsCbTag[XRAN_PORTS_NUM][XRAN_MAX_SECTOR_NR];
};
struct xran_io_shared_ctrl {
    /* io struct */
    struct xran_io_buf_ctrl
sFrontHaulTxBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR];
    struct xran_io_buf_ctrl
sFrontHaulTxPrbMapBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA
_NR];
    struct xran_io_buf_ctrl
sFrontHaulRxBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR];
    struct xran_io_buf_ctrl
sFrontHaulRxPrbMapBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA
_NR];
    struct xran_io_buf_ctrl
sFHPrachRxBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR];
    /* Cat B */
    struct xran_io_buf_ctrl
sFHSrsRxBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANT_ARRAY_ELM_NR]
;
    struct xran_io_buf_ctrl
sFHSrsRxPrbMapBbuIoBufCtrl[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANT_ARRAY_E
LM_NR];
    /* buffers lists */
    struct xran_flat_buffer
sFrontHaulTxBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR][XRAN_N
UM_OF_SYMBOL_PER_SLOT];
    struct xran_flat_buffer
sFrontHaulTxPrbMapBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR];
    struct xran_flat_buffer
sFrontHaulRxBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR][XRAN_N
UM_OF_SYMBOL_PER_SLOT];
    struct xran_flat_buffer
sFrontHaulRxPrbMapBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR];
    struct xran_flat_buffer
sFHPrachRxBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANTENNA_NR][XRAN_NUM
_OF_SYMBOL_PER_SLOT];
    /* Cat B SRS buffers */
    struct xran_flat_buffer
sFHSrsRxBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANT_ARRAY_ELM_NR][XRAN
_MAX_NUM_OF_SRS_SYMBOL_PER_SLOT];
    struct xran_flat_buffer
sFHSrsRxPrbMapBuffers[XRAN_N_FE_BUF_LEN][XRAN_MAX_SECTOR_NR][XRAN_MAX_ANT_ARRAY_ELM_NR
];
};
```

The Doxygen file and `xran_fh_o_du.h` provides more details on the definition and usage of these structures. Refer to Table 2, for FlexRAN 5G NR Reference Solution RefPHY (Doxygen).

5.4 xRAN Specific Functionality

Front haul interface implementation in the general case is abstracted away using the interface defined in `xran_fh_o_du.h`

The L1 application is not required to access O-RAN protocol primitives (eCPRI header, application header, and others) directly. It is recommended to use the interface to remove dependencies between different software modules such as the l1 pipeline and xRAN library.

5.4.1 External API

The U-plane and C-plane APIs can be used directly from the application if such an option is required. The set of header files can be exported and called directly.

```
xran_fh_o_du.h – xRAN main header file for O-DU scenario
xran_cp_api.h – Control plane functions
xran_pkt_cp.h – xRAN control plane packet definition
xran_pkt.h – xRAN packet definition
xran_pkt_up.h – xRAN User plane packet definition
xran_sync_api.h – api functions to check PTP status
xran_timer.h – API for timing
xran_transport.h – eCPRI transport layer definition and api
xran_up_api.h – user plane functions and definitions
xran_compression.h – interface to compression/decompression functions
```

Source code comments can provide more details on functions and structures available.

5.4.2 C-plane

Implementation of the C-plane set of functions is defined in `xran_cp_api.c` and is used to prepare the content of C-plane packets according to the given configuration. Users can enable/disable generation of C-plane messages using `enableCP` field in struct `xran_fh_init` structure during the initialization of O-RAN front haul. The time of generation of C-plane message for DL and UL is done “Slot-based,” and timing can be controlled using O-DU settings according to Table 4.

The C-plane module contains:

- Initialization of C-plane database to keep track of allocation of resources
- Code to prepare C-plane packet for TX (O-DU)
  - eCPRI header
  - append radio application header
  - append control section header
  - append control section
- Parser of C-plane packet for RX (O-RU emulation)

parses and checks Section 1 and Section 3 packet content

Sending and receiving packets is performed using xRAN ethdi sublayer functions.

More information on function arguments and parameters can be found in Doxygen documents and corresponding source code. Refer to Table 2, for *FlexRAN 5G NR Reference Solution Ref PHY* (Doxygen).

5.4.2.1 Creating a C-Plane Packet

1. API and Data Structures

A C-Plane message can be composed using the following API:

```
int xran_prepare_ctrl_pkt(struct rte_mbuf *mbuf,
    struct xran_cp_gen_params *params,
    uint8_t CC_ID, uint8_t Ant_ID, uint8_t seq_id);
```

`mbuf` is the pointer of a DPDK packet buffer, which is allocated from the caller.

`params` are the pointer of the structure which has the parameters to create the message.

`CC_ID` is the parameter to specify component carrier index, `Ant_ID` is the parameters to specify the antenna port index (RU port index).

`seq_id` is the sequence index for the message.

`params`, the parameters to create a C-Plane message are defined as the structure of `xran_cp_gen_params` with an example given below:

```
struct xran_cp_gen_params {
    uint8_t     dir;
    uint8_t     sectionType;
    uint16_t    numSections;
    struct xran_cp_header_params hdr;
    struct xran_section_gen_info *sections;
};
```

`dir` is the direction of the C-Plane message to be generated. Available parameters are defined as `XRAN_DIR_UL` and `XRAN_DIR_DL`.

`sectionType` is the section type for C-Plane message to generate, as O-RAN specification defines all sections in a C-Plane message shall have the same section type. If different section types are required, they shall be sent with separate C-Plane messages. Available types of sections are defined as `XRAN_CP_SECTIONTYPE_x`. Refer to Table 2, *O-RAN Specification*, Table 5-2 Section Types.

`numSections` is the total number of sections to generate, i.e., the number of the array in sections (struct `xran_section_gen_info`).

`hdr` is the structure to hold the information to generate the radio application and section header in the C-Plane message. It is defined as the structure of `xran_cp_header_params`. Not all parameters in this structure are used for the generation, and the required parameters are slightly different by the type of section, as described in Table 10 and References in the remarks column are corresponding Chapter numbers in the O-RAN *FrontHaul Working Group Control, User, and Synchronization Plane Specification* in Table 2.

Table 10. struct xran_cp_header_params – Common Radio Application Header

	Description	Remarks
`filterIdx`	Filter Index. Available values are defined as `XRAN_FILTERINDEX_xxxxx`.	5.4.4.3
`frameId`	Frame Index. It is modulo 256 of frame number.	5.4.4.4
`subframeId`	Sub-frame Index.	5.4.4.5
`slotId`	Slot Index. The maximum number is 15, as defined in the specification.	5.4.4.6

	Description	Remarks
startSymId	Start Symbol Index.	5.4.4.7

NOTE: References in the remarks column are corresponding Chapter numbers in the O-RAN FrontHaul Working Group Control, User, and Synchronization Plane Specification in Table 2.

Table 11. struct xran_cp_header_params – Section Specific Parameters

	Description	Section Type Applicable						Remarks
		0	1	3	5	6	7	
fftSize	FFT size in frame structure. Available values are defined as XRAN_FFTSIZE_xxxx	X		X				5.4.4.13
Scs	Subcarrier Spacing in the frame structure. Available values are defined as XRAN_SCS_xxxx	X		X				5.4.4.13
iqWidth	I/Q bit width in user data compression header. Should be set by zero for 16bits		X	X	X			5.4.4.10 6.3.3.13
compMeth	Compression Method in user data compression header. Available values are defined as XRAN_COMPMETHOD_xxxx		X	X	X			5.4.4.10 6.3.3.13
numUEs	Number of UEs. Applies to section type 6 and not supported in this release.					X		5.4.4.11
timeOffset	Time Offset. Time offset from the start of the slot to start of Cyclic Prefix.	X		X				5.4.4.12
cpLength	Cyclic Prefix Length.	X		X				5.4.4.14

Note:

1. Only sections types 1 and 3 are supported in the current release.
2. References in the remarks column are corresponding Chapter numbers in the *O-RAN Fronthaul Working Group Control, User, and Synchronization Plane Specification* in Table 2.

Sections are the pointer to the array of structure which has the parameters for section(s) and it is defined as below:

```
struct xran_section_gen_info {
    struct xran_section_info info;
    uint32_t    exDataSize;
    struct {
        uint16_t    type;
        uint16_t    len;
        void        *data;
        } exData[XRAN_MAX_NUM_EXTENSIONS];
    };
```

`info` is the structure to hold the information to generate section and it is defined as the structure of `xran_section_info`. Like `xran_cp_header_params`, all parameters are not required to generate section and Table 12 describes which parameters are required for each section.

Table 12. Parameters for Sections

	Description	Section Type applicable					Remarks
		0	1	3	5	6	
`Id`	Section Identifier.	X	X	X	X	X	5.4.5.1
`Rb`	Resource Block Indicator. Available values are defined as `XRAN_RBIND_xxxx`.	X	X	X	X	X	5.4.5.2
`symInc`	Symbol number Increment command. Available values are defined as `XRAN_SYMBOLNUMBER_xxxx`.	X	X	X	X	X	5.4.5.3
`startPrbc`	Starting PRB of data section description.	X	X	X	X	X	5.4.5.4
`numPrbc`	The number of contiguous PRBs per data section description. When `numPrbc` is greater than 255, it will be converted to zero by the macro (`XRAN_CONVERT_NUMPRBC`).	X	X	X	X	X	5.4.5.6
`reMask`	Resource Element Mask.	X	X	X	X		5.4.5.5
`numSymbol`	Number of Symbols.	X	X	X	X		5.4.5.7
`beamId`	Beam Identifier.		X	X			5.4.5.9
`freqOffset`	Frequency Offset.			X			5.4.5.11
`ueId`	UE identifier. They are not supported in this release.				X	X	5.4.5.10
`regFactor`	Regularization Factor. Not supported in this release					X	5.4.5.12
`Ef`	Extension Flag. They are not supported in this release.		X	X	X	X	5.4.5.8

Note:

1. Only sections types 1 and 3 are supported in the current release.
2. References in the remarks column are corresponding Chapter numbers in the *O-RAN FrontHaul Working Group Control, User, and Synchronization Plane Specification* in Table 2.

NOTE: `xran_section_info` has more parameters – type, `startSymId`, `iqWidth`, `compMeth`. These are the same parameters as those of radio application or section header but need to be copied into this structure again for the section data base.

`exDataSize` and `exData` are used to add section extensions for the section.

`exDataSize` is the number of elements in the `exData` array. The maximum number of elements is defined as `XRAN_MAX_NUM_EXTENSIONS` and it is defined by four in this release with the assumption that four different types of section extensions can be added to a section (section extension type 3 is excluded since it is not supported). `exData.type` is the type of section extension and `exData.len` is the length of structure of section extension parameter in `exData.data`. `exData.data` is the pointer to the structure of section extensions and different structures are used by the type of section extensions like below.

```
struct xran_sectionext1_info {
    uint16_t    rbNumber;    /**< number RBs to ext1 chain */
    uint16_t    bfwNumber;   /**< number of bf weights in this section */
    uint8_t     bfwiqWidth;
    uint8_t     bfwCompMeth;
    int16_t     *p_bfwIQ;    /**< pointer to formed section extention */
    int16_t     bfwIQ_sz;    /**< size of buffer with section extention information
*/
    union {
        uint8_t     exponent;
        uint8_t     blockScaler;
        uint8_t     compBitWidthShift;
        uint8_t     activeBeamspaceCoeffMask[XRAN_MAX_BFW_N];   /* ceil(N/8)*8,
should be multiple of 8 */
        } bfwCompParam;

    };
```

For section extension type 1, the structure of xran_sectionext1_info is used.

NOTE: The xRAN library will use beamforming weight (bfwIQ) as-is, i.e., xRAN library will not perform the compression, so the user should provide proper data to bfwIQ.

```
struct xran_sectionext2_info {
    uint8_t     bfAzPtWidth;
    uint8_t     bfAzPt;
    uint8_t     bfZePtWidth;
    uint8_t     bfZePt;
    uint8_t     bfAz3ddWidth;
    uint8_t     bfAz3dd;
    uint8_t     bfZe3ddWidth;
    uint8_t     bfZe3dd;
    uint8_t     bfAzSI;
    uint8_t     bfZeSI;
    };
```

For section extension type 2, the structure of `xran_sectionext2_info` is used. Each parameter will be packed as specified bit width.

```
struct xran_sectionext3_info {
    uint8_t     codebookIdx;
    uint8_t     layerId;
    uint8_t     numLayers;
    uint8_t     txScheme;
    uint16_t    crsReMask;
    uint8_t     crsShift;
    uint8_t     crsSymNum;
    uint16_t    numAntPort;
    uint16_t    beamIdAP1;
    uint16_t    beamIdAP2;
    uint16_t    beamIdAP3;
    };
```

For section extension type 3, the structure of `xran_sectionext3_info` is used.

```
struct xran_sectionext4_info {
    uint8_t     csf;
    uint8_t     pad0;
    uint16_t    modCompScaler;
    };
```

For section extension type 4, the structure of `xran_sectionext4_info` is used.

```
struct xran_sectionext5_info {
    uint8_t     num_sets;
    struct {
        uint16_t    csf;
        uint16_t    mcScaleReMask;
        uint16_t    mcScaleOffset;
        } mc[XRAN_MAX_MODCOMP_ADDPARMS];
    };
```

For section extension type 5, the structure of `xran_sectionext5_info` is used.

NOTE: Current implementation supports maximum two sets of additional parameters.

```
struct xran_sectionext6_info {
    uint8_t     rbgSize;
    uint8_t     pad;
    uint16_t    symbolMask;
    uint32_t    rbgMask;
    };
For section extension type 6, the structure of xran_sectionext6_info is used.

struct xran_sectionext10_info {
    uint8_t     numPortc;
    uint8_t     beamGrpType;
    uint16_t    beamID[XRAN_MAX_NUMPORTC_EXT10];
    };
```

For section extension type 10, the structure of `xran_sectionext10_info` is used.

```
struct xran_sectionext11_info {
    uint8_t     RAD;
    uint8_t     disableBFWs;


    uint8_t     numBundPrb;
    uint8_t     numSetBFWs;     /* Total number of beam forming weights set (L) */


    uint8_t     bfwCompMeth;
    uint8_t     bfwIqWidth;


    int         totalBfwIQLen;
    int         maxExtBufSize;  /* Maximum space of external buffer */
    uint8_t     *pExtBuf;       /* pointer to start of external buffer */
    void        *pExtBufShinfo; /* Pointer to rte_mbuf_ext_shared_info */
    };
```

For section extension type 11, the structure of `xran_sectionext11_info` is used.

To minimize memory copy for beamforming weights, when section extension 11 is required to send beamforming weights(BFWs), external flat buffer is being used in current release. If extension 11 is used, it will be used instead of `mbufs` that pre-allocated external buffers which BFWs have been prepared already. BFW can be prepared by xran_cp_prepare_ext11_bfws() and the example usage can be found from `app_init_xran_iq_content()` from `sample-app.c`.

5.4.2.1.1 Detail Procedures in API

The `xran_prepare_ctrl_pkt()` has several procedures to compose a C-Plane packet.

1. Append transport header:
   - Reserve eCPRI header space in the packet buffer
   - eCPRI version is fixed by `XRAN_ECPRI_VER (0x0001)`
   - Concatenation and transport layer fragmentation is not supported.
     `ecpri_concat=0, ecpri_seq_id.sub_seq_id=0 and ecpri_seq_id.e_bit=1`
   - The caller needs to provide a component carrier index, antenna index, and message identifier through function arguments.
     `CC_ID, Ant_ID and seq_id`
   - `ecpriRtcid` (`ecpri_xtc_id`) is composed with `CC_ID` and `Ant_ID` by `xran_compose_cid.`
     - DU port ID and band sector ID are fixed by zero in this release.
     - The output of xran_compose_cid is stored in network byte order.
   - The length of the payload is initialized by zero.
2. Append radio application header:
   - The `xran_append_radioapp_header()` checks the type of section through `params->sectionType` and determines proper function to append remaining header components.
     - Only section type 1 and 3 are supported, returns `XRAN_STATUS_INVALID_PARAM` for other types.
     - Each section uses a different function to compose the remaining header and size to calculate the total length in the transport header.
     - For section type 1, `xran_prepare_section1_hdr()` and `sizeof(struct xran_cp_radioapp_section1_header)`
     - For section type 3, `xran_prepare_section3_hdr()` and `sizeof(struct xran_cp_radioapp_section3_header)`
   - Reserves the space of common radio application header and composes header by `xran_prepare_radioapp_common_header()`.

     The header is stored in network byte order.
   - Appends remaining header components by the selected function above

     The header is stored in network byte order
3. Append section header and section:
   - The `xran_append_control_section()` determines proper size and function to append section header and contents.
     - For section type 1, `xran_prepare_section1()` and `sizeof(struct xran_cp_radioapp_section1)`
     - For section type 3, `xran_prepare_section3()` and `sizeof(struct xran_cp_radioapp_section3)`
   - Appends section header and section(s) by selected function above.
     - If multiple sections are configured, then those will be added.
     - Since fragmentation is not considered in this implementation, the total length of a single C-Plane message shall not exceed MTU size.

- The header and section(s) are stored in network byte order.
  - Appends section extensions if it is set (`ef=1`)
  - The xran_append_section_extensions() adds all configured extensions by its type.
  - The `xran_prepare_sectionext_x()` (`x = 1,2,4,5`) will be called by the type from and these functions will create extension field.

5.4.2.1.2 Example Usage of API

There are two reference usages of API to generate C-Plane messages:

- xran_cp_create_and_send_section() in xran_main.c
- generate_cpmsg_prach() in xran_common.c

The `xran_cp_create_and_send_section()` is to generate the C-Plane message with section type 1 for DL or UL symbol data scheduling.

This function has hardcoded values for some parameters such as:

- The filter index is fixed to `XRAN_FILTERINDEX_STANDARD`.
- RB indicator is fixed to `XRAN_RBIND_EVERY`.
- Symbol increment is not used (`XRAN_SYMBOLNUMBER_NOTINC`)
- Resource Element Mask is fixed to 0xfff

If section extensions include extension 1 or 11, direct `mbuf` will not be allocated/used and pre-allocated flat buffer will be attached to indirect `mbuf.` This external buffer will be used to compose C-Plane message and should have BFWs already by `xran_cp_populate_section_ext_1() or xran_cp_prepare_ext11_bfws()`.

Since current implementation uses single section single C-Plane message, if multi sections are present, this function will generate same amount of C-Plane messages with the number of sections.

After C-Plane message generation, it will send generated packet to TX ring after adding an Ethernet header and also will add section information of generated C-Plane packet to section database, to generate U-plane message by C-Plane configuration.

The `generate_cpmsg_prach()` is to generate the C-Plane message with section type 3 for PRACH scheduling.

This functions also has some hardcoded values for the following parameters:

- RB indicator is fixed to `XRAN_RBIND_EVERY`.
- Symbol increment is not used (`XRAN_SYMBOLNUMBER_NOTINC`).
- Resource Element Mask is fixed to `0xfff`.

This function does not send generated packet, `send_cpmsg()` should be called after this function call. The example can be found from `tx_cp_ul_cb()` in `xran_main.c`. Checking and parsing received PRACH symbol data by section information from the C-Plane are not implemented in this release.

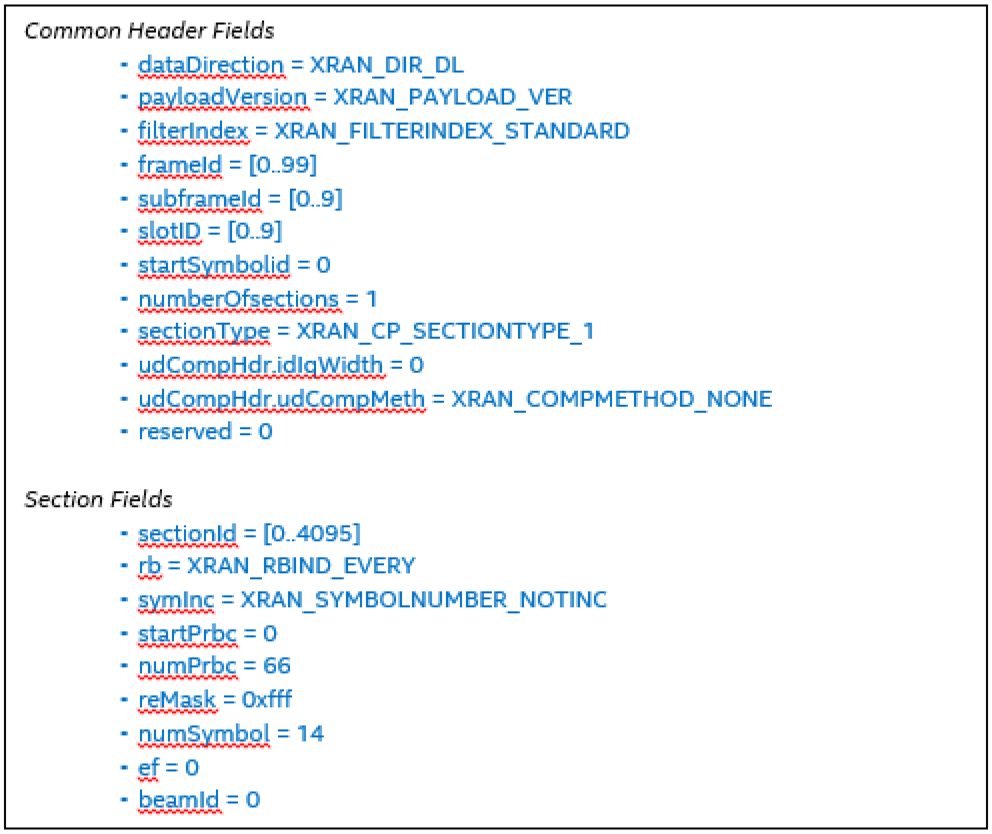
5.4.2.1.3 Example Configuration of C-Plane Messages

C-Plane messages can be composed through the API, and the sample application shows several reference usages of the configuration for different numerologies.

Below are the examples of the C-Plane message configuration with a sample application for `mmWave` – numerology 3, 100 MHz bandwidth, TDD (DDDS)

C-Plane Message – downlink symbol data for a downlink slot

- Single CP message with the single section of section type 1
- Configures single CP message for all consecutive downlink symbols
- Configures whole RBs (66) for a symbol
- Compression and beamforming are not used



Common Header Fields

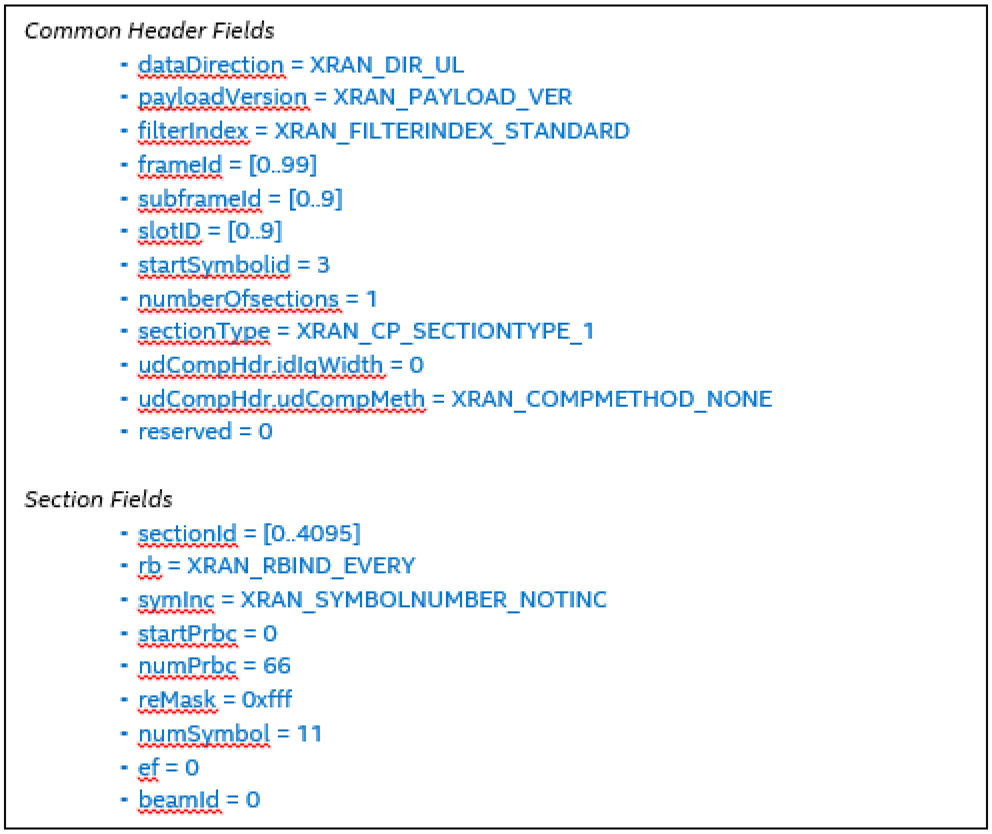
- dataDirection = XRAN_DIR_DL
- payloadVersion = XRAN_PAYLOAD_VER
- filterIndex = XRAN_FILTERINDEX_STANDARD
- frameId = [0..99]
- subframeId = [0..9]
- slotID = [0..9]
- startSymbolid = 0
- numberOfsections = 1
- sectionType = XRAN_CP_SECTIONTYPE_1
- udCompHdr.idIqWidth = 0
- udCompHdr.udCompMeth = XRAN_COMPMETHOD_NONE
- reserved = 0

Section Fields

- sectionId = [0..4095]
- rb = XRAN_RBIND_EVERY
- symInc = XRAN_SYMBOLNUMBER_NOTINC
- startPrbc = 0
- numPrbc = 66
- reMask = 0xfff
- numSymbol = 14
- ef = 0
- beamId = 0

C-Plane Message – uplink symbol data for uplink slot

- Single CP message with the single section of section type 1
- Configures single CP message for all consecutive uplink symbols (UL symbol starts from 3)
- Configures whole RBs (66) for a symbol
- Compression and beamforming are not used



Common Header Fields

- dataDirection = XRAN_DIR_UL
- payloadVersion = XRAN_PAYLOAD_VER
- filterIndex = XRAN_FILTERINDEX_STANDARD
- frameId = [0..99]
- subframeId = [0..9]
- slotID = [0..9]
- startSymbolid = 3
- numberOfsections = 1
- sectionType = XRAN_CP_SECTIONTYPE_1
- udCompHdr.idIqWidth = 0
- udCompHdr.udCompMeth = XRAN_COMPMETHOD_NONE
- reserved = 0

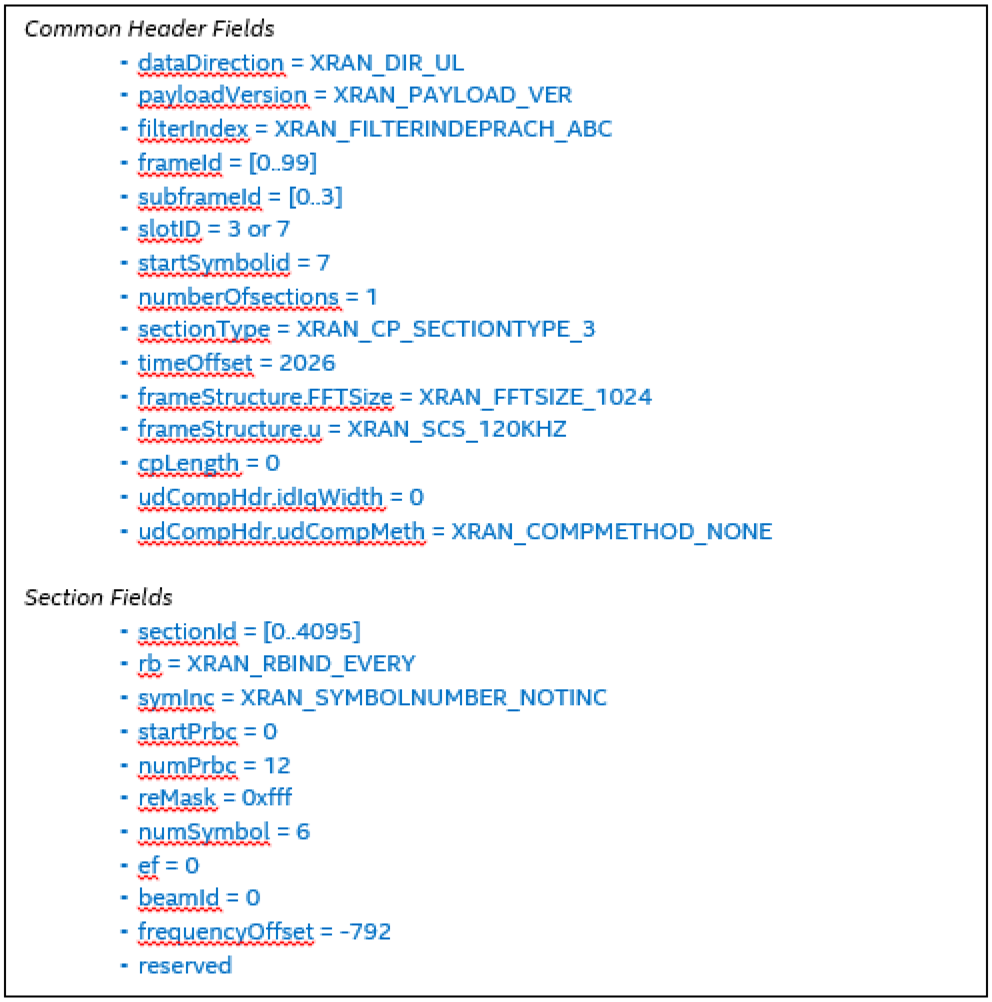
Section Fields

- sectionId = [0..4095]
- rb = XRAN_RBIND_EVERY
- symInc = XRAN_SYMBOLNUMBER_NOTINC
- startPrbc = 0
- numPrbc = 66
- reMask = 0xfff
- numSymbol = 11
- ef = 0
- beamId = 0

C-Plane Message – PRACH

- Single CP message with the single section of section type 3 including repetition
- Configures PRACH format A3, config index 81, and detail parameters are:
  - `Filter Index : 3`
  - `CP length : 0`
  - `Time offset : 2026`
  - `FFT size : 1024`
  - `Subcarrier spacing : 120KHz`
  - `Start symbol index : 7`
  - `Number of symbols : 6`
  - `Number of PRBCs : 12`
  - `Frequency offset : -792`

NOTE: Compression and beamforming are not used.



Common Header Fields

- dataDirection = XRAN_DIR_UL
- payloadVersion = XRAN_PAYLOAD_VER
- filterIndex = XRAN_FILTERINDEPRACH_ABC
- frameId = [0..99]
- subframeId = [0..3]
- slotID = 3 or 7
- startSymbolid = 7
- numberOfsections = 1
- sectionType = XRAN_CP_SECTIONTYPE_3
- timeOffset = 2026
- frameStructure.FFTSize = XRAN_FFTSIZE_1024
- frameStructure.u = XRAN_SCS_120KHZ
- cpLength = 0
- udCompHdr.idIqWidth = 0
- udCompHdr.udCompMeth = XRAN_COMPMETHOD_NONE

Section Fields

- sectionId = [0..4095]
- rb = XRAN_RBIND_EVERY
- symInc = XRAN_SYMBOLNUMBER_NOTINC
- startPrbc = 0
- numPrbc = 12
- reMask = 0xfff
- numSymbol = 6
- ef = 0
- beamId = 0
- frequencyOffset = -792
- reserved

5.4.2.2 Functions to Store/Retrieve Section Information

There are several functions to store/retrieve section information of C-Plane messages. Since U-plane messages must be generated by the information in the sections of a C-Plane message, it is required to store and retrieve section information.

5.4.2.2.1 APIs and Data Structure

APIs for initialization and release storage are:

- `int xran_cp_init_sectiondb(void *pHandle);`
- `int xran_cp_free_sectiondb(void *pHandle);`

APIs to store and retrieve section information are:

- `int xran_cp_add_section_info(void *pHandle, uint8_t dir, uint8_t cc_id, uint8_t ruport_id, uint8_t ctx_id, struct xran_section_info *info);`
- `int xran_cp_add_multisection_info(void *pHandle, uint8_t cc_id, uint8_t ruport_id, uint8_t ctx_id, struct xran_cp_gen_params *gen_info);`

- `struct xran_section_info *xran_cp_find_section_info(void *pHandle, uint8_t dir, uint8_t cc_id, uint8_t ruport_id, uint8_t ctx_id, uint16_t section_id);`
- `struct xran_section_info *xran_cp_iterate_section_info(void *pHandle, uint8_t dir, uint8_t cc_id, uint8_t ruport_id, uint8_t ctx_id, uint32_t *next);`
- `int xran_cp_getsize_section_info(void *pHandle, uint8_t dir, uint8_t cc_id, uint8_t ruport_id, uint8_t ctx_id);`

APIs to reset the storage for a new slot are:

```
int xran_cp_reset_section_info(void *pHandle, uint8_t dir, uint8_t cc_id, uint8_t
ruport_id, uint8_t ctx_id);
```

The structure of `xran_section_info` is used to store/retrieve information. This is the same structure used to generate a C-Plane message. Refer to Section 1, API and Data Structures for more details.

The storage for section information is declared as a multi-dimensional array and declared as a local static variable to limit direct access. Each item is defined as the structure of `xran_sectioninfo_db`, and it has the number of stored section information items (`cur_index`) and the array of the information (list), as shown below.

```
/**
 * This structure to store the section information of C-Plane
 * in order to generate and parse corresponding U-Plane */
struct xran_sectioninfo_db {
    uint32_t    cur_index;  /**< Current index to store for this eAXC */
    struct xran_section_info list[XRAN_MAX_NUM_SECTIONS]; /**< The array of section
information */
};
static struct xran_sectioninfo_db
sectiondb[XRAN_MAX_SECTIONDB_CTX][XRAN_DIR_MAX][XRAN_COMPONENT_CARRIERS_MAX][XRAN_MAX_
ANTENNA_NR*2 + XRAN_MAX_ANT_ARRAY_ELM_NR];
```

The maximum size of the array can be adjusted if required by system configuration. Since transmission and reception window of U-Plane can be overlapped with the start of new C-Plane for next slot, functions have context index to identify and protect the information. Currently the maximum number of context is defined by two and it can be adjusted if needed.

NOTE: Since the context index is not managed by the library and APIs are expecting it from the caller as a parameter, the caller shall consider a proper method to manage it to avoid corruption. The current reference implementation uses a slot and subframe index to calculate the context index.

5.4.2.2.2 Example Usage of APIs

There are references to show the usage of APIs as below.

- Initialization and release:

```
-    xran_cp_init_sectiondb(): xran_open() in lib/src/xran_main.c
-    xran_cp_free_sectiondb(): xran_close() in lib/src/xran_main.c
```

- Store section information:

```
-    xran_cp_add_section_info(): xran_cp_create_and_send_section() in xran_main.c
and send_cpmsg ()in lib/src/xran_common.c
```

- Retrieve section information:

```
-    xran_cp_iterate_section_info(): xran_process_tx_sym() in lib/src/xran_main.c
-    xran_cp_getsize_section_info(): xran_process_tx_sym() in lib/src/xran_main.c
```

- Reset the storage for a new slot:

```
-   xran_cp_reset_section_info(): tx_cp_dl_cb() and tx_cp_ul_cb() in
lib/src/xran_main.c
```

5.4.2.2.3 Function for RU emulation and Debug

`xran_parse_cp_pkt()` is a function which can be utilized for RU emulation or debug. It is defined below:

```
int xran_parse_cp_pkt(struct rte_mbuf *mbuf,
    struct xran_cp_recv_params *result,
    struct xran_recv_packet_info *pkt_info);
```

It parses a received C-Plane packet and retrieves the information from its headers and sections.

The retrieved information is stored in the structures:

- `struct xran_cp_recv_params:` section information from received C-Plane packet
- `struct xran_recv_packet_info:` transport layer header information (eCPRI header)

These functions can be utilized to debug or RU emulation purposes.

5.4.3 U-plane

Single Section is the default mode of xRAN packet creation. It assumes that there is only one section per packet, and all IQ samples are attached to it. Compression is not supported.

A message is built in the `mbuf` space given as a parameter. The library builds eCPRI header filling structure fields by taking the IQ sample size and populating a particular packet length and sequence number.

With block floating point compression, supported IQ bit widths are 8,9,10,12,14. With modulation compression, supported IQ bit widths are defined according to modulation order as in section A.5 of O-RAN spec..

Implementation of a U-plane set of functions is defined in `xran_up_api.c` and is used to prepare U-plane packet content according to the given configuration.

The following list of functions is implemented for U-plane:

- Build eCPRI header
- Build application header
- Build section header
- Append IQ samples to packet
- Prepare full symbol of O-RAN data for single `eAxC`
- Process RX packet per symbol.

The time of generation of a U-plane message for DL and UL is “symbol-based” and can be controlled using O-DU settings (O-RU), according to Table 4.

For more information on function arguments and parameters refer to Table 2, *Doxygen and corresponding source code*.

5.5 Supporting Code

The xRAN library has a set of functions used to assist in packet processing and data exchange not directly used for O-RAN packet processing.

5.5.1 Timing

The sense of time for the O-RAN protocol is obtained from system time, where the system timer is synchronized to GPS time via PTP protocol using the Linux PHP package. On the software side, a simple polling loop is utilized to get time up to nanosecond precision and particular packet processing jobs are scheduled via the DPDK timer.

```
long poll_next_tick(int interval)
{
        struct timespec start_time;
        struct timespec cur_time;
        long target_time;
        long delta;
        clock_gettime(CLOCK_REALTIME, &start_time);
        target_time = (start_time.tv_sec * NSEC_PER_SEC + start_time.tv_nsec +
interval * NSEC_PER_USEC) / (interval *              NSEC_PER_USEC) * interval;
        while(1)
        {
                 clock_gettime(CLOCK_REALTIME, &cur_time);
                 delta = (cur_time.tv_sec * NSEC_PER_SEC + cur_time.tv_nsec) -
target_time * NSEC_PER_USEC;
                 if(delta > 0 || (delta < 0 && abs(delta) < THRESHOLD))
                 {
                     break;
                 }
        }
        return delta;
}
```

Polling is used to achieve the required precision of symbol time. For example, in the mmWave scenario, the symbol time is 125 µs/14=~8.9 µs. Small deterministic tasks can be executed within the polling interval provided. It's smaller than the symbol interval time.

Current O-RAN library supports multiple O-RU of multiple numerologies, thus the sense of timing is based on the O-RU with highest numerology (smallest symbol time). It is required to configure the O-RU0 with highest numerology in the O-RAN configuration.

5.5.2 DPDK Timers

DPDK provides sets of primitives (`struct rte_rimer`) and functions (`rte_timer_reset_sync()` `rte_timer_manage()`) to schedule processing of function as timer. The timer is based on the TSC clock and is not synchronized to PTP time. As a result, this timer cannot be used as a periodic timer because the TSC clock can drift substantially relative to the system timer which in turn is synchronized to PTP (GPS)

Only single-shot timers are used to schedule processing based on events such as symbol time. The packet processing function calls `rte_timer_manage()` in the loop, and the resulting execution of timer function happens right after the timer was "armed".

5.5.3 xRAN Ethernet

The `xran_init_port()` function performs initialization of DPDK ETH port. Standard port configuration is used as per reference example from DPDK.

Jumbo Frames are used by default. `mbufs` size is extended to support 9600 bytes packets.

Configurable MTU size is supported starting from 20.11 release.

MAC address and VLAN tag are expected to be configured by Infrastructure software. Refer to A.4, Install and Configure Sample Application.

From an implementation perspective, modules provide functions to handle:

- Ethernet headers
- VLAN tag
- Send and Receive `mbuf`.

5.5.4 xRAN Ethdi

`Ethdi` provides functionality to work with the content of an Ethernet packet and dispatch processing to/from the xRAN layer. `Ethdi` instantiates a main PMD driver thread and dispatches packets between the ring and RX/TX using `rte_eth_rx_burst()` and `rte_eth_tx_burst()` DPDK functions.

For received packets, it maintains a set of handlers for `ethertype` handlers and xRAN layer register one O-RAN `ethtype 0xAEFE`, resulting in packets with this `ethertype` being routed to the xRAN processing function. This function checks the message type of the eCPRI header and dispatches packet to either C-plane processing or U-plane processing.

Initialization of memory pools, allocation, and freeing of the `mbuf` for Ethernet packets occur in this layer.

Refer to Table 2 for detailed information on functions implemented for xRAN `ethdi` can be found in the Doxygen provided with the FlexRAN release.

5.5.5 O-RAN One Way Delay Measurements

The support for the eCPRI one- way delay measurements which are specified by the O-RAN to be used with the Measured Transport support per Section 2.3.3.3 of the O-RAN-WG4.CUS.0-v4.00 specification and section 3.2.4.6 of the eCPRI_v2.0 specification is implemented in the file xran_delay_measurement.c. Structure definitions used by the owd measurement functions are in the file xran_fh_o_du.h for common data and port specific variables and parameters.

The implementation of this feature has been done under the assumption that the requestor is the O-DU and the recipient is the O-RU. All of the action_types per the eCPRI 2.0 have been implemented. In the current version the timestamps are obtained using the linux function clock_gettime using CLOCK_REALTIME as the clock_id argument.

The implementation supports both the O-RU and the O-DU side in order to do the unit test in loopback mode.

The one-delay measurements are enabled at configuration time and run right after the xran_start() function is executed. The total number of consecutive measurements per port should be a power of 2 and in order to minimize the system startup it is advisable that the number is 16 or below.

The following functions can be found in the xran_delay_measurement.c:

xran_ecpri_one_way_delay_measurement_transmitter() which is invoked from the process_dpdk_io() function if the one-way delay measurements are enabled. This is the main function for the owd transmitter.

xran_generate_delay_meas() is a general function used by the transmitter to send the appropriate messages based on actionType and filling up all the details for the ethernet and ecpri layers.

Process_delay_meas() this function is invoked from the handle_ecpri_ethertype() function when the ecpri message type is ECPRI_DELAY_MEASUREMENT. This is the main owd receiver function.

From the Process_delay_meas() and depending on the message received we can execute one of the following functions

xran_process_delmeas_request() If we received a request message.

xran_process_delmeas_request_w_fup() If we received a request with follow up message.

xran_process_delmeas_response() If we received a response message.

xran_process_delmeas_rem_request() If we received a remote request message

xran_delmeas_rem_request_w_fup() If we received a remote request with follow up message.

All of the receiver functions also can generate the appropriate send message by using the DPDK function rte_eth_tx_burst() to minimize the response delay.

Additional utility functions used by the owd implementation for managing of timestamps and time measurements are:

xran_ptp_ts_to_ns() that takes a TimeStamp argument from a received owd ecpri packet and places it in host order and returns the value in nanoseconds.

xran_timespec_to_ns() that takes an argument in timespec format like the return value from the linux function clock_gettime() and returns a value in nanoseconds.

xran_ns_to_timespec() that takes an argument in nanoseconds and returns a value by reference in timespec format.

xran_compute_and_report_delay_estimate() This function takes an average of the computed one way delay measurements and prints out the average value to the console expressed in nanoseconds. Currently we exclude the first 2 measurements from the average.

Utility functions in support of the owd ecpri packet formulation are:

xran_build_owd_meas_ecpri_hdr() Builds the ecpri header with message type ECPRI_DELAY_MEASUREMENT and writes the payload size in network order.

xran_add_at_and_measId_to_header() This function is used to write the action Type and MeasurementID to the eCPRI owd header.

The current implementation of the one way delay measurements only supports a fixed message size. The message is defined in the xran_pkt.h in the structure xran_ecpri_delay_meas_pl.

The one-way delay measurements have been tested with the sample-app for the Front Haul Interface Library and have not yet been integrated with the L1 Layer functions.

5.5.6 Mlog Usage

`Mlog` is a timing information log that contains information about task execution within the application. It is extensively used for the 5G NR L1 pipeline. By default, the xRAN library supports the gathering of `mlog` data for critical functions in the code to be able to debug timing-related issues for latency between different events in the code. The file `xran_lib_mlog_tasks_id.h` contains the IDs of tasks used within the xRAN library. The `mlog` timing information can be gathered along with the rest of the L1 application. The library expects that `mlog` is opened before the library is initialized for the first time `xran_init()`.

There are two compilation options:

- In `xran/lib/Makefile`: `MLOG_ENABLED` – enables usage of MLOG (enabled by default)
- In `libs/mlog/source/makecfg: MLOG_SYS_CLOCK_TICK` – enables usage of system clock for capturing of timing information instead of TSC (disabled by default as result TSC is used).

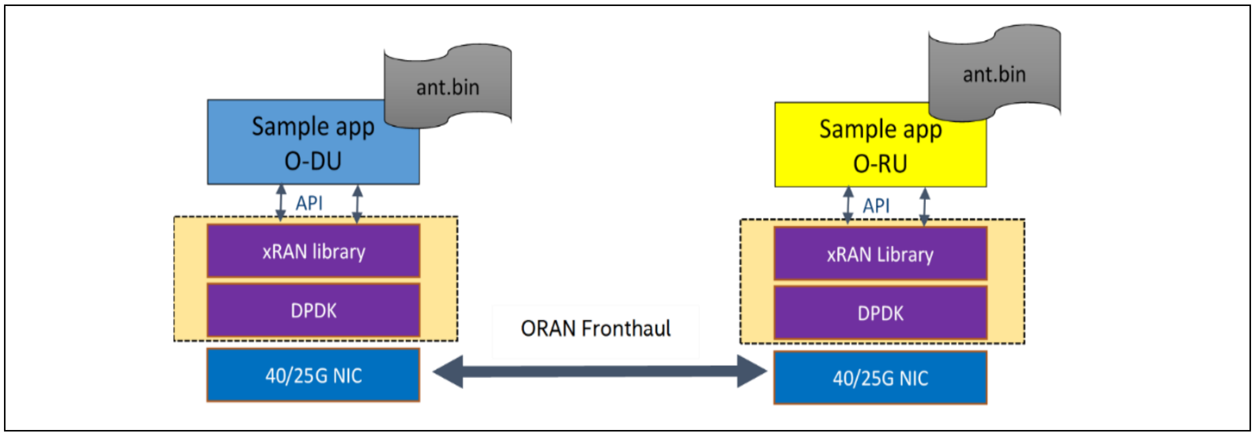
  The benefit of the second option is the possibility to capture mlog file data from two different systems (O-DU server and simulation of RU on a different server) and analyze information together as baseline “sense of time” is the same and synchronized to GPS via PTP.

§

6.0 Sample Application

Figure 26 illustrates a sample xRAN application.

Figure 26. Sample Application



The sample application was created to execute test scenarios with features of the xRAN library and test external API as well as timing. The sample application is named sample-app, and depending on configuration file settings can act as O-DU or simplified simulation of O-RU. The first O-DU should be run on the machine that acts as O-DU and the second as O-RU. Both machines are connected via ETH. The sample application on both sides executes using a constant configuration according to settings in corresponding config files (`./app/usecase/mu0_10mhz/config_file_o_du.dat` and `./app/usecase/mu0_10mhz/config_file_o_ru.dat`) and uses binary files (`ant.bin`) with IQ samples as input. Multiple-use cases for different numerologies and different BW are available as examples. Configuration files provide descriptions of each parameter, and in general, those are related to M-plane level settings as per the O-RAN Fronthaul specification, refer to Table 2.

From the start of the process, the application (O-DU) sends DL packets for the U-plane and C-plane and receives U-plane UL packets. Synchronization of O-DU and O-RU sides is achieved via IEEE 1588.

U-plane packets for UL and DL direction are constructed the same way except for the direction field.

Examples of default configurations used with the sample application for v20.04 release provided below:

1 Cell mmWave 100MHz TDD DDDS:

- Numerology 3 (mmWave)
- TTI period 125 µs
- 100 MHz Bandwidth: 792 subcarriers (all 66 RB utilized at all times)
- 4x4 MIMO
- No beamforming
- 1 Component carrier
- Jumbo Frame for Ethernet (up to 9728 bytes)
- Front haul throughput ~11.5 Gbps.

12 Cells Sub6 10MHz FDD:

- Numerology 0 (Sub-6)
- TTI period 1000 µs
- 10 MHz Bandwidth: 624 subcarriers (all 52 RB utilized at all times)
- 4x4 MIMO
- No beamforming
- 12 Component carrier
- Jumbo Frame for Ethernet (up to 9728 bytes)
- Front haul throughput ~13.7Gbps.

1 Cell Sub6 100 MHz TDD

- Numerology 1 (Sub-6)
- TTI period 500 µs
- 100 MHz Bandwidth: 3276 subcarriers (all 273 RB utilized at all times)
- 4x4 MIMO
- No beamforming
- 1 Component carrier
- Jumbo Frame for Ethernet (up to 9728 bytes)
- Front haul throughput ~11.7 Gbps.

1 Cell Sub6 100 MHz TDD (Category B):

- Numerology 1 (Sub-6)
- TTI period 500 µs
- 100 MHz Bandwidth: 3276 subcarriers (all 273 RB utilized at all times). 8 UEs per TTI per layer
- 8DL /4UL MIMO Layers
- Digital beamforming with 32T32R
- 1 Component carrier
- Jumbo Frame for Ethernet (up to 9728 bytes)
- Front haul throughput ~23.5 Gbps.

3 Cell Sub6 100MHz TDD Massive MIMO (Category B):

- Numerology 1 (Sub-6)
- TTI period 500 µs
- 100 Mhz Bandwidth: 3276 subcarriers (all 273 RB utilized at all times). 8 UEs per TTI per layer
- 16DL /8UL MIMO Layers
- Digital beamforming with 64T64R
- 1 Component carrier for each Cell
- Jumbo Frame for Ethernet (up to 9728 bytes)

- Front haul throughput ~44 Gbps.

Other configurations can be constructed by modifying the config files (see `app/usecase/`)

One_way Delay Measurements:

There are 4 usecases defined that are based on cat a, numerology 0 and 20 MHz Bw:

Common to all cases the following parameters are needed in the usecase_xu.cfg files where x=r for ORU and x=d for ODU.

```
oXuOwdmNumSamps=8   # Run 8 samples per port
oXuOwdmFltrType=0   # Simple average
oXuOwdmRespTimeOut=10000000 # 10 ms expressed in ns (Currently not enforced)
oXuOwdmMeasState=0  # Measurement state is INIT
oXuOwdmMeasId=0     # Measurement Id seed
oXuOwdmEnabled=1    # Measurements are enabled
oXuOwdmPlLength= n  # with 40 <= n <= 1400 bytes
```

For the ORU

oXuOwdmInitEn=0 #O-RU is always the recipient

For the ODU

oXuOwdmInitEn=1   #O-DU is always initiator

20 Corresponds to the Request/Response usecase with Payload Size 40 bytes

oXuOwdmMeasMeth=0 # Measurement Method REQUEST

21 Corresponds to the Remote Request usecase with Payload Size 512 bytes

oXuOwdmMeasMeth=1 # Measurement Method REM_REQ

22 Corresponds to the Request with Follow Up usecase with Payload Size 1024 bytes

oXuOwdmMeasMeth=2 # Measurement Method REQUESTwFUP

23 Corresponds to the Remote Request with Follow Up usecase with default Payload Size

oXuOwdmMeasMeth=3 # Measurement Method REM_REQ_WFUP

§

Appendix A Setup Configuration

A.1 Setup Configuration

The configuration shown in Figure 26 shows how to set up a test environment to execute O-RAN scenarios where O-DU and 0-RU are simulated using the sample application provided with the FlexRAN release package. This setup allows development and prototyping as well as testing of O-RAN specific functionality. The O-DU side can be instantiated with a full 5G NR L1 reference as well. The configuration differences of the 5G NR l1app configuration are provided below. Steps for running the sample application on the O-DU side and O-RU side are the same, except configuration file options may be different.

Figure 27. Setup for O-RAN Testing

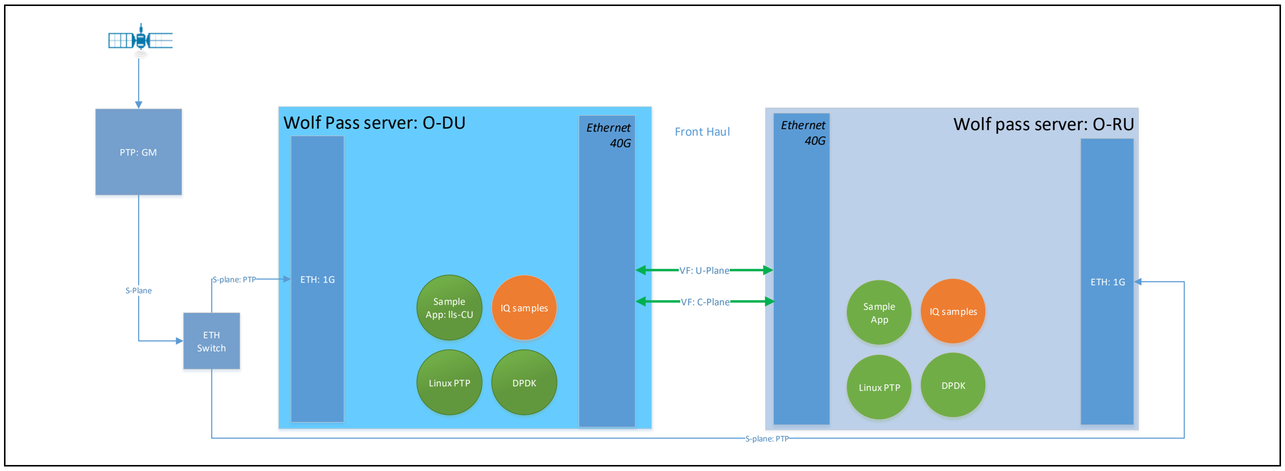


Figure 28. Setup for O-RAN Testing with PHY and Configuration C3

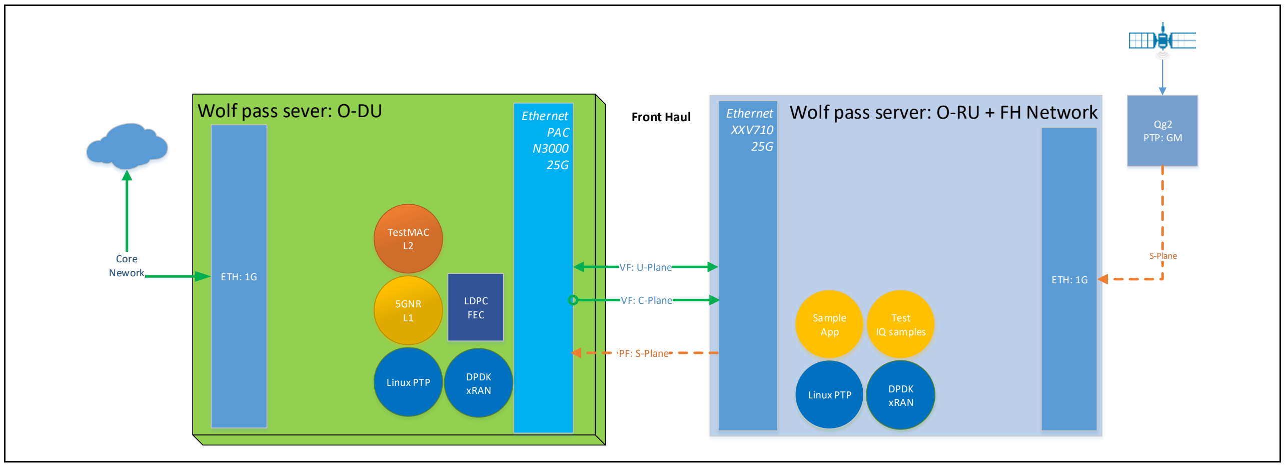
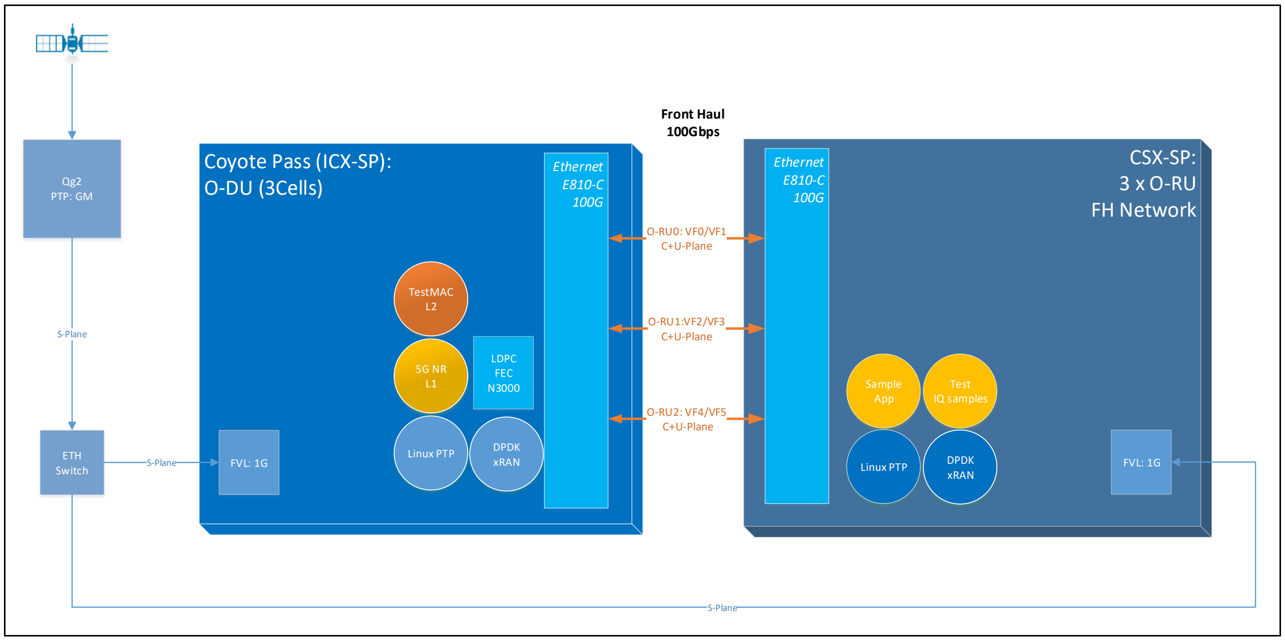


Figure 29. Setup for O-RAN Testing with PHY and Configuration C3 for Massive MIMO



A.2 Prerequisites

Each server in Figure 27 requires the following:

- Wolfpass server according to recommended BOM for FlexRAN such as Intel® Xeon® Skylake Gold 6148 FC-LGA3647 2.4 GHz 27.5 MB 150W 20 cores (two sockets) or higher
- Wilson City or Coyotee Pass server with Intel® Xeon® Icelake CPU for Massive-MIMO with L1 pipeline testing
- BIOS settings:
  - Intel® Virtualization Technology Enabled
  - Intel® VT for Directed I/O - Enabled
  - ACS Control - Enabled
  - Coherency Support - Disabled
- Front Haul networking cards:
  - Intel® Ethernet Converged Network Adapter XL710-QDA2
  - Intel® Ethernet Converged Network Adapter XXV710-DA2
  - Intel® Ethernet Converged Network Adapter E810-CQDA2
  - Intel® FPGA Programmable Acceleration Card (Intel® FPGA PAC) N3000
- Back (Mid) Haul networking card can be either:
  - Intel® Ethernet Connection X722 for 10GBASE-T
  - Intel® 82599ES 10-Gigabit SFI/SFP+ Network Connection
  - Other networking cards capable of HW timestamping for PTP synchronization.

NOTE: Both Back (mid) Haul and Front Haul NIC require support for PTP HW timestamping.

The recommended configuration for NICs is:

```
ethtool -i enp33s0f0
driver: i40e
version: 2.14.13
firmware-version: 8.20 0x80009bd4 1.2879.0
expansion-rom-version:
bus-info: 0000:21:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
ethtool -T enp33s0f0
Time stamping parameters for enp33s0f0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 4
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        ptpv1-l4-sync         (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
        ptpv1-l4-delay-req    (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
        ptpv2-l4-event        (HWTSTAMP_FILTER_PTP_V2_L4_EVENT)
        ptpv2-l4-sync         (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
        ptpv2-l4-delay-req    (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
        ptpv2-l2-event        (HWTSTAMP_FILTER_PTP_V2_L2_EVENT)
        ptpv2-l2-sync         (HWTSTAMP_FILTER_PTP_V2_L2_SYNC)
        ptpv2-l2-delay-req    (HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ)
        ptpv2-event           (HWTSTAMP_FILTER_PTP_V2_EVENT)
        ptpv2-sync            (HWTSTAMP_FILTER_PTP_V2_SYNC)
        ptpv2-delay-req       (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
```

The recommended configuration for Columbiaville NICs (base on Intel® Ethernet 800 Series (Columbiaville) CVL 2.3 release is:

```
ethtool -i enp81s0f0
driver: ice
version: 1.3.2
firmware-version: 2.3 0x80005D18
expansion-rom-version:
bus-info: 0000:51:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
ethtool -T enp81s0f0
Time stamping parameters for enp81s0f0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 1
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)

Recommended version of
iavf driver 4.0.2
ICE COMMS Package version 1.3.24.0
```

Note: If your firmware version does not match with the ones in the output images, you can download the correct version from the Intel Download Center. It is Intel's repository for the latest software and drivers for Intel products. The NVM Update Packages for Windows*, Linux*, ESX*, FreeBSD*, and EFI/EFI2 are located at:

https://downloadcenter.intel.com/download/24769 (700 series)

https://downloadcenter.intel.com/download/29736 (E810 series)

PTP Grand Master is required to be available in the network to provide synchronization of both O-DU and RU to GPS time.

The software package includes Linux* CentOS* operating system and RT patch according to FlexRAN Reference Solution Cloud-Native Setup document (refer to Table 2). Only real-time HOST is required.

1. Install Intel® C++ Compiler v19.0.3
2. Download DPDK v20.11
3. Patch DPDK with FlexRAN BBDev patch as per given release.
4. Double check that FlexRAN DPDK patch includes changes below relevant to O-RAN Front haul:

```
For Fortville:
diff --git a/drivers/net/i40e/i40e_ethdev.c b/drivers/net/i40e/i40e_ethdev.c
index 85a6a86..236fbe0 100644
--- a/drivers/net/i40e/i40e_ethdev.c
+++ b/drivers/net/i40e/i40e_ethdev.c
@@ -2207,7 +2207,7 @@ void i40e_flex_payload_reg_set_default(struct i40e_hw *hw)
     /* Map queues with MSIX interrupt */
     main_vsi->nb_used_qps = dev->data->nb_rx_queues -
              pf->nb_cfg_vmdq_vsi * RTE_LIBRTE_I40E_QUEUE_NUM_PER_VM;
-    i40e_vsi_queues_bind_intr(main_vsi, I40E_ITR_INDEX_DEFAULT);
+    i40e_vsi_queues_bind_intr(main_vsi, I40E_ITR_INDEX_NONE);
     i40e_vsi_enable_queues_intr(main_vsi);

     /* Map VMDQ VSI queues with MSIX interrupt */
@@ -2218,6 +2218,10 @@ void i40e_flex_payload_reg_set_default(struct i40e_hw *hw)
              i40e_vsi_enable_queues_intr(pf->vmdq[i].vsi);
     }
+    i40e_aq_debug_write_global_register(hw,
+                                             0x0012A504,
+                                             0, NULL);
+
     /* enable FDIR MSIX interrupt */
     if (pf->fdir.fdir_vsi) {
              i40e_vsi_queues_bind_intr(pf->fdir.fdir_vsi,
diff --git a/drivers/net/i40e/i40e_ethdev_vf.c b/drivers/net/i40e/i40e_ethdev_vf.c
index 001c301..6f9ffdb 100644
```

```
--- a/drivers/net/i40e/i40e_ethdev_vf.c
+++ b/drivers/net/i40e/i40e_ethdev_vf.c
@@ -640,7 +640,7 @@ struct rte_i40evf_xstats_name_off {

     map_info = (struct virtchnl_irq_map_info *)cmd_buffer;
     map_info->num_vectors = 1;
-    map_info->vecmap[0].rxitr_idx = I40E_ITR_INDEX_DEFAULT;
+    map_info->vecmap[0].rxitr_idx = I40E_ITR_INDEX_NONE;
     map_info->vecmap[0].vsi_id = vf->vsi_res->vsi_id;
     /* Alway use default dynamic MSIX interrupt */
     map_info->vecmap[0].vector_id = vector_id;
diff --git a/drivers/net/ixgbe/ixgbe_ethdev.c b/drivers/net/ixgbe/ixgbe_ethdev.c
index 26b1927..018eb8f 100644
--- a/drivers/net/ixgbe/ixgbe_ethdev.c
+++ b/drivers/net/ixgbe/ixgbe_ethdev.c
@@ -3705,7 +3705,7 @@ static int ixgbevf_dev_xstats_get_names(__rte_unused struct
rte_eth_dev *dev,
               * except for 82598EB, which remains constant.
               */
              if (dev_conf->txmode.mq_mode == ETH_MQ_TX_NONE &&
-                              hw->mac.type != ixgbe_mac_82598EB)
+                              hw->mac.type != ixgbe_mac_82598EB && hw->mac.type !=
ixgbe_mac_82599EB)
                         dev_info->max_tx_queues = IXGBE_NONE_MODE_TX_NB_QUEUES;
     }
     dev_info->min_rx_bufsize = 1024; /* cf BSIZEPACKET in SRRCTL register */
diff --git a/lib/librte_eal/common/include/rte_dev.h
b/lib/librte_eal/common/include/rte_dev.h
old mode 100644
new mode 100755

for Columbiaville
diff --git a/drivers/net/ice/ice_ethdev.c b/drivers/net/ice/ice_ethdev.c
index de189daba..d9aff341c 100644
--- a/drivers/net/ice/ice_ethdev.c
+++ b/drivers/net/ice/ice_ethdev.c
@@ -2604,8 +2604,13 @@ __vsi_queues_bind_intr(struct ice_vsi *vsi, uint16_t
msix_vect,

                PMD_DRV_LOG(INFO, "queue %d is binding to vect %d",
                            base_queue + i, msix_vect);
-               /* set ITR0 value */
-               ICE_WRITE_REG(hw, GLINT_ITR(0, msix_vect), 0x10);
+               /* set ITR0 value
+                * Empirical configuration for optimal real time latency
+                * reduced interrupt throttling to 2 ms
+                * Columbiaville pre-PRQ : local patch subject to change
+                */
+               ICE_WRITE_REG(hw, GLINT_ITR(0, msix_vect), 0x1);
+               ICE_WRITE_REG(hw, QRX_ITR(base_queue + i), QRX_ITR_NO_EXPR_M);
                ICE_WRITE_REG(hw, QINT_RQCTL(base_queue + i), val);
                ICE_WRITE_REG(hw, QINT_TQCTL(base_queue + i), val_tx);
        }
```

5. Build and install the DPDK.

```
See https://doc.dpdk.org/guides/prog_guide/build-sdk-meson.html
Insert VFIO module
```

6. Make sure that the i40e is patched with the code below to get the best latency of packet processing.

```
--- i40e.h    2018-11-30 11:27:00.000000000 +0000
+++ i40e_patched.h    2019-03-06 15:49:06.877522427 +0000
```

```
@@ -451,7 +451,7 @@

 #define I40E_QINT_RQCTL_VAL(qp, vector, nextq_type) \
     (I40E_QINT_RQCTL_CAUSE_ENA_MASK | \
-    (I40E_RX_ITR << I40E_QINT_RQCTL_ITR_INDX_SHIFT) | \
+    (I40E_ITR_NONE << I40E_QINT_RQCTL_ITR_INDX_SHIFT) | \
     ((vector) << I40E_QINT_RQCTL_MSIX_INDX_SHIFT) | \
     ((qp) << I40E_QINT_RQCTL_NEXTQ_INDX_SHIFT) | \
     (I40E_QUEUE_TYPE_##nextq_type << I40E_QINT_RQCTL_NEXTQ_TYPE_SHIFT))

--- i40e_main.c      2018-11-30 11:27:00.000000000 +0000
+++ i40e_main_patched.c        2019-03-06 15:46:13.521518062 +0000
@@ -15296,6 +15296,9 @@
              pf->hw_features |= I40E_HW_HAVE_CRT_RETIMER;
     /* print a string summarizing features */
     i40e_print_features(pf);
+
+    /* write to this register to clear rx descriptor */
+    i40e_aq_debug_write_register(hw, 0x0012A504, 0, NULL);

     return 0;
```

A.3 Configuration of System

1. Boot Linux with the following arguments:

```
cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-3.10.0-1062.12.1.rt56.1042.el7.x86_64 root=/dev/mapper/centos-
root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap intel_iommu=on
iommu=pt usbcore.autosuspend=-1 selinux=0 enforcing=0 nmi_watchdog=0
softlockup_panic=0 audit=0 intel_pstate=disable cgroup_memory=1 cgroup_enable=memory
mce=off idle=poll hugepagesz=1G hugepages=16 hugepagesz=2M hugepages=0
default_hugepagesz=1G isolcpus=1-19,21-39 rcu_nocbs=1-19,21-39 kthread_cpus=0,20
irqaffinity=0,20 nohz_full=1-19,21-39
```

2. Boot Linux with the following arguments for Icelake CPU:

```
cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-3.10.0-957.10.1.rt56.921.el7.x86_64 root=/dev/mapper/centos-root
ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet
intel_iommu=off usbcore.autosuspend=-1 selinux=0 enforcing=0 nmi_watchdog=0
softlockup_panic=0 audit=0 intel_pstate=disable cgroup_disable=memory mce=off
hugepagesz=1G hugepages=40 hugepagesz=2M hugepages=0 default_hugepagesz=1G
isolcpus=1-23,25-47 rcu_nocbs=1-23,25-47 kthread_cpus=0 irqaffinity=0 nohz_full=1-
23,25-47
```

3. Download from Intel Website and install updated version of i40e driver if needed. The current recommended version of i40e is 2.14.13. However, any latest version of i40e after 2.9.21 expected to be functional for O-RAN FH.

4. For Columbiaville download Intel® Ethernet 800 Series (Columbiaville) CVL2.3 B0/C0 Sampling Sample Validation Kit (SVK) from Intel Customer Content Library. The current recommended version of ICE driver is 1.3.2 with ICE COMMS Package version 1.3.24.0. IAVF recommended version 4.0.2

5. Identify PCIe Bus address of the Front Haul NIC (Fortville):

```
lspci|grep Eth
86:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
86:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
88:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
88:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
```

6. Identify PCIe Bus address of the Front Haul NIC (Columbiaville):

```
lspci |grep Eth
18:00.0 Ethernet controller: Intel Corporation Device 1593 (rev 02)
```

```
18:00.1 Ethernet controller: Intel Corporation Device 1593 (rev 02)
18:00.2 Ethernet controller: Intel Corporation Device 1593 (rev 02)
18:00.3 Ethernet controller: Intel Corporation Device 1593 (rev 02)
51:00.0 Ethernet controller: Intel Corporation Device 1593 (rev 02)
51:00.1 Ethernet controller: Intel Corporation Device 1593 (rev 02)
51:00.2 Ethernet controller: Intel Corporation Device 1593 (rev 02)
51:00.3 Ethernet controller: Intel Corporation Device 1593 (rev 02)
```

7. Identify the Ethernet device name:

```
ethtool -i enp33s0f0
driver: i40e
version: 2.14.13
firmware-version: 8.20 0x80009bd4 1.2879.0
expansion-rom-version:
bus-info: 0000:21:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yesEnable
```

or

```
ethtool -i enp81s0f0
driver: ice
version: 1.3.2
firmware-version: 2.3 0x80005D18
expansion-rom-version:
bus-info: 0000:51:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

8. Enable 3 virtual functions (VFs) on the each of two ports of each NIC:

```
#!/bin/bash

echo 0 > /sys/bus/pci/devices/0000\:88\:00.0/sriov_numvfs
echo 0 > /sys/bus/pci/devices/0000\:88\:00.1/sriov_numvfs

echo 0 > /sys/bus/pci/devices/0000\:86\:00.0/sriov_numvfs
echo 0 > /sys/bus/pci/devices/0000\:86\:00.1/sriov_numvfs

modprobe -r iavf
modprobe iavf

echo 3 > /sys/bus/pci/devices/0000\:88\:00.0/sriov_numvfs
echo 3 > /sys/bus/pci/devices/0000\:88\:00.1/sriov_numvfs

echo 3 > /sys/bus/pci/devices/0000\:86\:00.0/sriov_numvfs
echo 3 > /sys/bus/pci/devices/0000\:86\:00.1/sriov_numvfs

a=8

if [ -z "$1" ]
then
b=0
elif [ $1 -lt $a ]
then
b=$1
else
echo " Usage $0 qos with 0<= qos <= 7 with 0 as a default if no qos is provided"
```

```
exit 1
fi

#O-DU
ip link set enp136s0f0 vf 0 mac 00:11:22:33:00:00 vlan 1 qos $b
ip link set enp136s0f1 vf 0 mac 00:11:22:33:00:10 vlan 1 qos $b

ip link set enp136s0f0 vf 1 mac 00:11:22:33:01:00 vlan 2 qos $b
ip link set enp136s0f1 vf 1 mac 00:11:22:33:01:10 vlan 2 qos $b

ip link set enp136s0f0 vf 2 mac 00:11:22:33:02:00 vlan 3 qos $b
ip link set enp136s0f1 vf 2 mac 00:11:22:33:02:10 vlan 3 qos $b

#O-RU
ip link set enp134s0f0 vf 0 mac 00:11:22:33:00:01 vlan 1 qos $b
ip link set enp134s0f1 vf 0 mac 00:11:22:33:00:11 vlan 1 qos $b

ip link set enp134s0f0 vf 1 mac 00:11:22:33:01:01 vlan 2 qos $b
ip link set enp134s0f1 vf 1 mac 00:11:22:33:01:11 vlan 2 qos $b

ip link set enp134s0f0 vf 2 mac 00:11:22:33:02:01 vlan 3 qos $b
ip link set enp134s0f1 vf 2 mac 00:11:22:33:02:11 vlan 3 qos $b
```

where output is next:

```
ip link show
...
9: enp134s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 3c:fd:fe:b9:f9:60 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:11:22:33:00:01, vlan 1, spoof checking on, link-state auto, trust off
    vf 1 MAC 00:11:22:33:01:01, vlan 2, spoof checking on, link-state auto, trust off
    vf 2 MAC 00:11:22:33:02:01, vlan 3, spoof checking on, link-state auto, trust off
11: enp134s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 3c:fd:fe:b9:f9:61 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:11:22:33:00:11, vlan 1, spoof checking on, link-state auto, trust off
    vf 1 MAC 00:11:22:33:01:11, vlan 2, spoof checking on, link-state auto, trust off
    vf 2 MAC 00:11:22:33:02:11, vlan 3, spoof checking on, link-state auto, trust off
12: enp136s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 3c:fd:fe:b9:f8:b4 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:11:22:33:00:00, vlan 1, spoof checking on, link-state auto, trust off
    vf 1 MAC 00:11:22:33:01:00, vlan 2, spoof checking on, link-state auto, trust off
    vf 2 MAC 00:11:22:33:02:00, vlan 3, spoof checking on, link-state auto, trust off
14: enp136s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 3c:fd:fe:b9:f8:b5 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:11:22:33:00:10, vlan 1, spoof checking on, link-state auto, trust off
    vf 1 MAC 00:11:22:33:01:10, vlan 2, spoof checking on, link-state auto, trust off
    vf 2 MAC 00:11:22:33:02:10, vlan 3, spoof checking on, link-state auto, trust off
...
```

More information about VFs supported by Intel NICs can be found at https://doc.dpdk.org/guides/nics/intel_vf.html.

The resulting configuration can look like the listing below, where six new VFs were added for each O-DU and O-RU port:

```
lspci|grep Eth
86:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
86:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
86:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:02.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
86:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
```

```
86:0a.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
88:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
88:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:02.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.1 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
88:0a.2 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev 02)
```

9. Example where O-DU and O-RU simulation run on the same system:

O-DU:

```
cat ./run_o_du.sh
#! /bin/bash

ulimit -c unlimited
echo 1 > /proc/sys/kernel/core_uses_pid

./build/sample-app --usecasefile ./usecase/cat_b/mu1_100mhz/301/usecase_du.cfg --num_eth_vfs 6 \
--vf_addr_o_xu_a "0000:88:02.0,0000:88:0a.0" \
--vf_addr_o_xu_b "0000:88:02.1,0000:88:0a.1" \
--vf_addr_o_xu_c "0000:88:02.2,0000:88:0a.2"
```

O-RU:

```
cat ./run_o_ru.sh
#! /bin/bash
ulimit -c unlimited
echo 1 > /proc/sys/kernel/core_uses_pid

./build/sample-app --usecasefile ./usecase/cat_b/mu1_100mhz/301/usecase_ru.cfg --num_eth_vfs 6 \
--vf_addr_o_xu_a "0000:86:02.0,0000:86:0a.0" \
--vf_addr_o_xu_b "0000:86:02.1,0000:86:0a.1" \
--vf_addr_o_xu_c "0000:86:02.2,0000:86:0a.2"
```

A.4 Install and Configure Sample Application

To install and configure the sample application:

1. Set up the environment:

```
For Skylake and Cascadelake
export GTEST_ROOT=`pwd`/gtest-1.7.0
export RTE_SDK=`pwd`/dpdk-20.11
export RTE_TARGET=x86_64-native-linuxapp-icc
export DIR_WIRELESS_SDK_ROOT=`pwd`/wireless_sdk
export WIRELESS_SDK_TARGET_ISA=avx512
export SDK_BUILD=build-${WIRELESS_SDK_TARGET_ISA}-icc
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
export MLOG_DIR=`pwd`/flexran_l1_sw/libs/mlog
export XRAN_DIR=`pwd`/flexran_xran

for Icelake
export GTEST_ROOT=`pwd`/gtest-1.7.0
export RTE_SDK=`pwd`/dpdk-20.11
export RTE_TARGET=x86_64-native-linuxapp-icc
export DIR_WIRELESS_SDK_ROOT=`pwd`/wireless_sdk
export WIRELESS_SDK_TARGET_ISA=snc
export SDK_BUILD=build-${WIRELESS_SDK_TARGET_ISA}-icc
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
export MLOG_DIR=`pwd`/flexran_l1_sw/libs/mlog
export XRAN_DIR=`pwd`/flexran_xran
```

2. export `FLEXRAN_SDK=${DIR_WIRELESS_SDK}/install` Compile `mlog` library:

```
[turner@xran home]$ cd $MLOG_DIR
[turner@xran xran]$ ./build.sh
```

3. Compile xRAN library and test the application:

```
[turner@xran home]$ cd $XRAN_DIR
[turner@xran xran]$ ./build.sh
```

4. Configure the sample app.

   IQ samples can be generated using Octave* and script libs`/xran/app/gen_test.m`. (CentOS* has `octave-3.8.2-20.el7.x86_64` compatible with `get_test.m`)

   Other IQ sample test vectors can be used as well. The format of IQ samples is binary int16_t I and Q for N slots of the OTA RF signal. For example, for `mmWave`, it corresponds to `792RE*2*14symbol*8slots*10 ms = 3548160` bytes per antenna. Refer to comments in `gen_test.m` to correctly specify the configuration for IQ test vector generation.

   Update `usecase_du.dat` (or `usecase_ru.cfg`) with a suitable configuration for your scenario.

   Update `config_file_o_du.dat` (or `config_file_o_ru.dat`) with a suitable configuration for your scenario.

   Update `run_o_du.sh` (`run_o_ru.sh`) with PCIe bus address of VF0 and VF1 used for U-plane and C-plane correspondingly.

5. Run the application using `run_o_du.sh` (`run_o_ru.sh`).

A.4.1 Install and Configure FlexRAN 5G NR L1 Application

The 5G NR layer 1 application can be used for executing the scenario for mmWave with either the RU sample application or just the O-DU side. The current release supports the constant configuration of the slot pattern and RB allocation on the PHY side. The build process follows the same basic steps as for the sample application above and is similar to compiling 5G NR l1app for mmWave with Front Haul FPGA. Please follow the general build process in the FlexRAN 5G NR Reference Solution L1 User Guide (refer to Table 2.)

1. xRAN library is enabled by default l1 application:
2. Build the 5G NR L1 application using the command:

```
./flexran_build.sh -r 5gnr_mmw -i avx512 -m sdk -m fb -m mlog -m wls -m
5gnr_l1app_mmw  -m xran -m 5gnr_testmac
```

3. Configure the L1app using `bin/nr5g/gnb/l1/phycfg_xran.xml and xrancfg_sub6.xml (or other xml if it is mmwave or massive MIMO)`.

```
<XranConfig>
    <version>20.08</version>
        <!-- numbers of O-RU connected to O-DU. All O-RUs are the same capabilities.
Max O-RUs is per XRAN_PORTS_NUM i.e. 4 -->
    <oRuNum>1</oRuNum>
    <!--  # 10G,25G,40G,100G speed of Physical connection on O-RU -->
    <oRuEthLinkSpeed>25</oRuEthLinkSpeed>
    <!--  # 1, 2, 3 total number of links per O-RU (Fronthaul Ethernet link in IOT
spec) -->
    <oRuLinesNumber>1</oRuLinesNumber>

    <!-- O-RU 0 -->
    <PciBusAddoRu0Vf0>0000:51:01.0</PciBusAddoRu0Vf0>
    <PciBusAddoRu0Vf1>0000:51:01.1</PciBusAddoRu0Vf1>
    <PciBusAddoRu0Vf2>0000:51:01.2</PciBusAddoRu0Vf2>
    <PciBusAddoRu0Vf3>0000:51:01.3</PciBusAddoRu0Vf3>

    <!-- O-RU 1 -->
    <PciBusAddoRu1Vf0>0000:51:01.4</PciBusAddoRu1Vf0>
    <PciBusAddoRu1Vf1>0000:51:01.5</PciBusAddoRu1Vf1>
    <PciBusAddoRu1Vf2>0000:51:01.6</PciBusAddoRu1Vf2>
    <PciBusAddoRu1Vf3>0000:51:01.7</PciBusAddoRu1Vf3>

    <!-- O-RU 2 -->
    <PciBusAddoRu2Vf0>0000:51:02.0</PciBusAddoRu2Vf0>
    <PciBusAddoRu2Vf1>0000:51:02.1</PciBusAddoRu2Vf1>
    <PciBusAddoRu2Vf2>0000:51:02.2</PciBusAddoRu2Vf2>
    <PciBusAddoRu2Vf3>0000:51:02.3</PciBusAddoRu2Vf3>

    <!-- O-RU 4 -->
    <PciBusAddoRu3Vf0>0000:00:00.0</PciBusAddoRu3Vf0>
    <PciBusAddoRu3Vf1>0000:00:00.0</PciBusAddoRu3Vf1>
    <PciBusAddoRu3Vf2>0000:00:00.0</PciBusAddoRu3Vf2>
    <PciBusAddoRu3Vf3>0000:00:00.0</PciBusAddoRu3Vf3>

    <!-- remote O-RU 0 Eth Link 0 VF0, VF1-->
    <oRuRem0Mac0>00:11:22:33:00:01<oRuRem0Mac0>
    <oRuRem0Mac1>00:11:22:33:00:11<oRuRem0Mac1>
    <!-- remote O-RU 0 Eth Link 1 VF2, VF3 -->
    <oRuRem0Mac2>00:11:22:33:00:21<oRuRem0Mac2>
    <oRuRem0Mac3>00:11:22:33:00:31<oRuRem0Mac3>

    <!-- remote O-RU 1 Eth Link 0 VF4, VF5-->
    <oRuRem1Mac0>00:11:22:33:01:01<oRuRem1Mac0>
    <oRuRem1Mac1>00:11:22:33:01:11<oRuRem1Mac1>
    <!-- remote O-RU 1 Eth Link 1 VF6, VF7 -->
    <oRuRem1Mac2>00:11:22:33:01:21<oRuRem1Mac2>
    <oRuRem1Mac3>00:11:22:33:01:31<oRuRem1Mac3>

    <!-- remote O-RU 2 Eth Link 0 VF8, VF9 -->
    <oRuRem2Mac0>00:11:22:33:02:01<oRuRem2Mac0>
    <oRuRem2Mac1>00:11:22:33:02:11<oRuRem2Mac1>
    <!-- remote O-RU 2 Eth Link 1 VF10, VF11-->
    <oRuRem2Mac2>00:11:22:33:02:21<oRuRem2Mac2>
    <oRuRem2Mac3>00:11:22:33:02:31<oRuRem2Mac3>

    <!-- remote O-RU 2 Eth Link 0 VF12, VF13 -->
    <oRuRem3Mac0>00:11:22:33:03:01<oRuRem3Mac0>
    <oRuRem3Mac1>00:11:22:33:03:11<oRuRem3Mac1>
    <!-- remote O-RU 2 Eth Link 1 VF14, VF15-->
    <oRuRem3Mac2>00:11:22:33:03:21<oRuRem3Mac2>
    <oRuRem3Mac3>00:11:22:33:03:31<oRuRem3Mac3>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu0NumCc>1</oRu0NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu0Cc0PhyId>0</oRu0Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu0Cc1PhyId>1</oRu0Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu0Cc2PhyId>2</oRu0Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu0Cc3PhyId>3</oRu0Cc3PhyId>
        <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu0Cc4PhyId>4</oRu0Cc4PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu0Cc5PhyId>5</oRu0Cc5PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu0Cc6PhyId>6</oRu0Cc6PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu0Cc7PhyId>7</oRu0Cc7PhyId>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu0Cc8PhyId>8</oRu0Cc8PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu0Cc9PhyId>9</oRu0Cc9PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu0Cc10PhyId>10</oRuCc10PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu0Cc11PhyId>11</oRu0Cc11PhyId>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu1NumCc>1</oRu1NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu1Cc0PhyId>1</oRu1Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu1Cc1PhyId>1</oRu1Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu1Cc2PhyId>2</oRu1Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu1Cc3PhyId>3</oRu1Cc3PhyId>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu2NumCc>1</oRu2NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu2Cc0PhyId>2</oRu2Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu2Cc1PhyId>1</oRu2Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu2Cc2PhyId>2</oRu2Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu2Cc3PhyId>3</oRu2Cc3PhyId>

    <!-- XRAN Thread (core where the XRAN polling function is pinned: Core, priority,
Policy [0: SCHED_FIFO 1: SCHED_RR] -->
    <xRANThread>19, 96, 0</xRANThread>

    <!-- core mask for XRAN Packets Worker (core where the XRAN packet processing is
pinned): Core, priority, Policy [0: SCHED_FIFO 1: SCHED_RR] -->
    <xRANWorker>0x0, 96, 0</xRANWorker>
    <!-- XRAN: Category of O-RU 0 - Category A, 1 - Category B -->
    <Category>0</Category>

    <!-- XRAN: enable sleep on PMD cores -->
    <xranPmdSleep>0</xranPmdSleep>


    <!-- RU Settings -->
    <Tadv_cp_dl>25</Tadv_cp_dl>
    <!-- Reception Window C-plane DL-->
    <T2a_min_cp_dl>50</T2a_min_cp_dl>
    <T2a_max_cp_dl>140</T2a_max_cp_dl>
    <!-- Reception Window C-plane UL-->
    <T2a_min_cp_ul>50</T2a_min_cp_ul>
    <T2a_max_cp_ul>140</T2a_max_cp_ul>
    <!-- Reception Window U-plane -->
    <T2a_min_up>25</T2a_min_up>
    <T2a_max_up>140</T2a_max_up>
    <!-- Transmission Window U-plane -->
    <Ta3_min>20</Ta3_min>
    <Ta3_max>32</Ta3_max>

    <!-- O-DU Settings -->
    <!-- MTU size -->
    <MTU>9600</MTU>
    <!-- VLAN Tag used for C-Plane -->
    <c_plane_vlan_tag>1</c_plane_vlan_tag>
    <u_plane_vlan_tag>2</u_plane_vlan_tag>

    <!-- Transmission Window Fast C-plane DL -->
    <T1a_min_cp_dl>70</T1a_min_cp_dl>
    <T1a_max_cp_dl>100</T1a_max_cp_dl>
    <!-- Transmission Window Fast C-plane UL -->
    <T1a_min_cp_ul>60</T1a_min_cp_ul>
    <T1a_max_cp_ul>70</T1a_max_cp_ul>
    <!-- Transmission Window U-plane -->
    <T1a_min_up>35</T1a_min_up>
    <T1a_max_up>50</T1a_max_up>
    <!-- Reception Window U-Plane-->
    <Ta4_min>0</Ta4_min>
    <Ta4_max>45</Ta4_max>

    <!-- Enable Control Plane -->
    <EnableCp>1</EnableCp>

    <DynamicSectionEna>0</DynamicSectionEna>
    <!-- Enable Dynamic section allocation for UL -->
    <DynamicSectionEnaUL>0</DynamicSectionEnaUL>
    <xRANSFNWrap>0</xRANSFNWrap>
    <!-- Total Number of DL PRBs per symbol (starting from RB 0) that is transmitted
(used for testing. If 0, then value is used from PHY_CONFIG_API) -->
    <xRANNumDLPRBs>0</xRANNumDLPRBs>
    <!-- Total Number of UL PRBs per symbol (starting from RB 0) that is received
(used for testing. If 0, then value is used from PHY_CONFIG_API) -->
    <xRANNumULPRBs>0</xRANNumULPRBs>
    <!-- refer to alpha as defined in section 9.7.2 of O-RAN spec. this value should
be alpha*(1/1.2288ns), range 0 - 1e7 (ns) -->
    <Gps_Alpha>0</Gps_Alpha>
    <!-- beta value as defined in section 9.7.2 of ORAN spec. range -32767 ~ +32767 -
->
    <Gps_Beta>0</Gps_Beta>

    <!-- XRAN: Compression mode on O-DU <-> O-RU 0 - no comp 1 - BFP -->
    <xranCompMethod>0</xranCompMethod>

    <oRu0nPrbElemDl>1</oRu0nPrbElemDl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu0PrbElemDl0>0,48,0,14,1,1,1,9,1,0,0</oRu0PrbElemDl0>
    <oRu0PrbElemDl1>48,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemDl1>
    <oRu0PrbElemDl2>96,48,0,14,3,1,1,9,1,0,0</oRu0PrbElemDl2>
    <oRu0PrbElemDl3>144,48,0,14,4,1,1,9,1,0,0</oRu0PrbElemDl3>
    <oRu0PrbElemDl4>144,36,0,14,5,1,1,9,1,0,0</oRu0PrbElemDl4>
    <oRu0PrbElemDl5>180,36,0,14,6,1,1,9,1,0,0</oRu0PrbElemDl5>
    <oRu0PrbElemDl6>216,36,0,14,7,1,1,9,1,0,0</oRu0PrbElemDl6>
    <oRu0PrbElemDl7>252,21,0,14,8,1,1,9,1,0,0</oRu0PrbElemDl7>
    <oRu0nPrbElemUl>1</nPrbElemUl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu0PrbElemUl0>0,48,0,14,1,1,1,9,1,0,0</oRu0PrbElemUl0>
    <oRu0PrbElemUl1>48,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemUl1>
    <oRu0PrbElemUl2>72,36,0,14,3,1,1,9,1,0,0</oRu0PrbElemUl2>
    <oRu0PrbElemUl3>108,36,0,14,4,1,1,9,1,0,0</oRu0PrbElemUl3>
    <oRu0PrbElemUl4>144,36,0,14,5,1,1,9,1,0,0</oRu0PrbElemUl4>
    <oRu0PrbElemUl5>180,36,0,14,6,1,1,9,1,0,0</oRu0PrbElemUl5>
    <oRu0PrbElemUl6>216,36,0,14,7,1,1,9,1,0,0</oRu0PrbElemUl6>
    <oRu0PrbElemUl7>252,21,0,14,8,1,1,9,1,0,0</oRu0PrbElemUl7>

</XranConfig>
```

4. Modify `bin/nr5g/gnb/l1/dpdk.sh` (change PCIe addresses from VFs).

```
         $RTE_SDK/usertools/dpdk-devbind.py --bind=vfio-pci 0000:21:02.0
         $RTE_SDK/usertools/dpdk-devbind.py --bind=vfio-pci 0000:21:02.1
```

5. Use configuration of test mac per:

```
/bin/nr5g/gnb.testmac/cascade_lake-sp/csxsp_mu1_100mhz_mmimo_hton_xran.cfg
phystart 4 0 40200
<!--   mmWave mu 3 100MHz                    -->
TEST_FD, 1002, 1, fd/mu3_100mhz/2/fd_testconfig_tst2.cfg
```

6. To execute l1app with O-DU functionality according to O-RAN Fronthaul specification, enter:

```
[root@xran flexran] cd ./bin/nr5g/gnb/l1
[root@xran l1]#./l1.sh -xran
where output corresponding L1 is:
[root@sc12-xran-sub6 l1]# ./l1.sh -xranmmw
Radio mode with XRAN - mmWave 100Mhz
DPDK WLS MODE
kernel.sched_rt_runtime_us = -1
kernel.shmmax = 2147483648
kernel.shmall = 2147483648
Note: Forwarding request to 'systemctl disable irqbalance.service'.
using configuration file phycfg_xran_mmw.xml
>> Running... ./l1app table 0 1 --cfgfile=phycfg_xran_mmw.xml
FlexRAN SDK bblib_layerdemapping_5gnr version #DIRTY#
FlexRAN SDK bblib_layermapping_5gnr version #DIRTY#
FlexRAN SDK bblib_cestimate_5gnr_version #DIRTY#
FlexRAN SDK bblib_pucch_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_llr_demapping version #DIRTY#
FlexRAN SDK bblib_pdcch_remapping_5gnr_version version #DIRTY#
FlexRAN SDK bblib_reed_muller version #DIRTY#
FlexRAN SDK bblib_lte_modulation version #DIRTY#
FlexRAN SDK bblib_polar_decoder_5gnr version #DIRTY#
FlexRAN SDK bblib_polar_rate_dematching_5gnr version #DIRTY#
FlexRAN SDK bblib_PhaseNoise_5G version #DIRTY#
FlexRAN SDK bblib_mimo_mmse_detection_5gnr version #DIRTY#
FlexRAN SDK bblib_fd_correlation version #DIRTY#
FlexRAN SDK bblib_scramble_5gnr version #DIRTY#
FlexRAN SDK bblib_pucch_equ_5gnr version #DIRTY#
FlexRAN SDK bblib_ta_compensation_version_5gnr #DIRTY#
FlexRAN SDK bblib_polar_encoder_5gnr version #DIRTY#
FlexRAN SDK bblib_prach_5gnr version #DIRTY#
FlexRAN SDK bblib_fft_ifft version #DIRTY#
FlexRAN SDK bblib_pucch_5gnr version #DIRTY#
FlexRAN SDK bblib_common version #DIRTY#
FlexRAN SDK bblib_lte_crc version #DIRTY#
FlexRAN SDK bblib_lte_dft_idft version #DIRTY#
FlexRAN SDK bblib_irc_rnn_calculation_5gnr_version #DIRTY#
FlexRAN SDK bblib_mmse_irc_mimo_5gnr_version #DIRTY#
FlexRAN SDK bblib_srs_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_zf_matrix_gen version #DIRTY#
FlexRAN SDK bblib_beamforming_dl_expand version #DIRTY#
=========================
5GNR PHY Application
=========================
---------------------------
PhyCfg.xml Version: 20.04
---------------------------
 --version=20.04
 --successiveNoApi=15
 --wls_dev_name=wls0
 --wlsMemorySize=0x3F600000
 --dlIqLog=0
 --ulIqLog=0
 --iqLogDumpToFile=0x0
 --phyMlog=1
 --phyStats=1
 --dpdkMemorySize=8192
 --dpdkIovaMode=0
 --dpdkBasebandFecMode=1
 --dpdkBasebandDevice=0000:1f:00.1
 --radioEnable=4
 --ferryBridgeMode=1
 --ferryBridgeEthPort=1
 --ferryBridgeSyncPorts=0
 --ferryBridgeOptCableLoopback=0
 --radioCfg0PCIeEthDev=0000:19:00.0
 --radioCfg0DpdkRx=1
 --radioCfg0DpdkTx=2
 --radioCfg0TxAnt=2
 --radioCfg0RxAnt=2
 --radioCfg0RxAgc=0
 --radioCfg0NumCell=1
 --radioCfg0Cell0PhyId=0
 --radioCfg0Cell1PhyId=1
 --radioCfg0Cell2PhyId=2
 --radioCfg0Cell3PhyId=3
 --radioCfg0Cell4PhyId=4
 --radioCfg0Cell5PhyId=5
 --radioCfg0riuMac=11:22:33:44:55:66
 --radioCfg1PCIeEthDev=0000:03:00.1
 --radioCfg1DpdkRx=1
 --radioCfg1DpdkTx=1
 --radioCfg1TxAnt=4
 --radioCfg1RxAnt=4
 --radioCfg1RxAgc=0
 --radioCfg1NumCell=1
 --radioCfg1Cell0PhyId=2
 --radioCfg1Cell1PhyId=3
 --radioCfg1Cell2PhyId=2
 --radioCfg1Cell3PhyId=3
 --radioCfg1riuMac=ac:1f:6b:2c:9f:07
 --radioCfg2PCIeEthDev=0000:05:00.0
 --radioCfg2DpdkRx=10
 --radioCfg2DpdkTx=11
 --radioCfg2TxAnt=4
 --radioCfg2RxAnt=4
 --radioCfg2RxAgc=0
 --radioCfg2NumCell=2
 --radioCfg2Cell0PhyId=4
 --radioCfg2Cell1PhyId=5
 --radioCfg2Cell2PhyId=2
 --radioCfg2Cell3PhyId=3
 --radioCfg2riuMac=ac:1f:6b:2c:9f:07
 --radioCfg3PCIeEthDev=0000:05:00.1
```

```
--radioCfg3DpdkRx=12
--radioCfg3DpdkTx=13
--radioCfg3TxAnt=4
--radioCfg3RxAnt=4
--radioCfg3RxAgc=0
--radioCfg3NumCell=2
--radioCfg3Cell0PhyId=6
--radioCfg3Cell1PhyId=7
--radioCfg3Cell2PhyId=2
--radioCfg3Cell3PhyId=3
--radioCfg3riuMac=ac:1f:6b:2c:9f:07
--radioCfg4PCIeEthDev=0000:00:08.0
--radioCfg4DpdkRx=14
--radioCfg4DpdkTx=15
--radioCfg4TxAnt=4
--radioCfg4RxAnt=4
--radioCfg4RxAgc=0
--radioCfg4NumCell=2
--radioCfg4Cell0PhyId=8
--radioCfg4Cell1PhyId=9
--radioCfg4Cell2PhyId=2
--radioCfg4Cell3PhyId=3
--radioCfg4riuMac=ac:1f:6b:2c:9f:07
--radioCfg5PCIeEthDev=0000:08:00.0
--radioCfg5DpdkRx=16
--radioCfg5DpdkTx=16
--radioCfg5TxAnt=4
--radioCfg5RxAnt=4
--radioCfg5RxAgc=0
--radioCfg5NumCell=2
--radioCfg5Cell0PhyId=10
--radioCfg5Cell1PhyId=11
--radioCfg5Cell2PhyId=2
--radioCfg5Cell3PhyId=3
--radioCfg5riuMac=ac:1f:6b:2c:9f:07
--radioCfg6PCIeEthDev=0000:00:05.0
--radioCfg6DpdkRx=16
--radioCfg6DpdkTx=16
--radioCfg6TxAnt=4
--radioCfg6RxAnt=4
--radioCfg1RxAgc=0
--radioCfg6NumCell=2
--radioCfg6Cell0PhyId=12
--radioCfg6Cell1PhyId=13
--radioCfg6Cell2PhyId=2
--radioCfg6Cell3PhyId=3
--radioCfg6riuMac=ac:1f:6b:2c:9f:07
--radioCfg7PCIeEthDev=0000:00:06.0
--radioCfg7DpdkRx=16
--radioCfg7DpdkTx=16
--radioCfg7TxAnt=4
--radioCfg7RxAnt=4
--radioCfg7RxAgc=0
--radioCfg7NumCell=2
--radioCfg7Cell0PhyId=14
--radioCfg7Cell1PhyId=15
--radioCfg7Cell2PhyId=2
--radioCfg7Cell3PhyId=3
--radioCfg7riuMac=ac:1f:6b:2c:9f:07
--radioPort0=0
--radioPort1=1
--radioPort2=2
```

```
--radioPort3=3
--radioPort4=4
--radioPort5=5
--radioPort6=6
--radioPort7=7
--PdschSymbolSplit=0
--PdschDlWeightSplit=0
--FecEncSplit=4
--PuschChanEstSplit=0
--PuschMmseSplit=0
--PuschLlrRxSplit=0
--PuschUlWeightSplit=0
--FecDecEarlyTermDisable=0
--FecDecNumIter=0
--FecDecSplit=4
--llrOutDecimalDigit=2
--IrcEnableThreshold=-10
--CEInterpMethod=0
--PucchSplit=0
--SrsCeSplit=0
--prachDetectThreshold=10000
--MlogSubframes=128
--MlogCores=20
--MlogSize=3084
--systemThread=0, 0, 0
--timerThread=0, 96, 0
--xRANThread=4, 96, 0
--xRANWorker=0x0, 96, 0
--FpgaDriverCpuInfo=2, 96, 0
--FrontHaulCpuInfo=3, 96, 0
--radioDpdkMaster=2, 99, 0
--BbuPoolSleepEnable=1
--BbuPoolThreadCorePriority=94
--BbuPoolThreadCorePolicy=0
--BbuPoolThreadDefault_0_63=0x68
--BbuPoolThreadDefault_64_127=0x0
--BbuPoolThreadSrs_0_63=0x0
--BbuPoolThreadSrs_64_127=0x0
--BbuPoolThreadDlbeam_0_63=0x0
--BbuPoolThreadDlbeam_64_127=0x0
--BbuPoolThreadUrllc=8
--FrontHaulTimeAdvance=9450
--nEthPorts=459523
--nPhaseCompFlag=1
--nFecFpgaVersionMu3=0xFC101800
--nFecFpgaVersionMu0_1=0x0319d420
--nFhFpgaVersionMu3=0x8001000F
--nFhFpgaVersionMu0_1=0x90010008
--dpdkXranDeviceCP=0000:21:02.1
--dpdkXranDeviceUP=0000:21:02.0
--DuMac=00:11:22:33:44:66
--RuMac=00:11:22:33:44:55
--Category=0
--xranPmdSleep=0
--Tadv_cp_dl=25
--T2a_min_cp_dl=50
--T2a_max_cp_dl=140
--T2a_min_cp_ul=50
--T2a_max_cp_ul=140
--T2a_min_up=25
--T2a_max_up=140
--Ta3_min=20
```

```
 --Ta3_max=32
 --MTU=9600
 --c_plane_vlan_tag=1
 --u_plane_vlan_tag=2
 --T1a_min_cp_dl=70
 --T1a_max_cp_dl=100
 --T1a_min_cp_ul=60
 --T1a_max_cp_ul=70
 --T1a_min_up=35
 --T1a_max_up=50
 --Ta4_min=0
 --Ta4_max=45
 --DynamicSectionEna=0
 --xRANSFNWrap=0
 --xRANNumDLPRBs=0
 --xRANNumULPRBs=0
 --Gps_Alpha=0
 --Gps_Beta=0
 --xranCompMethod=0
 --nPrbElemDl=0
 --PrbElemDl0=0,48,0,14,1,1,1,9,1
 --PrbElemDl1=48,48,0,14,2,1,1,9,1
 --PrbElemDl2=96,48,0,14,3,1,1,9,1
 --PrbElemDl3=144,48,0,14,4,1,1,9,1
 --PrbElemDl4=144,36,0,14,5,1,1,9,1
 --PrbElemDl5=180,36,0,14,6,1,1,9,1
 --PrbElemDl6=216,36,0,14,7,1,1,9,1
 --PrbElemDl7=252,21,0,14,8,1,1,9,1
 --nPrbElemUl=0
 --PrbElemUl0=0,48,0,14,1,1,1,9,1
 --PrbElemUl1=48,48,0,14,2,1,1,9,1
 --PrbElemUl2=72,36,0,14,3,1,1,9,1
 --PrbElemUl3=108,36,0,14,4,1,1,9,1
 --PrbElemUl4=144,36,0,14,5,1,1,9,1
 --PrbElemUl5=180,36,0,14,6,1,1,9,1
 --PrbElemUl6=216,36,0,14,7,1,1,9,1
 --PrbElemUl7=252,21,0,14,8,1,1,9,1
 --StreamStats=0
 --StreamIp=127.0.0.1
 --StreamPort=2000

wls_dev_filename: wls0
phycfg_apply: Initialize Radio Interface with XRAN library
Setting FecEncSplit to 1 to run on HW accelerator
Setting FecDecSplit to 1 to run on HW accelerator

timer_set_tsc_freq_from_clock: System clock (rdtsc) resolution 1596249953 [Hz]
                               Ticks per usec 1596
MLogOpen: filename(l1mlog.bin) mlogSubframes (128), mlogCores(20), mlogSize(3084)
mlog_mask (-1)
    mlogSubframes (128), mlogCores(20), mlogSize(3084)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1596250020 [Hz]
        Ticks per us 1596
    MLog Storage: 0x7f6e5b0e3100 -> 0x7f6e5b86b52c [ 7898156 bytes ]
    localMLogFreqReg: 1596. Storing: 1596
    Mlog Open successful

di_xran_init
di_xran_cfg_setup successful
 xran_init: MTU 9600
BBDEV_FEC_ACCL_NR5G
```

```
hw-accelerated bbdev 0000:1f:00.1
total cores 40 c_mask 0x14 core 4 [id] system_core 2 [id] pkt_proc_core 0x0 [mask]
pkt_aux_core 0 [id] timing_core 4 [id]
xran_ethdi_init_dpdk_io: Calling rte_eal_init:wls0 -c 0x14 -n2 --iova-mode=pa --
socket-mem=8192 --socket-limit=8192 --proc-type=auto --file-prefix wls0 -w
0000:00:00.0 -w 0000:1f:00.1
EAL: Detected 40 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Auto-detected process type: PRIMARY
EAL: Multi-process socket /var/run/dpdk/wls0/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available hugepages reported in hugepages-2048kB
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: PCI device 0000:1f:00.1 on NUMA socket 0
EAL:   probe driver: 8086:d90 intel_fpga_5gnr_fec_vf
EAL:   using IOMMU type 1 (Type 1)
EAL: PCI device 0000:21:02.0 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
initializing port 0 for TX, drv=net_i40e_vf
Port 0 MAC: 00 11 22 33 44 66
Port 0: nb_rxd 4096 nb_txd 4096

Checking link status portid [0]  EAL: PCI device 0000:21:02.1 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
initializing port 1 for TX, drv=net_i40e_vf
Port 1 MAC: 00 11 22 33 44 66
Port 1: nb_rxd 4096 nb_txd 4096
Checking link status portid [1]  vf 0 local  SRC MAC: 00 11 22 33 44 66
vf 0 remote DST MAC: 00 11 22 33 44 55
vf 1 local  SRC MAC: 00 11 22 33 44 66
vf 1 remote DST MAC: 00 11 22 33 44 55
xran_init successful, pHandle = 0x5581f440
bbdev_init:
Socket ID: 0
FEC is accelerated through BBDEV:  0000:1f:00.1
wls_layer_init[wls0] nWlsMemorySize[1063256064]
wls_lib: Open wls0 (DPDK memzone)
wls_lib: WLS_Open 0x2bf600000
wls_lib: link: 0 <-> 1
wls_lib: Mode 0
wls_lib: WLS shared management memzone: wls0
wls_lib: hugePageSize on the system is 1073741824
wls_lib: WLS_Alloc [1063256064] bytes


=====================================================================================
======================
PHY VERSION
=====================================================================================
======================
Version: #DIRTY#
IMG-date: Apr 27 2020
IMG-time: 12:54:54
=====================================================================================
======================
DEPENDENCIES VERSIONS
=====================================================================================
======================
FlexRAN BBU pooling version #DIRTY#
FlexRAN SDK bblib_layerdemapping_5gnr version #DIRTY#
FlexRAN SDK bblib_layermapping_5gnr version #DIRTY#
FlexRAN SDK bblib_cestimate_5gnr_version #DIRTY#
FlexRAN SDK bblib_pucch_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_llr_demapping version #DIRTY#
FlexRAN SDK bblib_pdcch_remapping_5gnr_version version #DIRTY#
FlexRAN SDK bblib_reed_muller version #DIRTY#
FlexRAN SDK bblib_lte_modulation version #DIRTY#
FlexRAN SDK bblib_polar_decoder_5gnr version #DIRTY#
FlexRAN SDK bblib_polar_rate_dematching_5gnr version #DIRTY#
FlexRAN SDK bblib_PhaseNoise_5G version #DIRTY#
FlexRAN SDK bblib_mimo_mmse_detection_5gnr version #DIRTY#
FlexRAN SDK bblib_fd_correlation version #DIRTY#
FlexRAN SDK bblib_scramble_5gnr version #DIRTY#
FlexRAN SDK bblib_pucch_equ_5gnr version #DIRTY#
FlexRAN SDK bblib_ta_compensation_version_5gnr #DIRTY#
FlexRAN SDK bblib_polar_encoder_5gnr version #DIRTY#
FlexRAN SDK bblib_prach_5gnr version #DIRTY#
FlexRAN SDK bblib_fft_ifft version #DIRTY#
FlexRAN SDK bblib_pucch_5gnr version #DIRTY#
FlexRAN SDK bblib_lte_crc version #DIRTY#
FlexRAN SDK bblib_common version #DIRTY#
=====================================================================================
======================

=====================================================================================
======================
Non BBU threads in application
=====================================================================================
======================
nr5g_gnb_phy2mac_api_proc_stats_thread: [PID: 112583] binding on [CPU  0] [PRIO:  0]
[POLICY:  1]
wls_rx_handler (non-rt):                [PID: 112587] binding on [CPU  0]
=====================================================================================
======================
PHY>welcome to application console
PHY>Received MSG_TYPE_PHY_UL_IQ_SAMPLES
Processing MSG_TYPE_PHY_UL_IQ_SAMPLES: 0
phydi_read_write_iq_samples: direction[1] nNumerologyMult[8] fftSize[1024, 11088,
SRS: 792] numSubframe[80] numAntenna[2] numPorts[2] nIsRadioMode[1] carrNum[0]
TimerModeFreqDomain[1] PhaseCompensationEnable[0]
filename_in_ul_iq[/home/turner/xran/master/npg_wireless-
flexran_l1_5g_test/fd/mu3_100mhz/2/../../../ul/mu3_100mhz/1/uliq00_tst1.bin]
filename_in_prach_iq[]
Received MSG_TYPE_PHY_CONFIG_REQ: 0
Processing MSG_TYPE_PHY_CONFIG_REQ: 0
phy_bbupool_set_config: Using cores: 0x0000000000000068 for BBU Pool
nBbuPoolSleepEnable: 1
BBU Pooling: queueId = 0, the according nCoreNum = 3, the according cpuSetMask = 0x68
BBU Pooling: gCoreIdxMap[0] = 3 is available!
BBU Pooling: gCoreIdxMap[1] = 5 is available!
BBU Pooling: gCoreIdxMap[2] = 6 is available!
BBU Pooling: taskId =  0 taskName =     DL_L1_CONFIG is registered
BBU Pooling: taskId =  1 taskName =   DL_L1_PDSCH_TB is registered
BBU Pooling: taskId =  2 taskName = DL_L1_PDSCH_SCRAMBLER is registered
BBU Pooling: taskId =  3 taskName = DL_L1_PDSCH_SYMBOL_TX is registered
BBU Pooling: taskId =  4 taskName = DL_L1_PDSCH_RS_GEN is registered
BBU Pooling: taskId =  5 taskName = DL_L1_CONTROL_CHANNELS is registered
BBU Pooling: taskId =  6 taskName =     UL_L1_CONFIG is registered
BBU Pooling: taskId =  7 taskName =  UL_L1_PUSCH_CE0 is registered
BBU Pooling: taskId =  8 taskName =  UL_L1_PUSCH_CE7 is registered
BBU Pooling: taskId =  9 taskName = UL_L1_PUSCH_MMSE0_PRE is registered
BBU Pooling: taskId = 10 taskName = UL_L1_PUSCH_MMSE7_PRE is registered
BBU Pooling: taskId = 11 taskName = UL_L1_PUSCH_MMSE0 is registered
BBU Pooling: taskId = 12 taskName = UL_L1_PUSCH_MMSE7 is registered
BBU Pooling: taskId = 13 taskName =  UL_L1_PUSCH_LLR is registered
BBU Pooling: taskId = 14 taskName = UL_L1_PUSCH_DECODE is registered
BBU Pooling: taskId = 15 taskName =   UL_L1_PUSCH_TB is registered
BBU Pooling: taskId = 16 taskName =      UL_L1_PUCCH is registered
BBU Pooling: taskId = 17 taskName =      UL_L1_PRACH is registered
BBU Pooling: taskId = 18 taskName =        UL_L1_SRS is registered
BBU Pooling: taskId = 19 taskName =       DL_L1_POST is registered
BBU Pooling: taskId = 20 taskName =       UL_L1_POST is registered
BBU Pooling: next taskList of     DL_L1_CONFIG:    DL_L1_PDSCH_TB
DL_L1_PDSCH_RS_GEN    DL_L1_CONTROL_CHANNELS
BBU Pooling: next taskList of   DL_L1_PDSCH_TB:               N/A

BBU Pooling: next taskList of DL_L1_PDSCH_SCRAMBLER:  DL_L1_PDSCH_SYMBOL_TX
BBU Pooling: next taskList of DL_L1_PDSCH_SYMBOL_TX:        DL_L1_POST
BBU Pooling: next taskList of DL_L1_PDSCH_RS_GEN:  DL_L1_PDSCH_SYMBOL_TX
BBU Pooling: next taskList of DL_L1_CONTROL_CHANNELS:        DL_L1_POST
BBU Pooling: next taskList of     UL_L1_CONFIG:        UL_L1_POST
BBU Pooling: next taskList of  UL_L1_PUSCH_CE0:  UL_L1_PUSCH_MMSE0
UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of  UL_L1_PUSCH_CE7:  UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE0_PRE:  UL_L1_PUSCH_MMSE0
UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE7_PRE:  UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE0:   UL_L1_PUSCH_LLR
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE7:   UL_L1_PUSCH_LLR
BBU Pooling: next taskList of  UL_L1_PUSCH_LLR:  UL_L1_PUSCH_DECODE
BBU Pooling: next taskList of UL_L1_PUSCH_DECODE:               N/A

BBU Pooling: next taskList of   UL_L1_PUSCH_TB:        UL_L1_POST
BBU Pooling: next taskList of      UL_L1_PUCCH:        UL_L1_POST
BBU Pooling: next taskList of      UL_L1_PRACH:        UL_L1_POST
BBU Pooling: next taskList of        UL_L1_SRS:        UL_L1_POST
BBU Pooling: next taskList of       DL_L1_POST:               N/A

BBU Pooling: next taskList of       UL_L1_POST:               N/A

enter RtThread Launch
3 thread associated with queue 0:coreIdx 0 1 2
Leave RtThread Launch
launching Thread 0 Queue 0 uCoreIdx 0 CoreId 3 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 1 Queue 0 uCoreIdx 1 CoreId 5 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 2 Queue 0 uCoreIdx 2 CoreId 6 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

bbupool_core_main: the server's coreNum = 40, the nCore = 3,nRtCoreMask = 0x68, the
nFeIfCore = 0,nFeIfCoreMask = 0x0
bbupool_core_main pthread_setaffinity_np succeed: coreId = 0, result = 0
nr5g_gnb_mac2phy_api_proc_print_phy_init [0]:
    nCarrierIdx: 0
    nDMRSTypeAPos: 2
    nPhyCellId: 100
    nDLAbsFrePointA: 27968160
    nULAbsFrePointA: 27968160
    nDLBandwidth: 100
    nULBandwidth: 100
    nDLFftSize: 1024
    nULFftSize: 1024
    nSSBPwr: 0
    nSSBAbsFre: 0
    nSSBPeriod: 2
    nSSBSubcSpacing: 3
    nSSBSubcOffset: 0
    nSSBPrbOffset: 0
    nMIB[0]: 255
    nMIB[1]: 255
    nMIB[2]: 255
    nDLK0: 0
    nULK0: 0
    nSSBMask[0]: 63
    nSSBMask[1]: 0
    nNrOfTxAnt: 2
    nNrOfRxAnt: 2
    nNrOfDLPorts: 2
    nNrOfULPorts: 2
    nCarrierAggregationLevel: 0
    nFrameDuplexType: 1
    nSubcCommon: 3
    nTddPeriod: 5 (TDD)
    SlotConfig:
        Slot Sym 0 Sym 1 Sym 2 Sym 3 Sym 4 Sym 5 Sym 6 Sym 7 Sym 8 Sym 9 Sym10 Sym11
Sym12 Sym13
           0    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           1    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           2    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           3    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     GD     GD
UL     UL
           4    UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL
UL     UL

    nPrachConfIdx: 81
    nPrachSubcSpacing: 3
    nPrachZeroCorrConf: 2
    nPrachRestrictSet: 0
    nPrachRootSeqIdx: 0
    nPrachFreqStart: 0
    nPrachFdm: 1
    nPrachSsbRach: 0
    nPrachNrofRxRU: 2
    nCyclicPrefix: 0
    nGroupHopFlag: 0
    nSequenceHopFlag: 0
    nHoppingId: 0
read_table: File table/common/pss_table.bin of size 381 read_size: 381
read_table: File table/common/sss_table.bin of size 128016 read_size: 128016
read_table: File table/common/srs_zc_36_plus.bin of size 905916 read_size: 905916
read_table: File table/common/pucch_zc_36_plus.bin of size 383040 read_size: 383040
read_table: File table/common/srs_wiener_sinc_comb2.bin of size 81216 read_size:
81216
read_table: File table/common/srs_wiener_sinc_comb4.bin of size 81216 read_size:
81216
BBU Pooling Info: maximum period length was configured, preMaxSF = 8000, postMasSF =
8000
set_slot_type SlotPattern:
    Slot:       0    1    2    3    4
        0      DL   DL   DL   SP   UL
```

```
PHYDI-INIT[from 0] PhyInstance: 0

---------------------------------------------------------
Global Variables:
---------------------------------------------------------
gCarrierAggLevel:                     0
gCarrierAggLevelInit:                 1
gSupportedAVX2                        1
---------------------------------------------------------

Received MSG_TYPE_PHY_START_REQ: 0
Processing MSG_TYPE_PHY_START_REQ: 0

xran_max_frame 99
XRAN_UP_VF: 0x0000
XRAN_CP_VF: 0x0001
xran_timing_source_thread [CPU  4] [PID: 112582]
O-DU: thread_run start time: 04/27/20 20:20:33.000000010 UTC [125]
Start C-plane DL 25 us after TTI  [trigger on sym 3]
Start C-plane UL 55 us after TTI  [trigger on sym 7]
Start U-plane DL 50 us before OTA [offset  in sym -5]
Start U-plane UL 45 us OTA        [offset  in sym 6]
C-plane to U-plane delay 25 us after TTI
Start Sym timer 8928 ns
interval_us 125
PHYDI-START[from 0] PhyInstance: 0, Mode: 4, Count: 100040207, Period: 0,
NumSlotPerSfn: 80
gnb_start_xran: gxRANStarted[0] CC 1 Ant 4 AntElm 0
XRAN front haul xran_mm_init
xran_sector_get_instances [0]: CC 0 handle 0x7f6e397307c0
Handle: 0x1994ce00 Instance: 0x7f6e397307c0
gnb_start_xran [0]: CC 0 handle 0x7f6e397307c0
Sucess xran_mm_init Instance 0x7f6e397307c0
nSectorNum 1
ru_0_cc_0_idx_0: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 0] nNumberOfBuffers 2240
nBufferSize 5856
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 0] mb pool 0x2e817b900
ru_0_cc_0_idx_1: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 1] nNumberOfBuffers 35840
nBufferSize 24
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 1] mb pool 0x2e7266c40
ru_0_cc_0_idx_2: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 2] nNumberOfBuffers 2240
nBufferSize 48416
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 2] mb pool 0x2e5cb4600
ru_0_cc_0_idx_3: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 3] nNumberOfBuffers 2240
nBufferSize 5856
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 3] mb pool 0x2df2872c0
ru_0_cc_0_idx_4: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 4] nNumberOfBuffers 35840
nBufferSize 24
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 4] mb pool 0x2de372600
ru_0_cc_0_idx_5: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 5] nNumberOfBuffers 2240
nBufferSize 48416
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 5] mb pool 0x2dcdbffc0
ru_0_cc_0_idx_6: [ handle 0x7f6e397307c0 0 0 ] [nPoolIndex 6] nNumberOfBuffers 2240
nBufferSize 8192
CC:[ handle 0x7f6e397307c0 ru 0 cc_idx 0 ] [nPoolIndex 6] mb pool 0x2d6392c80
gnb_init_xran_cp
init xran successfully
-----------------------------------------------------------------------------
mem_mgr_display_size:
    Num Memory Alloc:             5,186
    Total Memory Size:    4,389,524,920
-----------------------------------------------------------------------------
```

```
BBU Pooling: enter multicell Activate!
BBU Pooling Info: bbupool rt thread start on CoreIdx 2 coreId 6 at 547270377116554 at
sf=0 with queue 0 successfully
BBU Pooling: active result: Q_id = 0,currenSf = 0, curCellNum = 0, activesfn = 4,
CellNumInActSfn = 1
BBU Pooling: multiCell Activate sucessfully!
BBU Pooling Info: bbupool rt thread start on CoreIdx 0 coreId 3 at 547270377104408 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 1 coreId 5 at 547270377117634 at
sf=0 with queue 0 successfully
phy_bbupool_rx_handler: PhyId[0] nSfIdx[4] frame,slot[0,5] gNumSlotPerSfn[80]
==== l1app Time: 5001 ms NumCarrier: 1 NumBbuCores: 3  rxPcktCnt: 93621 rachPcktCnt
46811 Total Proc Time: [ 62.00.. 98.39..209.00] usces====
==== [o-du][rx 619683 pps 123936 kbps 2621619][tx 1996407 pps 399281 kbps 9181862]
[on_time 619683 early 0 late 0 corrupt 0 pkt_dupl 16 Total 619683] IO Util: 79.61 %
```

7. To execute testmac with O-DU functionality according to O-RAN Fronthaul specification, enter:

```
[root@xran flexran] cd ./bin/nr5g/gnb/testmac
```

8. To execute test case type:

```
./l2.sh --testfile=./cascade_lake-sp/csxsp_mu1_100mhz_mmimo_hton_xran.cfg
```

where output corresponding to Test MAC:

```
[root@sc12-xran-sub6 testmac]# ./l2.sh --testfile=./cascade_lake-
sp/csxsp_mu1_100mhz_mmimo_hton_xran.cfg
kernel.sched_rt_runtime_us = -1
kernel.shmmax = 2147483648
kernel.shmall = 2147483648
Note: Forwarding request to 'systemctl disable irqbalance.service'.
start 5GNR Test MAC
=========================
5GNR Testmac Application
=========================
testmac_cfg_set_cfg_filename: Coult not find string 'cfgfile' in command line. Using
default File: testmac_cfg.xml
---------------------------
TestMacCfg.xml Version: 20.04
---------------------------
 --version=20.04
 --wls_dev_name=wls0
 --wlsMemorySize=0x3F600000
 --dpdkIovaMode=0
 --PhyStartMode=1
 --PhyStartPeriod=40
 --PhyStartCount=0
 --MlogSubframes=128
 --MlogCores=3
 --MlogSize=2048
 --wlsRxThread=1, 90, 0
 --systemThread=0, 0, 0
 --runThread=0, 89, 0
 --urllcThread=19, 90, 0

wls_dev_filename: wls0
sys_reg_signal_handler:[err] signal handler in NULL
sys_reg_signal_handler:[err] signal handler in NULL
timer_set_tsc_freq_from_clock: System clock (rdtsc) resolution 1596245684 [Hz]
                               Ticks per usec 1596
```

```
MLogOpen: filename(testmac-mlog.bin) mlogSubframes (128), mlogCores(3),
mlogSize(2048) mlog_mask (-1)
    mlogSubframes (128), mlogCores(3), mlogSize(2048)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1596250375 [Hz]
        Ticks per us 1596
    MLog Storage: 0x7f84cae86100 -> 0x7f84caf46920 [ 788512 bytes ]
    localMLogFreqReg: 1596. Storing: 1596
    Mlog Open successful
Calling rte_eal_init: testmac -c1 --proc-type=auto --file-prefix wls0 --iova-mode=pa
EAL: Detected 40 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Auto-detected process type: SECONDARY
EAL: Multi-process socket /var/run/dpdk/wls0/mp_socket_112640_1f1baf0a9b316
EAL: Selected IOVA mode 'PA'
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: PCI device 0000:19:00.0 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:19:00.1 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:1d:00.0 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:1d:00.1 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:21:00.0 on NUMA socket 0
EAL:   probe driver: 8086:158b net_i40e
EAL: PCI device 0000:21:00.1 on NUMA socket 0
EAL:   probe driver: 8086:158b net_i40e
EAL: PCI device 0000:21:02.0 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
EAL:   using IOMMU type 1 (Type 1)
EAL: PCI device 0000:21:02.1 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
EAL: PCI device 0000:21:0a.0 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
EAL:   0000:21:0a.0 cannot find TAILQ entry for PCI device!
EAL: Requested device 0000:21:0a.0 cannot be used
EAL: PCI device 0000:21:0a.1 on NUMA socket 0
EAL:   probe driver: 8086:154c net_i40e_vf
EAL:   0000:21:0a.1 cannot find TAILQ entry for PCI device!
EAL: Requested device 0000:21:0a.1 cannot be used
EAL: PCI device 0000:67:00.0 on NUMA socket 0
EAL:   probe driver: 8086:37d2 net_i40e
EAL: PCI device 0000:67:00.1 on NUMA socket 0
EAL:   probe driver: 8086:37d2 net_i40e
wls_lib: Open wls0 (DPDK memzone)
wls_lib: WLS_Open 0x2bf600000
wls_lib: link: 1 <-> 0
wls_lib: Mode 1
wls_lib: WLS shared management memzone: wls0
wls_lib: hugePageSize on the system is 1073741824
wls_lib: WLS_Alloc [1063256064] bytes
wls_lib: Connecting to remote peer ...
wls_lib: Connected to remote peer
wls_mac_create_mem_array: pMemArray[0xf3500f0] pMemArrayMemory[0x280000000]
totalSize[1063256064] nBlockSize[262144] numBlocks[4056]
WLS_EnqueueBlock [1]
WLS inited ok [383]
==================================================================================
======================
TESTMAC VERSION
```

```
=========================================================================================
======================
$Version: #DIRTY# $ (x86)
IMG-date: Apr 27 2020
IMG-time: 12:55:58
=========================================================================================
======================
=========================================================================================
======================
Testmac threads in application
=========================================================================================
======================
testmac_run_thread:        [PID: 112644] binding on [CPU  0] [PRIO: 89] [POLICY:  1]
wls_mac_rx_task:           [PID: 112643] binding on [CPU  1] [PRIO: 90] [POLICY:  1]
=========================================================================================
======================

testmac_set_phy_start: mode[1], period[40], count[0]

testmac_run_load_files:
Loading DL Config Files:
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/dl/testmac_dl_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/dl/testmac_dl_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/dl/testmac_dl_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/dl/testmac_dl_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/dl/testmac_dl_mu3_100mhz.cfg
Loading UL Config Files:
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/ul/testmac_ul_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/ul/testmac_ul_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/ul/testmac_ul_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/ul/testmac_ul_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/ul/testmac_ul_mu3_100mhz.cfg
Loading FD Config Files:
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu1_40mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/fd/testmac_fd_mu3_100mhz.cfg

TESTMAC DL TESTS:
    Numerology[0] Bandwidth[5]
         1001  1002  1003  1004  1005  1006  1007  1008
    Numerology[0] Bandwidth[10]
         1001  1002  1003  1004  1005  1006  1007  1008
    Numerology[0] Bandwidth[20]
```

```
          1001  1002  1003  1004  1005  1006  1007  1008
     Numerology[1] Bandwidth[100]
          1200  1201  1202  1203  1204  1205  1206  1207  1210  1211
          1212  1213  1214  1215  1216  1217  1218  1219  1220  1221
          1222  1223  1224  1225  1226  1227  1228  1229  1230  1241
          1242  1243  1244  1245  1250  1251  1252  1300  1301  1302
          1303  1304  1305  1402  1404  1408  1416  1500  1501  1502
          1503  1504  1505  1506  2213  2214  2215  2217  2218  2219
          2223  2224  2225  2227  2228  2229  2500  2501  2502  2503
          2504  3213  3214  3215  3217  3218  3219  3223  3224  3225
          3227  3228  3229
     Numerology[3] Bandwidth[100]
          1001  1002  1003  1005  1006  1007  1008  1009  1010  1011
          1012  1013  1014  1015  1016  1017  1018  1019  1030  1031
          1032  1033  2001  2002  2003  2030  2033  3001  3002  3003
          3030

TESTMAC UL TESTS:
     Numerology[0] Bandwidth[5]
          1001  1002  1003
     Numerology[0] Bandwidth[10]
          1001  1002
     Numerology[0] Bandwidth[20]
          1001  1002  1003  1004  1005  1006  1007  1008
     Numerology[1] Bandwidth[100]
          1010  1030  1031  1032  1033  1034  1035  1036  1037  1038
          1039  1040  1041  1042  1043  1070  1071  1072  1073  1074
          1080  1081  1082  1083  1084  1085  1086  1087  1091  1092
          1093  1094  1095  1096  1100  1101  1102  1103  1104  1105
          1106  1107  1108  1110  1111  1113  1114  1115  1116  1117
          1118  1119  1120  1121  1122  1123  1124  1130  1131  1132
          1133  1134  1135  1136  1137  1138  1139  1140  1141  1142
          1143  1150  1152  1153  1154  1155  1156  1157  1159  1160
          1161  1162  1163  1164  1165  1166  1167  1168  1169  1170
          1171  1172  1173  1200  1201  1202  1203  1204  1205  1206
          1207  1208  1209  1210  1211  1212  1213  1214  1215  1216
          1217  1218  1219  1220  1221  1222  1230  1231  1232  1233
          1234  1235  1236  1237  1402  1404  1408  1416  1420  1421
          1422  1423  1424  1425  1426  1427  1428  1429  1430  1431
          1432  1433  1434  1435  1436  1437  1438  1500  1503  1504
          1505  1506  1507  1508  1511  1512  1513  1514  1515  1516
          1540  1541  1542  1563  1564  1565  1566  1567  1568  1569
          1570  1571  1572  1573  1574  1600  1601  1602  1603  1604
          1605  1606  1607  1608  1609  1610  1611  1612  1613  1614
          1615  1616  1617  1618  1619  1620  1621  1622  1623  1624
          1625  1626  1627  1628  1629  1630  1631  1632  1633  1634
          1635  1636  1637  1638  1639  1640  1641  1642  1700  1701
          2236  2237  3236  3237
     Numerology[3] Bandwidth[100]
          1001  1002  1003  1004  1005  1006  1007  1010  1011  1012
          1013  1014  1015  1020  1021  1022  1023  1024  1025  1026
          1027  1028  1029  1030  1031  1032  1033  1034  1035  1036
          1037  1040  1041  1042  1043  1044  1045  1046  1050  1051
          1052  1053  1054  1059  1060  1061  1062  1063  1064  1065
          1066  1067  1070  1071  1073  1074  1081  1082  1083  1084
          1085  1086  2001  2002  2003  3001  3002  3003

TESTMAC FD TESTS:
     Numerology[0] Bandwidth[5]
          1001  6001  8001 10001 12001
     Numerology[0] Bandwidth[10]
          1001  2001  4001  6001  8001 10001 12001  1002  2002  4002
```

```
         6002  8002 10002 12002  1003
    Numerology[0] Bandwidth[20]
         1002  1004  1012  1014  1015  1016  1017  1018  1020  1021
         1022  1023  1024  1025  1030  1031  1032  1033  1200  1201
         1202  1206  1207  1208  1209  1210  1211  1212  1220  1221
         1222  1223  1224  1225  1226  1227  1228
    Numerology[1] Bandwidth[40]
         1001  1002  1003
    Numerology[1] Bandwidth[100]
         1001  1200  1201  1202  1203  1204  1205  1206  1207  1208
         1209  1210  1300  1301  1302  1303  1304  1305  1306  1307
         1308  1350  1351  1352  1353  1354  1355  1356  1357  1358
         1370  1371  1372  1373  1401  1402  1403  1404  1405  1406
         1411  1412  1490  1494  1500  1501  1502  1503  1504  1510
         1511  1512  1513  1514  1515  1520  1521  1522  1523  1524
         1525  1526  1527  1528  1529  1530  1531  1532  1540  1541
         1700  1701  1702  2520  2521  2522  2523  2524  2525  2526
         2527  2528  2529  2530  2531  2532  3524  3525  3526  3527
         3528  3529  3530  3531  3532  4524  4525  4526  4527  4528
         4529  4530  4531  4532
    Numerology[3] Bandwidth[100]
         1001  1002  1004  1005  1006  1007  1008  1009  1010  1011
         1012  1013  1014  1015  1061  1062  1063  1064  1065  1080
         1081  1082  2001  3001
    testmac_run_parse_file Parsing config file: ./cascade_lake-
sp/csxsp_mu1_100mhz_mmimo_hton_xran.cfg
testmac_set_phy_start: mode[4], period[0], count[100040200]
    Adding Test[1002]. NumCarr[1], Current Directory:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/
        Carrier[0]: ConfigFile: fd/mu3_100mhz/2/fd_testconfig_tst2.cfg
-----------------------------------------------------------------------------------
---
Running Test[1002]. NumCarr[1], Current Directory:
/home/turner/xran/master/npg_wireless-flexran_l1_5g_test/
Carrier[0]: ConfigFile: fd/mu3_100mhz/2/fd_testconfig_tst2.cfg
TESTMAC>welcome to application console

MLogRestart
MLogOpen: filename(testmac-mlog.bin) mlogSubframes (128), mlogCores(3),
mlogSize(2048) mlog_mask (-1)
    mlogSubframes (128), mlogCores(3), mlogSize(2048)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1596249901 [Hz]
        Ticks per us 1596
    MLog Storage: 0x7f84bc000900 -> 0x7f84bc0c1120 [ 788512 bytes ]
    localMLogFreqReg: 1596. Storing: 1596
    Mlog Open successful

testmac_mac2phy_set_num_cells: Setting Max Cells: 1
testmac_config_parse: test_num[1002] test_type[2] numcarrier[1]
host_config_set_int Error(nPrachSsbRach, 3): Out of range: [min(0), max(1)]
Queueing MSG_TYPE_PHY_UL_IQ_SAMPLES(0)
Received MSG_TYPE_PHY_UL_IQ_SAMPLES(0)
Queueing MSG_TYPE_PHY_CONFIG_REQ(0) and sending list
Received MSG_TYPE_PHY_CONFIG_RESP(0)
Queueing MSG_TYPE_PHY_START_REQ(0) and sending list
Received MSG_TYPE_PHY_START_RESP(0)
==== testmac Time: 5000 ms NumCarrier: 1 Total Proc Time: [  0.00..  4.11.. 14.00]
usces====
Core Utilization [Core: 1] [Util %:  2.97%]
==== testmac Time: 10000 ms NumCarrier: 1 Total Proc Time: [  2.00..  4.10.. 13.00]
usces====
Core Utilization [Core: 1] [Util %:  4.81%]
==== testmac Time: 15000 ms NumCarrier: 1 Total Proc Time: [  2.00..  4.10..  6.00]
usces====
```

A.4.2 Configure FlexRAN 5G NR L1 Application for multiple O-RUs with multiple numerologies

The 5G NR layer 1 application can be used for executing the scenario for multiple cells with multiple numerologies. The current release supports the constant configuration of different numerologies on different O-RU ports. It is required that the first O-RU (O-RU0) to be configured with highest numerology. The configuration procedure is similar as described in above section. Please refer to the configuration file located in
bin\nr5g\gnb\l1\orancfg\sub3_mu0_20mhz_sub6_mu1_100mhz_4x4\gnb\xrancfg_sub6_oru.xml

A.4.3 Install and Configure FlexRAN 5G NR L1 Application for Massive - MIMO

The 5G NR layer 1 application can be used for executing the scenario for Massive-MIMO with either the RU sample application or just the O-DU side. 3 cells scenario with 64T64R Massive MIMO is targeted for Icelake system with Columbiavile NIC. The current release supports the constant configuration of the slot pattern and RB allocation on the PHY side. Please follow the general build process in the FlexRAN 5G NR Reference Solution L1 User Guide (refer to Table 2.)

1. xRAN library is enabled by default l1 application
2. Build the 5G NR L1 application using the command:
```
./flexran_build.sh -r 5gnr_l1app_sub6 -i snc -m sdk -m fb -m mlog -m wls -m
5gnr_l1app_mmw  -m xran -m 5gnr_testmac
```
3. Configure the L1app using `bin/nr5g/gnb/l1/xrancfg_sub6_mmimo.xml`.
```
<XranConfig>
    <version>20.08</version>
    <!-- numbers of O-RU connected to O-DU. All O-RUs are the same capabilities. Max
O-RUs is per XRAN_PORTS_NUM i.e. 4 -->
    <oRuNum>3</oRuNum>
    <!--  # 10G,25G,40G,100G speed of Physical connection on O-RU -->
    <oRuEthLinkSpeed>25</oRuEthLinkSpeed>
    <!--  # 1, 2, 3 total number of links per O-RU (Fronthaul Ethernet link in IOT
spec) -->
    <oRuLinesNumber>2</oRuLinesNumber>
    <!--  (1) - C- plane and U-plane on the same set of VFs. (0) - C-plane and U-
Plane use dedicated VFs -->
    <oRuCUon1Vf>1</oRuCUon1Vf>

    <!-- O-RU 0 -->
    <PciBusAddoRu0Vf0>0000:51:01.0</PciBusAddoRu0Vf0>
    <PciBusAddoRu0Vf1>0000:51:01.1</PciBusAddoRu0Vf1>
    <PciBusAddoRu0Vf2>0000:51:01.2</PciBusAddoRu0Vf2>
    <PciBusAddoRu0Vf3>0000:51:01.3</PciBusAddoRu0Vf3>

    <!-- O-RU 1 -->
    <PciBusAddoRu1Vf0>0000:51:01.2</PciBusAddoRu1Vf0>
    <PciBusAddoRu1Vf1>0000:51:01.3</PciBusAddoRu1Vf1>
    <PciBusAddoRu1Vf2>0000:51:01.6</PciBusAddoRu1Vf2>
    <PciBusAddoRu1Vf3>0000:51:01.7</PciBusAddoRu1Vf3>
    <!-- O-RU 2 -->
    <PciBusAddoRu2Vf0>0000:51:01.4</PciBusAddoRu2Vf0>
    <PciBusAddoRu2Vf1>0000:51:01.5</PciBusAddoRu2Vf1>
    <PciBusAddoRu2Vf2>0000:51:02.2</PciBusAddoRu2Vf2>
    <PciBusAddoRu2Vf3>0000:51:02.3</PciBusAddoRu2Vf3>

    <!-- O-RU 4 -->
    <PciBusAddoRu3Vf0>0000:00:00.0</PciBusAddoRu3Vf0>
    <PciBusAddoRu3Vf1>0000:00:00.0</PciBusAddoRu3Vf1>
    <PciBusAddoRu3Vf2>0000:00:00.0</PciBusAddoRu3Vf2>
    <PciBusAddoRu3Vf3>0000:00:00.0</PciBusAddoRu3Vf3>

    <!-- remote O-RU 0 Eth Link 0 VF0, VF1-->
    <oRuRem0Mac0>00:11:22:33:00:01<oRuRem0Mac0>
    <oRuRem0Mac1>00:11:22:33:00:11<oRuRem0Mac1>
    <!-- remote O-RU 0 Eth Link 1 VF2, VF3 -->
    <oRuRem0Mac2>00:11:22:33:00:21<oRuRem0Mac2>
    <oRuRem0Mac3>00:11:22:33:00:31<oRuRem0Mac3>

    <!-- remote O-RU 1 Eth Link 0 VF4, VF5-->
    <oRuRem1Mac0>00:11:22:33:01:01<oRuRem1Mac0>
    <oRuRem1Mac1>00:11:22:33:01:11<oRuRem1Mac1>
    <!-- remote O-RU 1 Eth Link 1 VF6, VF7 -->
    <oRuRem1Mac2>00:11:22:33:01:21<oRuRem1Mac2>
    <oRuRem1Mac3>00:11:22:33:01:31<oRuRem1Mac3>

    <!-- remote O-RU 2 Eth Link 0 VF8, VF9 -->
    <oRuRem2Mac0>00:11:22:33:02:01<oRuRem2Mac0>
    <oRuRem2Mac1>00:11:22:33:02:11<oRuRem2Mac1>
    <!-- remote O-RU 2 Eth Link 1 VF10, VF11-->
    <oRuRem2Mac2>00:11:22:33:02:21<oRuRem2Mac2>
    <oRuRem2Mac3>00:11:22:33:02:31<oRuRem2Mac3>

    <!-- remote O-RU 2 Eth Link 0 VF12, VF13 -->
    <oRuRem3Mac0>00:11:22:33:03:01<oRuRem3Mac0>
    <oRuRem3Mac1>00:11:22:33:03:11<oRuRem3Mac1>
    <!-- remote O-RU 2 Eth Link 1 VF14, VF15-->
    <oRuRem3Mac2>00:11:22:33:03:21<oRuRem3Mac2>
    <oRuRem3Mac3>00:11:22:33:03:31<oRuRem3Mac3>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu0NumCc>1</oRu0NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu0Cc0PhyId>0</oRu0Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu0Cc1PhyId>1</oRu0Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu0Cc2PhyId>2</oRu0Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu0Cc3PhyId>3</oRu0Cc3PhyId>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu1NumCc>1</oRu1NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu1Cc0PhyId>1</oRu1Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu1Cc1PhyId>1</oRu1Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu1Cc2PhyId>2</oRu1Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu1Cc3PhyId>3</oRu1Cc3PhyId>

    <!--  Number of cells (CCs) running on this O-RU  [1 - Cell , 2 - Cells, 3 -
Cells , 4 - Cells ] -->
    <oRu2NumCc>1</oRu2NumCc>
    <!-- First Phy instance ID mapped to this O-RU CC0  -->
    <oRu2Cc0PhyId>2</oRu2Cc0PhyId>
    <!-- Second Phy instance ID mapped to this O-RU CC1 -->
    <oRu2Cc1PhyId>1</oRu2Cc1PhyId>
    <!-- Third Phy instance ID mapped to this O-RU CC2  -->
    <oRu2Cc2PhyId>2</oRu2Cc2PhyId>
    <!-- Forth Phy instance ID mapped to this O-RU CC3  -->
    <oRu2Cc3PhyId>3</oRu2Cc3PhyId>

    <!-- XRAN Thread (core where the XRAN polling function is pinned: Core, priority,
Policy [0: SCHED_FIFO 1: SCHED_RR] -->
    <xRANThread>22, 96, 0</xRANThread>

    <!-- core mask for XRAN Packets Worker (core where the XRAN packet processing is
pinned): Core, priority, Policy [0: SCHED_FIFO 1: SCHED_RR] -->
    <xRANWorker>0x3800000, 96, 0</xRANWorker>
    <!-- XRAN: Category of O-RU 0 - Category A, 1 - Category B -->
    <Category>1</Category>

    <!-- XRAN: enable sleep on PMD cores -->
    <xranPmdSleep>0</xranPmdSleep>

    <!-- RU Settings -->
    <Tadv_cp_dl>25</Tadv_cp_dl>
    <!-- Reception Window C-plane DL-->
    <T2a_min_cp_dl>285</T2a_min_cp_dl>
    <T2a_max_cp_dl>429</T2a_max_cp_dl>
    <!-- Reception Window C-plane UL-->
    <T2a_min_cp_ul>285</T2a_min_cp_ul>
    <T2a_max_cp_ul>429</T2a_max_cp_ul>
    <!-- Reception Window U-plane -->
    <T2a_min_up>71</T2a_min_up>
    <T2a_max_up>428</T2a_max_up>
    <!-- Transmission Window U-plane -->
    <Ta3_min>20</Ta3_min>
    <Ta3_max>32</Ta3_max>

    <!-- O-DU Settings -->
    <!-- MTU size -->
    <MTU>9600</MTU>
    <!-- VLAN Tag used for C-Plane -->
    <c_plane_vlan_tag>1</c_plane_vlan_tag>
    <u_plane_vlan_tag>2</u_plane_vlan_tag>

    <!-- Transmission Window Fast C-plane DL -->
    <T1a_min_cp_dl>258</T1a_min_cp_dl>
    <T1a_max_cp_dl>429</T1a_max_cp_dl>
    <!-- Transmission Window Fast C-plane UL -->
    <T1a_min_cp_ul>285</T1a_min_cp_ul>
    <T1a_max_cp_ul>300</T1a_max_cp_ul>
    <!-- Transmission Window U-plane -->
    <T1a_min_up>96</T1a_min_up>
    <T1a_max_up>196</T1a_max_up>
    <!-- Reception Window U-Plane-->
    <Ta4_min>0</Ta4_min>
    <Ta4_max>75</Ta4_max>
    <!-- Enable Control Plane -->
    <EnableCp>1</EnableCp>

    <DynamicSectionEna>0</DynamicSectionEna>
    <!-- Enable Dynamic section allocation for UL -->
    <DynamicSectionEnaUL>0</DynamicSectionEnaUL>
    <xRANSFNWrap>1</xRANSFNWrap>
    <!-- Total Number of DL PRBs per symbol (starting from RB 0) that is transmitted
(used for testing. If 0, then value is used from PHY_CONFIG_API) -->
    <xRANNumDLPRBs>0</xRANNumDLPRBs>
    <!-- Total Number of UL PRBs per symbol (starting from RB 0) that is received
(used for testing. If 0, then value is used from PHY_CONFIG_API) -->
    <xRANNumULPRBs>0</xRANNumULPRBs>
    <!-- refer to alpha as defined in section 9.7.2 of O-RAN spec. this value should
be alpha*(1/1.2288ns), range 0 - 1e7 (ns) -->
    <Gps_Alpha>0</Gps_Alpha>
    <!-- beta value as defined in section 9.7.2 of O-RAN spec. range -32767 ~ +32767
-->
    <Gps_Beta>0</Gps_Beta>

    <!-- XRAN: Compression mode on O-DU <-> O-RU 0 - no comp 1 - BFP -->
    <xranCompMethod>1</xranCompMethod>

    <oRu0nPrbElemDl>6</oRu0nPrbElemDl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu0PrbElemDl0>0,48,0,14,1,1,1,9,1,0,0</oRu0PrbElemDl0>
    <oRu0PrbElemDl1>48,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemDl1>
    <oRu0PrbElemDl2>96,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemDl2>
    <oRu0PrbElemDl3>144,48,0,14,4,1,1,9,1,0,0</oRu0PrbElemDl3>
    <oRu0PrbElemDl4>192,48,0,14,5,1,1,9,1,0,0</oRu0PrbElemDl4>
    <oRu0PrbElemDl5>240,33,0,14,6,1,1,9,1,0,0</oRu0PrbElemDl5>
    <oRu0PrbElemDl6>240,33,0,14,7,1,1,9,1,0,0</oRu0PrbElemDl6>
    <oRu0PrbElemDl7>252,21,0,14,8,1,1,9,1,0,0</oRu0PrbElemDl7>

    <!-- extType = 11 -->
    <oRu0ExtBfwDl0>2,24,0,0,9,1</oRu0ExtBfwDl0>
    <oRu0ExtBfwDl1>2,24,0,0,9,1</oRu0ExtBfwDl1>
    <oRu0ExtBfwDl2>2,24,0,0,9,1</oRu0ExtBfwDl2>
    <oRu0ExtBfwDl3>2,24,0,0,9,1</oRu0ExtBfwDl3>
    <oRu0ExtBfwDl4>2,24,0,0,9,1</oRu0ExtBfwDl4>
    <oRu0ExtBfwDl5>2,17,0,0,9,1</oRu0ExtBfwDl5>

    <oRu0nPrbElemUl>6</oRu0nPrbElemUl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu0PrbElemUl0>0,48,0,14,1,1,1,9,1,0,0</oRu0PrbElemUl0>
    <oRu0PrbElemUl1>48,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemUl1>
    <oRu0PrbElemUl2>96,48,0,14,2,1,1,9,1,0,0</oRu0PrbElemUl2>
    <oRu0PrbElemUl3>144,48,0,14,4,1,1,9,1,0,0</oRu0PrbElemUl3>
    <oRu0PrbElemUl4>192,48,0,14,5,1,1,9,1,0,0</oRu0PrbElemUl4>
    <oRu0PrbElemUl5>240,33,0,14,6,1,1,9,1,0,0</oRu0PrbElemUl5>
    <oRu0PrbElemUl6>240,33,0,14,7,1,1,9,1,0,0</oRu0PrbElemUl6>
    <oRu0PrbElemUl7>252,21,0,14,8,1,1,9,1,0,0</oRu0PrbElemUl7>

    <!-- extType = 11 -->
    <oRu0ExtBfwUl0>2,24,0,0,9,1</oRu0ExtBfwUl0>
    <oRu0ExtBfwUl1>2,24,0,0,9,1</oRu0ExtBfwUl1>
    <oRu0ExtBfwUl2>2,24,0,0,9,1</oRu0ExtBfwUl2>
    <oRu0ExtBfwUl3>2,24,0,0,9,1</oRu0ExtBfwUl3>
    <oRu0ExtBfwUl4>2,24,0,0,9,1</oRu0ExtBfwUl4>
    <oRu0ExtBfwUl5>2,17,0,0,9,1</oRu0ExtBfwUl5>

    <oRu0nPrbElemSrs>1</oRu0nPrbElemSrs>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update, 
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu0PrbElemSrs0>0,273,0,14,1,1,1,9,1,0,0</oRu0PrbElemSrs0>

    <oRu1nPrbElemDl>2</oRu1nPrbElemDl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update, 
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu1PrbElemDl0>0,48,0,14,0,1,1,9,1,0,0</oRu1PrbElemDl0>
    <oRu1PrbElemDl1>48,48,0,14,2,1,1,9,1,0,0</oRu1PrbElemDl1>
    <oRu1PrbElemDl2>96,48,0,14,3,1,1,9,1,0,0</oRu1PrbElemDl2>
    <oRu1PrbElemDl3>144,48,0,14,4,1,1,9,1,0,0</oRu1PrbElemDl3>
    <oRu1PrbElemDl4>144,36,0,14,5,1,1,9,1,0,0</oRu1PrbElemDl4>
    <oRu1PrbElemDl5>180,36,0,14,6,1,1,9,1,0,0</oRu1PrbElemDl5>
    <oRu1PrbElemDl6>216,36,0,14,7,1,1,9,1,0,0</oRu1PrbElemDl6>
    <oRu1PrbElemDl7>252,21,0,14,8,1,1,9,1,0,0</oRu1PrbElemDl7>

    <!-- extType = 11 -->
    <oRu1ExtBfwDl0>2,24,0,0,9,1</oRu1ExtBfwDl0>
    <oRu1ExtBfwDl1>2,24,0,0,9,1</oRu1ExtBfwDl1>

    <oRu1nPrbElemUl>2</oRu1nPrbElemUl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update, 
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu1PrbElemUl0>0,48,0,14,1,1,1,9,1,0,0</oRu1PrbElemUl0>
    <oRu1PrbElemUl1>48,48,0,14,2,1,1,9,1,0,0</oRu1PrbElemUl1>
    <oRu1PrbElemUl2>72,36,0,14,3,1,1,9,1,0,0</oRu1PrbElemUl2>
    <oRu1PrbElemUl3>108,36,0,14,4,1,1,9,1,0,0</oRu1PrbElemUl3>
    <oRu1PrbElemUl4>144,36,0,14,5,1,1,9,1,0,0</oRu1PrbElemUl4>
    <oRu1PrbElemUl5>180,36,0,14,6,1,1,9,1,0,0</oRu1PrbElemUl5>
    <oRu1PrbElemUl6>216,36,0,14,7,1,1,9,1,0,0</oRu1PrbElemUl6>
    <oRu1PrbElemUl7>252,21,0,14,8,1,1,9,1,0,0</oRu1PrbElemUl7>

    <!-- extType = 11 -->
    <oRu1ExtBfwUl0>2,24,0,0,9,1</oRu1ExtBfwUl0>
    <oRu1ExtBfwUl1>2,24,0,0,9,1</oRu1ExtBfwUl1>

    <oRu1nPrbElemSrs>1</oRu1nPrbElemSrs>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update, 
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu1PrbElemSrs0>0,273,0,14,1,1,1,9,1,0,0</oRu1PrbElemSrs0>

    <oRu2nPrbElemDl>2</oRu2nPrbElemDl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update, 
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu2PrbElemDl0>0,48,0,14,1,1,1,9,1,0,0</oRu2PrbElemDl0>
    <oRu2PrbElemDl1>48,48,0,14,2,1,1,9,1,0,0</oRu2PrbElemDl1>
    <oRu2PrbElemDl2>96,48,0,14,3,1,1,9,1,0,0</oRu2PrbElemDl2>
    <oRu2PrbElemDl3>144,48,0,14,4,1,1,9,1,0,0</oRu2PrbElemDl3>
    <oRu2PrbElemDl4>144,36,0,14,5,1,1,9,1,0,0</oRu2PrbElemDl4>
    <oRu2PrbElemDl5>180,36,0,14,6,1,1,9,1,0,0</oRu2PrbElemDl5>
    <oRu2PrbElemDl6>216,36,0,14,7,1,1,9,1,0,0</oRu2PrbElemDl6>
    <oRu2PrbElemDl7>252,21,0,14,8,1,1,9,1,0,0</oRu2PrbElemDl7>
    <!-- extType = 11 -->
    <oRu2ExtBfwDl0>2,24,0,0,9,1</oRu2ExtBfwDl0>
    <oRu2ExtBfwDl1>2,24,0,0,9,1</oRu2ExtBfwDl1>

    <oRu2nPrbElemUl>2</oRu2nPrbElemUl>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu2PrbElemUl0>0,48,0,14,1,1,1,9,1,0,0</oRu2PrbElemUl0>
    <oRu2PrbElemUl1>48,48,0,14,2,1,1,9,1,0,0</oRu2PrbElemUl1>
    <oRu2PrbElemUl2>72,36,0,14,3,1,1,9,1,0,0</oRu2PrbElemUl2>
    <oRu2PrbElemUl3>108,36,0,14,4,1,1,9,1,0,0</oRu2PrbElemUl3>
    <oRu2PrbElemUl4>144,36,0,14,5,1,1,9,1,0,0</oRu2PrbElemUl4>
    <oRu2PrbElemUl5>180,36,0,14,6,1,1,9,1,0,0</oRu2PrbElemUl5>
    <oRu2PrbElemUl6>216,36,0,14,7,1,1,9,1,0,0</oRu2PrbElemUl6>
    <oRu2PrbElemUl7>252,21,0,14,8,1,1,9,1,0,0</oRu2PrbElemUl7>

    <!-- extType = 11 -->
    <oRu2ExtBfwUl0>2,24,0,0,9,1</oRu2ExtBfwUl0>
    <oRu2ExtBfwUl1>2,24,0,0,9,1</oRu2ExtBfwUl1>

    <oRu2nPrbElemSrs>1</oRu2nPrbElemSrs>
    <!--nRBStart, nRBSize, nStartSymb, numSymb, nBeamIndex, bf_weight_update,
compMethod, iqWidth, BeamFormingType, Scalefactor, REMask -->
    <!-- weight base beams -->
    <oRu2PrbElemSrs0>0,273,0,14,1,1,1,9,1,0,0</oRu2PrbElemSrs0>

</XranConfig>
```

4. Modify `./bin/nr5g/gnb/l1/dpdk.sh` (change PCIe addresses from VFs).

```
ethDevice0=0000:51:01.0
ethDevice1=0000:51:01.1
ethDevice2=0000:51:01.2
ethDevice3=0000:51:01.3
ethDevice4=0000:51:01.4
ethDevice5=0000:51:01.5
ethDevice6=
ethDevice7=
ethDevice8=
ethDevice9=
ethDevice10=
ethDevice11=
fecDevice0=0000:92:00.0
```

5. Use configuration of test mac per:

```
/bin/nr5g/gnb/testmac/icelake-sp/icxsp_mu1_100mhz_mmimo_64x64_hton_xran.cfg
phystart 4 0 100200
TEST_FD, 3370, 3, fd/mu1_100mhz/376/fd_testconfig_tst376.cfg,
fd/mu1_100mhz/377/fd_testconfig_tst377.cfg,
fd/mu1_100mhz/377/fd_testconfig_tst377.cfg
```

6. To execute l1app with O-DU functionality according to O-RAN Fronthaul specification, enter:

```
[root@xran flexran] cd ./bin/nr5g/gnb/l1
 ./l1.sh -xranmmimo
Radio mode with XRAN - Sub6 100Mhz Massive-MIMO (CatB)
DPDK WLS MODE
kernel.sched_rt_runtime_us = -1
kernel.shmmax = 2147483648
kernel.shmall = 2147483648
Note: Forwarding request to 'systemctl disable irqbalance.service'.
using configuration file phycfg_xran.xml
using configuration file xrancfg_sub6_mmimo.xml
```

```
>> Running... ./l1app table 0 1 --cfgfile=phycfg_xran.xml --
xranfile=xrancfg_sub6_mmimo.xml
FlexRAN SDK bblib_layerdemapping_5gnr version #DIRTY#
FlexRAN SDK bblib_layermapping_5gnr version #DIRTY#
FlexRAN SDK bblib_cestimate_5gnr_version #DIRTY#
FlexRAN SDK bblib_pucch_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_llr_demapping version #DIRTY#
FlexRAN SDK bblib_pdcch_remapping_5gnr_version version #DIRTY#
FlexRAN SDK bblib_reed_muller version #DIRTY#
FlexRAN SDK bblib_lte_modulation version #DIRTY#
FlexRAN SDK bblib_polar_decoder_5gnr version #DIRTY#
FlexRAN SDK bblib_polar_rate_dematching_5gnr version #DIRTY#
FlexRAN SDK bblib_PhaseNoise_5G version #DIRTY#
FlexRAN SDK bblib_mimo_mmse_detection_5gnr version #DIRTY#
FlexRAN SDK bblib_fd_correlation version #DIRTY#
FlexRAN SDK bblib_scramble_5gnr version #DIRTY#
FlexRAN SDK bblib_pucch_equ_5gnr version #DIRTY#
FlexRAN SDK bblib_ta_compensation_version_5gnr #DIRTY#
FlexRAN SDK bblib_polar_encoder_5gnr version #DIRTY#
FlexRAN SDK bblib_prach_5gnr version #DIRTY#
FlexRAN SDK bblib_fft_ifft version #DIRTY#
FlexRAN SDK bblib_pucch_5gnr version #DIRTY#
FlexRAN SDK bblib_lte_ldpc_decoder version #DIRTY#
FlexRAN SDK bblib_lte_ldpc_encoder version #DIRTY#
FlexRAN SDK bblib_lte_LDPC_ratematch version #DIRTY#
FlexRAN SDK bblib_lte_rate_dematching_5gnr version #DIRTY#
FlexRAN SDK bblib_common version #DIRTY#
FlexRAN SDK bblib_lte_crc version #DIRTY#
FlexRAN SDK bblib_lte_dft_idft version #DIRTY#
FlexRAN SDK bblib_irc_rnn_calculation_5gnr_version #DIRTY#
FlexRAN SDK bblib_mmse_irc_mimo_5gnr_version #DIRTY#
FlexRAN SDK bblib_srs_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_zf_matrix_gen version #DIRTY#
FlexRAN SDK bblib_beamforming_dl_expand version #DIRTY#
=========================
5GNR PHY Application
=========================


--------------------------------------------------------
File[phycfg_xran.xml] Version: 20.08
--------------------------------------------------------
 --version=20.08
 --successiveNoApi=15
 --wls_dev_name=wls0
 --wlsMemorySize=0x3F600000
 --dlIqLog=0
 --ulIqLog=0
 --iqLogDumpToFile=0
 --phyMlog=1
 --phyStats=1
 --dpdkMemorySize=18432
 --dpdkIovaMode=0
 --dpdkBasebandFecMode=1
 --dpdkBasebandDevice=0000:92:00.0
 --radioEnable=4
 --ferryBridgeMode=1
 --ferryBridgeEthPort=1
 --ferryBridgeSyncPorts=0
 --ferryBridgeOptCableLoopback=0
 --radioCfg0PCIeEthDev=0000:19:00.0
 --radioCfg0DpdkRx=1
```

```
--radioCfg0DpdkTx=2
--radioCfg0TxAnt=2
--radioCfg0RxAnt=2
--radioCfg0RxAgc=0
--radioCfg0NumCell=1
--radioCfg0Cell0PhyId=0
--radioCfg0Cell1PhyId=1
--radioCfg0Cell2PhyId=2
--radioCfg0Cell3PhyId=3
--radioCfg0Cell4PhyId=4
--radioCfg0Cell5PhyId=5
--radioCfg0riuMac=11:22:33:44:55:66
--radioCfg1PCIeEthDev=0000:03:00.1
--radioCfg1DpdkRx=1
--radioCfg1DpdkTx=1
--radioCfg1TxAnt=4
--radioCfg1RxAnt=4
--radioCfg1RxAgc=0
--radioCfg1NumCell=1
--radioCfg1Cell0PhyId=2
--radioCfg1Cell1PhyId=3
--radioCfg1Cell2PhyId=2
--radioCfg1Cell3PhyId=3
--radioCfg1riuMac=ac:1f:6b:2c:9f:07
--radioCfg2PCIeEthDev=0000:05:00.0
--radioCfg2DpdkRx=10
--radioCfg2DpdkTx=11
--radioCfg2TxAnt=4
--radioCfg2RxAnt=4
--radioCfg2RxAgc=0
--radioCfg2NumCell=2
--radioCfg2Cell0PhyId=4
--radioCfg2Cell1PhyId=5
--radioCfg2Cell2PhyId=2
--radioCfg2Cell3PhyId=3
--radioCfg2riuMac=ac:1f:6b:2c:9f:07
--radioCfg3PCIeEthDev=0000:05:00.1
--radioCfg3DpdkRx=12
--radioCfg3DpdkTx=13
--radioCfg3TxAnt=4
--radioCfg3RxAnt=4
--radioCfg3RxAgc=0
--radioCfg3NumCell=2
--radioCfg3Cell0PhyId=6
--radioCfg3Cell1PhyId=7
--radioCfg3Cell2PhyId=2
--radioCfg3Cell3PhyId=3
--radioCfg3riuMac=ac:1f:6b:2c:9f:07
--radioCfg4PCIeEthDev=0000:00:08.0
--radioCfg4DpdkRx=14
--radioCfg4DpdkTx=15
--radioCfg4TxAnt=4
--radioCfg4RxAnt=4
--radioCfg4RxAgc=0
--radioCfg4NumCell=2
--radioCfg4Cell0PhyId=8
--radioCfg4Cell1PhyId=9
--radioCfg4Cell2PhyId=2
--radioCfg4Cell3PhyId=3
--radioCfg4riuMac=ac:1f:6b:2c:9f:07
--radioCfg5PCIeEthDev=0000:08:00.0
--radioCfg5DpdkRx=16
```

```
--radioCfg5DpdkTx=16
--radioCfg5TxAnt=4
--radioCfg5RxAnt=4
--radioCfg5RxAgc=0
--radioCfg5NumCell=2
--radioCfg5Cell0PhyId=10
--radioCfg5Cell1PhyId=11
--radioCfg5Cell2PhyId=2
--radioCfg5Cell3PhyId=3
--radioCfg5riuMac=ac:1f:6b:2c:9f:07
--radioCfg6PCIeEthDev=0000:00:05.0
--radioCfg6DpdkRx=16
--radioCfg6DpdkTx=16
--radioCfg6TxAnt=4
--radioCfg6RxAnt=4
--radioCfg1RxAgc=0
--radioCfg6NumCell=2
--radioCfg6Cell0PhyId=12
--radioCfg6Cell1PhyId=13
--radioCfg6Cell2PhyId=2
--radioCfg6Cell3PhyId=3
--radioCfg6riuMac=ac:1f:6b:2c:9f:07
--radioCfg7PCIeEthDev=0000:00:06.0
--radioCfg7DpdkRx=16
--radioCfg7DpdkTx=16
--radioCfg7TxAnt=4
--radioCfg7RxAnt=4
--radioCfg7RxAgc=0
--radioCfg7NumCell=2
--radioCfg7Cell0PhyId=14
--radioCfg7Cell1PhyId=15
--radioCfg7Cell2PhyId=2
--radioCfg7Cell3PhyId=3
--radioCfg7riuMac=ac:1f:6b:2c:9f:07
--radioPort0=0
--radioPort1=1
--radioPort2=2
--radioPort3=3
--radioPort4=4
--radioPort5=5
--radioPort6=6
--radioPort7=7
--PdschSymbolSplit=0
--PdschDlWeightSplit=0
--FecEncSplit=4
--PuschChanEstSplit=0
--PuschMmseSplit=0
--PuschLlrRxSplit=0
--PuschUlWeightSplit=0
--FecDecEarlyTermDisable=0
--FecDecNumIter=12
--FecDecSplit=4
--llrOutDecimalDigit=2
--IrcEnableThreshold=-10
--PuschNoiseScale=2
--CEInterpMethod=0
--PucchSplit=0
--SrsCeSplit=0
--prachDetectThreshold=0
--MlogSubframes=128
--MlogCores=40
--MlogSize=10000
```

```
 --systemThread=2, 0, 0
 --timerThread=0, 96, 0
 --FpgaDriverCpuInfo=3, 96, 0
 --FrontHaulCpuInfo=3, 96, 0
 --radioDpdkMaster=2, 99, 0
 --BbuPoolSleepEnable=1
 --BbuPoolThreadCorePriority=94
 --BbuPoolThreadCorePolicy=0
 --BbuPoolThreadDefault_0_63=0xF0
 --BbuPoolThreadDefault_64_127=0x0
 --BbuPoolThreadSrs_0_63=0x0
 --BbuPoolThreadSrs_64_127=0x0
 --BbuPoolThreadDlbeam_0_63=0x0
 --BbuPoolThreadDlbeam_64_127=0x0
 --BbuPoolThreadUrllc=0x100
 --FrontHaulTimeAdvance=7450
 --nEthPorts=462607
 --nPhaseCompFlag=0
 --nFecFpgaVersionMu3=0x20010900
 --nFecFpgaVersionMu0_1=0x0423D420
 --nFhFpgaVersionMu3=0x8001000F
 --nFhFpgaVersionMu0_1=0x90010008
 --StreamStats=0
 --StreamIp=10.255.83.5
 --StreamPort=4010

wls_dev_filename: wls0
phycfg_apply: Initialize Radio Interface with XRAN library
Setting FecEncSplit to 1 to run on HW accelerator
Setting FecDecSplit to 1 to run on HW accelerator



--------------------------------------------------------
File[xrancfg_sub6_mmimo.xml] Version: 20.08
--------------------------------------------------------
 --version=20.08
 --oRuNum=3
 --oRuEthLinkSpeed=25
 --oRuLinesNumber=2
 --oRuCUon1Vf=1
 --PciBusAddoRu0Vf0=0000:51:01.0
 --PciBusAddoRu0Vf1=0000:51:01.1
 --PciBusAddoRu0Vf2=0000:51:01.2
 --PciBusAddoRu0Vf3=0000:51:01.3
 --PciBusAddoRu1Vf0=0000:51:01.2
 --PciBusAddoRu1Vf1=0000:51:01.3
 --PciBusAddoRu1Vf2=0000:51:01.6
 --PciBusAddoRu1Vf3=0000:51:01.7
 --PciBusAddoRu2Vf0=0000:51:01.4
 --PciBusAddoRu2Vf1=0000:51:01.5
 --PciBusAddoRu2Vf2=0000:51:02.2
 --PciBusAddoRu2Vf3=0000:51:02.3
 --PciBusAddoRu3Vf0=0000:00:00.0
 --PciBusAddoRu3Vf1=0000:00:00.0
 --PciBusAddoRu3Vf2=0000:00:00.0
 --PciBusAddoRu3Vf3=0000:00:00.0
 --oRuRem0Mac0=00:11:22:33:00:01
 --oRuRem0Mac1=00:11:22:33:00:11
 --oRuRem0Mac2=00:11:22:33:00:21
 --oRuRem0Mac3=00:11:22:33:00:31
 --oRuRem1Mac0=00:11:22:33:01:01
```

```
--oRuRem1Mac1=00:11:22:33:01:11
--oRuRem1Mac2=00:11:22:33:01:21
--oRuRem1Mac3=00:11:22:33:01:31
--oRuRem2Mac0=00:11:22:33:02:01
--oRuRem2Mac1=00:11:22:33:02:11
--oRuRem2Mac2=00:11:22:33:02:21
--oRuRem2Mac3=00:11:22:33:02:31
--oRuRem3Mac0=00:11:22:33:03:01
--oRuRem3Mac1=00:11:22:33:03:11
--oRuRem3Mac2=00:11:22:33:03:21
--oRuRem3Mac3=00:11:22:33:03:31
--oRu0NumCc=1
--oRu0Cc0PhyId=0
--oRu0Cc1PhyId=1
--oRu0Cc2PhyId=2
--oRu0Cc3PhyId=3
--oRu1NumCc=1
--oRu1Cc0PhyId=1
--oRu1Cc1PhyId=1
--oRu1Cc2PhyId=2
--oRu1Cc3PhyId=3
--oRu2NumCc=1
--oRu2Cc0PhyId=2
--oRu2Cc1PhyId=1
--oRu2Cc2PhyId=2
--oRu2Cc3PhyId=3
--xRANThread=22, 96, 0
--xRANWorker=0x3800000, 96, 0
--Category=1
--xranPmdSleep=0
--Tadv_cp_dl=25
--T2a_min_cp_dl=285
--T2a_max_cp_dl=429
--T2a_min_cp_ul=285
--T2a_max_cp_ul=429
--T2a_min_up=71
--T2a_max_up=428
--Ta3_min=20
--Ta3_max=32
--MTU=9600
--c_plane_vlan_tag=1
--u_plane_vlan_tag=2
--T1a_min_cp_dl=258
--T1a_max_cp_dl=429
--T1a_min_cp_ul=285
--T1a_max_cp_ul=300
--T1a_min_up=96
--T1a_max_up=196
--Ta4_min=0
--Ta4_max=75
--EnableCp=1
--DynamicSectionEna=0
--DynamicSectionEnaUL=0
--xRANSFNWrap=1
--xRANNumDLPRBs=0
--xRANNumULPRBs=0
--Gps_Alpha=0
--Gps_Beta=0
--xranCompMethod=1
--oRu0nPrbElemDl=6
--oRu0PrbElemDl0=0,48,0,14,1,1,1,9,1,0,0
--oRu0PrbElemDl1=48,48,0,14,2,1,1,9,1,0,0
```

```
--oRu0PrbElemDl2=96,48,0,14,2,1,1,9,1,0,0
--oRu0PrbElemDl3=144,48,0,14,4,1,1,9,1,0,0
--oRu0PrbElemDl4=192,48,0,14,5,1,1,9,1,0,0
--oRu0PrbElemDl5=240,33,0,14,6,1,1,9,1,0,0
--oRu0PrbElemDl6=240,33,0,14,7,1,1,9,1,0,0
--oRu0PrbElemDl7=252,21,0,14,8,1,1,9,1,0,0
--oRu0ExtBfwDl0=2,24,0,0,9,1
--oRu0ExtBfwDl1=2,24,0,0,9,1
--oRu0ExtBfwDl2=2,24,0,0,9,1
--oRu0ExtBfwDl3=2,24,0,0,9,1
--oRu0ExtBfwDl4=2,24,0,0,9,1
--oRu0ExtBfwDl5=2,17,0,0,9,1
--oRu0nPrbElemUl=6
--oRu0PrbElemUl0=0,48,0,14,1,1,1,9,1,0,0
--oRu0PrbElemUl1=48,48,0,14,2,1,1,9,1,0,0
--oRu0PrbElemUl2=96,48,0,14,2,1,1,9,1,0,0
--oRu0PrbElemUl3=144,48,0,14,4,1,1,9,1,0,0
--oRu0PrbElemUl4=192,48,0,14,5,1,1,9,1,0,0
--oRu0PrbElemUl5=240,33,0,14,6,1,1,9,1,0,0
--oRu0PrbElemUl6=240,33,0,14,7,1,1,9,1,0,0
--oRu0PrbElemUl7=252,21,0,14,8,1,1,9,1,0,0
--oRu0ExtBfwUl0=2,24,0,0,9,1
--oRu0ExtBfwUl1=2,24,0,0,9,1
--oRu0ExtBfwUl2=2,24,0,0,9,1
--oRu0ExtBfwUl3=2,24,0,0,9,1
--oRu0ExtBfwUl4=2,24,0,0,9,1
--oRu0ExtBfwUl5=2,17,0,0,9,1
--oRu0nPrbElemSrs=1
--oRu0PrbElemSrs0=0,273,0,14,1,1,1,9,1,0,0
--oRu1nPrbElemDl=2
--oRu1PrbElemDl0=0,48,0,14,0,1,1,9,1,0,0
--oRu1PrbElemDl1=48,48,0,14,2,1,1,9,1,0,0
--oRu1PrbElemDl2=96,48,0,14,3,1,1,9,1,0,0
--oRu1PrbElemDl3=144,48,0,14,4,1,1,9,1,0,0
--oRu1PrbElemDl4=144,36,0,14,5,1,1,9,1,0,0
--oRu1PrbElemDl5=180,36,0,14,6,1,1,9,1,0,0
--oRu1PrbElemDl6=216,36,0,14,7,1,1,9,1,0,0
--oRu1PrbElemDl7=252,21,0,14,8,1,1,9,1,0,0
--oRu1ExtBfwDl0=2,24,0,0,9,1
--oRu1ExtBfwDl1=2,24,0,0,9,1
--oRu1nPrbElemUl=2
--oRu1PrbElemUl0=0,48,0,14,1,1,1,9,1,0,0
--oRu1PrbElemUl1=48,48,0,14,2,1,1,9,1,0,0
--oRu1PrbElemUl2=72,36,0,14,3,1,1,9,1,0,0
--oRu1PrbElemUl3=108,36,0,14,4,1,1,9,1,0,0
--oRu1PrbElemUl4=144,36,0,14,5,1,1,9,1,0,0
--oRu1PrbElemUl5=180,36,0,14,6,1,1,9,1,0,0
--oRu1PrbElemUl6=216,36,0,14,7,1,1,9,1,0,0
--oRu1PrbElemUl7=252,21,0,14,8,1,1,9,1,0,0
--oRu1ExtBfwUl0=2,24,0,0,9,1
--oRu1ExtBfwUl1=2,24,0,0,9,1
--oRu1nPrbElemSrs=1
--oRu1PrbElemSrs0=0,273,0,14,1,1,1,9,1,0,0
--oRu2nPrbElemDl=2
--oRu2PrbElemDl0=0,48,0,14,1,1,1,9,1,0,0
--oRu2PrbElemDl1=48,48,0,14,2,1,1,9,1,0,0
--oRu2PrbElemDl2=96,48,0,14,3,1,1,9,1,0,0
--oRu2PrbElemDl3=144,48,0,14,4,1,1,9,1,0,0
--oRu2PrbElemDl4=144,36,0,14,5,1,1,9,1,0,0
--oRu2PrbElemDl5=180,36,0,14,6,1,1,9,1,0,0
--oRu2PrbElemDl6=216,36,0,14,7,1,1,9,1,0,0
--oRu2PrbElemDl7=252,21,0,14,8,1,1,9,1,0,0
```

```
 --oRu2ExtBfwDl0=2,24,0,0,9,1
 --oRu2ExtBfwDl1=2,24,0,0,9,1
 --oRu2nPrbElemUl=2
 --oRu2PrbElemUl0=0,48,0,14,1,1,1,9,1,0,0
 --oRu2PrbElemUl1=48,48,0,14,2,1,1,9,1,0,0
 --oRu2PrbElemUl2=72,36,0,14,3,1,1,9,1,0,0
 --oRu2PrbElemUl3=108,36,0,14,4,1,1,9,1,0,0
 --oRu2PrbElemUl4=144,36,0,14,5,1,1,9,1,0,0
 --oRu2PrbElemUl5=180,36,0,14,6,1,1,9,1,0,0
 --oRu2PrbElemUl6=216,36,0,14,7,1,1,9,1,0,0
 --oRu2PrbElemUl7=252,21,0,14,8,1,1,9,1,0,0
 --oRu2ExtBfwUl0=2,24,0,0,9,1
 --oRu2ExtBfwUl1=2,24,0,0,9,1
 --oRu2nPrbElemSrs=1
 --oRu2PrbElemSrs0=0,273,0,14,1,1,1,9,1,0,0


timer_set_tsc_freq_from_clock: System clock (rdtsc) resolution 1496526035 [Hz]
                               Ticks per usec 1496
MLogOpen: filename(l1mlog.bin) mlogSubframes (128), mlogCores(40), mlogSize(10000)
mlog_mask (-1)
    mlogSubframes (128), mlogCores(40), mlogSize(10000)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1496525824 [Hz]
        Ticks per us 1496
    MLog Storage: 0x7f7403835100 -> 0x7f740690b830 [ 51210032 bytes ]
    localMLogFreqReg: 1496. Storing: 1496
    Mlog Open successful

gnb_io_xran_init
num_o_ru 3 EthLinesNumber 2 where VFs 1 per EthLine
VF[0] 0000:51:01.0 [C+U Plane]
VF[1] 0000:51:01.1 [C+U Plane]
VF[2] 0000:51:01.2 [C+U Plane]
VF[3] 0000:51:01.3 [C+U Plane]
VF[4] 0000:51:01.4 [C+U Plane]
VF[5] 0000:51:01.5 [C+U Plane]
oRu0nPrbElemDl0: oRu0: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemDl1: oRu0: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemDl2: oRu0: nRBStart 96,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):2 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemDl3: oRu0: nRBStart 144,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 4,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):3 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemDl4: oRu0: nRBStart 192,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 5,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):4 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemDl5: oRu0: nRBStart 240,nRBSize 33,nStartSymb 0,numSymb 14,nBeamIndex 6,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,17,0,0,9,1):5 numBundPrb 2, numSetBFW 17, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
```

```
oRu0nPrbElemUl0: oRu0: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemUl1: oRu0: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemUl2: oRu0: nRBStart 96,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):2 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemUl3: oRu0: nRBStart 144,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 4,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):3 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemUl4: oRu0: nRBStart 192,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 5,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):4 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemUl5: oRu0: nRBStart 240,nRBSize 33,nStartSymb 0,numSymb 14,nBeamIndex 6,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,17,0,0,9,1):5 numBundPrb 2, numSetBFW 17, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu0nPrbElemSrs0: oRu0: nRBStart 0,nRBSize 273,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
oRu1nPrbElemDl0: oRu1: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 0,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu1nPrbElemDl1: oRu1: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu1nPrbElemUl0: oRu1: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu1nPrbElemUl1: oRu1: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu1nPrbElemSrs0: oRu1: nRBStart 0,nRBSize 273,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
oRu2nPrbElemDl0: oRu2: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu2nPrbElemDl1: oRu2: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu2nPrbElemUl0: oRu2: nRBStart 0,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):0 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu2nPrbElemUl1: oRu2: nRBStart 48,nRBSize 48,nStartSymb 0,numSymb 14,nBeamIndex 2,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
(2,24,0,0,9,1):1 numBundPrb 2, numSetBFW 24, RAD 0, disableBFW 0, bfwIqWidth 9,
bfwCompMeth 1
oRu2nPrbElemSrs0: oRu2: nRBStart 0,nRBSize 273,nStartSymb 0,numSymb 14,nBeamIndex 1,
bf_weight_update 1 compMethod 1, iqWidth 9 BeamFormingType 1 scaler 0 remask 0x0
```

```
gnb_io_xran_cfg_setup successful
 xran_init: MTU 9600
 xran_init: MTU 9600
 xran_init: MTU 9600
PF Eth line speed 25G
PF Eth lines per O-xU port 2
BBDEV_FEC_ACCL_NR5G
hw-accelerated bbdev 0000:92:00.0
total cores 48 c_mask 0x3c00004 core 22 [id] system_core 2 [id] pkt_proc_core
0x3800000 [mask] pkt_aux_core 0 [id] timing_core 22 [id]
xran_ethdi_init_dpdk_io: Calling rte_eal_init:wls0 -c 0x3c00004 -n2 --iova-mode=pa --
socket-mem=18432 --socket-limit=18432 --proc-type=auto --file-prefix wls0 -w
0000:00:00.0 -w 0000:92:00.0
EAL: Detected 48 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Auto-detected process type: PRIMARY
EAL: Multi-process socket /var/run/dpdk/wls0/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available hugepages reported in hugepages-2048kB
EAL: Probing VFIO support...
EAL: PCI device 0000:92:00.0 on NUMA socket 0
EAL:   probe driver: 8086:d8f intel_fpga_5gnr_fec_pf
xran_init_mbuf_pool: socket 0
EAL: PCI device 0000:51:01.0 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 0 for TX, drv=net_iavf
Port 0 MAC: 00 11 22 33 00 00
Port 0: nb_rxd 4096 nb_txd 4096
[0] mempool_rx__0
[0] mempool_small__0
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [0]   ... done
Port 0 Link Up - speed 100000 Mbps - full-duplex
EAL: PCI device 0000:51:01.1 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 1 for TX, drv=net_iavf
Port 1 MAC: 00 11 22 33 00 10
Port 1: nb_rxd 4096 nb_txd 4096
[1] mempool_rx__1
[1] mempool_small__1
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [1]   ... done
Port 1 Link Up - speed 100000 Mbps - full-duplex
EAL: PCI device 0000:51:01.2 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 2 for TX, drv=net_iavf
Port 2 MAC: 00 11 22 33 01 00
Port 2: nb_rxd 4096 nb_txd 4096
[2] mempool_rx__2
[2] mempool_small__2
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [2]   ... done
Port 2 Link Up - speed 100000 Mbps - full-duplex
EAL: PCI device 0000:51:01.3 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 3 for TX, drv=net_iavf
Port 3 MAC: 00 11 22 33 01 10
Port 3: nb_rxd 4096 nb_txd 4096
[3] mempool_rx__3
```

```
[3] mempool_small__3
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [3]   ... done
Port 3 Link Up - speed 100000 Mbps - full-duplex
EAL: PCI device 0000:51:01.4 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 4 for TX, drv=net_iavf
Port 4 MAC: 00 11 22 33 02 00
Port 4: nb_rxd 4096 nb_txd 4096
[4] mempool_rx__4
[4] mempool_small__4
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [4]   ... done
Port 4 Link Up - speed 100000 Mbps - full-duplex
EAL: PCI device 0000:51:01.5 on NUMA socket 0
EAL:   probe driver: 8086:1889 net_iavf
initializing port 5 for TX, drv=net_iavf
Port 5 MAC: 00 11 22 33 02 10
Port 5: nb_rxd 4096 nb_txd 4096
[5] mempool_rx__5
[5] mempool_small__5
iavf_init_rss(): RSS is enabled by PF by default

Checking link status portid [5]   ... done
Port 5 Link Up - speed 100000 Mbps - full-duplex
[ 0] vf  0 local  SRC MAC: 00 11 22 33 00 00
[ 0] vf  0 remote DST MAC: 00 11 22 33 00 01
[ 0] vf  1 local  SRC MAC: 00 11 22 33 00 10
[ 0] vf  1 remote DST MAC: 00 11 22 33 00 11
[ 1] vf  2 local  SRC MAC: 00 11 22 33 01 00
[ 1] vf  2 remote DST MAC: 00 11 22 33 01 01
[ 1] vf  3 local  SRC MAC: 00 11 22 33 01 10
[ 1] vf  3 remote DST MAC: 00 11 22 33 01 11
[ 2] vf  4 local  SRC MAC: 00 11 22 33 02 00
[ 2] vf  4 remote DST MAC: 00 11 22 33 02 01
[ 2] vf  5 local  SRC MAC: 00 11 22 33 02 10
[ 2] vf  5 remote DST MAC: 00 11 22 33 02 11
created dl_gen_ring_up_0
created dl_gen_ring_up_1
created dl_gen_ring_up_2
xran_init successful, pHandle = 0x7f7393b23040


bbdev_init:
Socket ID: 0
FEC is accelerated through BBDEV:  0000:92:00.0
wls_layer_init[wls0] nWlsMemorySize[1063256064]
wls_lib: Open wls0 (DPDK memzone)
wls_lib: WLS_Open 0x43f600000
wls_lib: link: 0 <-> 1
wls_lib: Mode 0
wls_lib: WLS shared management memzone: wls0
wls_lib: hugePageSize on the system is 1073741824
wls_lib: WLS_Alloc [1063256064] bytes


==================================================================================
======================
PHY VERSION
```

```
=====================================================================================
======================
Version: #DIRTY#
IMG-date: Aug  5 2020
IMG-time: 18:31:18
=====================================================================================
======================
DEPENDENCIES VERSIONS
=====================================================================================
======================
FlexRAN BBU pooling version #DIRTY#
FlexRAN SDK bblib_layerdemapping_5gnr version #DIRTY#
FlexRAN SDK bblib_layermapping_5gnr version #DIRTY#
FlexRAN SDK bblib_cestimate_5gnr_version #DIRTY#
FlexRAN SDK bblib_pucch_cestimate_5gnr version #DIRTY#
FlexRAN SDK bblib_llr_demapping version #DIRTY#
FlexRAN SDK bblib_pdcch_remapping_5gnr_version version #DIRTY#
FlexRAN SDK bblib_reed_muller version #DIRTY#
FlexRAN SDK bblib_lte_modulation version #DIRTY#
FlexRAN SDK bblib_polar_decoder_5gnr version #DIRTY#
FlexRAN SDK bblib_polar_rate_dematching_5gnr version #DIRTY#
FlexRAN SDK bblib_PhaseNoise_5G version #DIRTY#
FlexRAN SDK bblib_mimo_mmse_detection_5gnr version #DIRTY#
FlexRAN SDK bblib_fd_correlation version #DIRTY#
FlexRAN SDK bblib_scramble_5gnr version #DIRTY#
FlexRAN SDK bblib_pucch_equ_5gnr version #DIRTY#
FlexRAN SDK bblib_ta_compensation_version_5gnr #DIRTY#
FlexRAN SDK bblib_polar_encoder_5gnr version #DIRTY#
FlexRAN SDK bblib_prach_5gnr version #DIRTY#
FlexRAN SDK bblib_fft_ifft version #DIRTY#
FlexRAN SDK bblib_pucch_5gnr version #DIRTY#
FlexRAN SDK bblib_lte_crc version #DIRTY#
FlexRAN SDK bblib_common version #DIRTY#
=====================================================================================
======================

=====================================================================================
======================
Non BBU threads in application
=====================================================================================
======================
nr5g_gnb_phy2mac_api_proc_stats_thread: [PID:  29438] binding on [CPU  2] [PRIO:  0]
[POLICY:  1]
wls_rx_handler (non-rt):                [PID:  29445] binding on [CPU  2]
=====================================================================================
======================

PHY>welcome to application console


PHY>
PHY>
PHY>Received MSG_TYPE_PHY_ADD_REMOVE_CORE
Processing MSG_TYPE_PHY_ADD_REMOVE_CORE
phy_bbupool_set_core[0] (add): 137170526192 [0x0000001ff0001ff0] Current: 0
[0x0000000000000000]
nr5g_gnb_mac2phy_api_set_options: PDSCH_SPLIT[4] nCellMask[0x00000001]
nr5g_gnb_mac2phy_api_set_options: PDSCH_DL_WEIGHT_SPLIT[4] nCellMask[0x00000001]
nr5g_gnb_mac2phy_api_set_options: PUSCH_CHANEST_SPLIT[2] nCellMask[0x00000001]
nr5g_gnb_mac2phy_api_set_options: PUSCH_MMSE_SPLIT[4] nCellMask[0x00000001]
nr5g_gnb_mac2phy_api_set_options: PUSCH_LLR_RX_SPLIT[2] nCellMask[0x00000001]
nr5g_gnb_mac2phy_api_set_options: PUSCH_UL_WEIGHT_SPLIT[2] nCellMask[0x00000001]
```

```
nr5g_gnb_mac2phy_api_set_options: FEC_DEC_NUM_ITER[3] nCellMask[0x00ffffff]
Received MSG_TYPE_PHY_UL_IQ_SAMPLES
Received MSG_TYPE_PHY_UL_IQ_SAMPLES
Received MSG_TYPE_PHY_UL_IQ_SAMPLES
Processing MSG_TYPE_PHY_UL_IQ_SAMPLES: 0
phydi_read_write_iq_samples: direction[1] nNumerologyMult[2] fftSize[4096, 45864,
SRS: 3276] numSubframe[20] numAntenna[64] numPorts[8] nIsRadioMode[1] carrNum[0]
TimerModeFreqDomain[1] PhaseCompensationEnable[0]
filename_in_ul_iq[/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/mu1_100mh
z/376/uliq00_tst376.bin] filename_in_prach_iq[]
Processing MSG_TYPE_PHY_UL_IQ_SAMPLES: 1
phydi_read_write_iq_samples: direction[1] nNumerologyMult[2] fftSize[4096, 45864,
SRS: 3276] numSubframe[20] numAntenna[64] numPorts[8] nIsRadioMode[1] carrNum[1]
TimerModeFreqDomain[1] PhaseCompensationEnable[0]
filename_in_ul_iq[/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/mu1_100mh
z/377/uliq00_tst377.bin] filename_in_prach_iq[]
Processing MSG_TYPE_PHY_UL_IQ_SAMPLES: 2
phydi_read_write_iq_samples: direction[1] nNumerologyMult[2] fftSize[4096, 45864,
SRS: 3276] numSubframe[20] numAntenna[64] numPorts[8] nIsRadioMode[1] carrNum[2]
TimerModeFreqDomain[1] PhaseCompensationEnable[0]
filename_in_ul_iq[/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/mu1_100mh
z/377/uliq00_tst377.bin] filename_in_prach_iq[]
Received MSG_TYPE_PHY_CONFIG_REQ: 0
Received MSG_TYPE_PHY_CONFIG_REQ: 1
Received MSG_TYPE_PHY_CONFIG_REQ: 2
Processing MSG_TYPE_PHY_CONFIG_REQ: 0
phy_bbupool_init: Changing Core Mask0 [0xf0] to [0x1ff0001ff0]
phy_bbupool_set_config: Using cores: 0x0000001ff0001ff0 for BBU Pool
nBbuPoolSleepEnable: 1
BBU Pooling: queueId = 0, the according nCoreNum = 18, the according cpuSetMask =
0x1ff0001ff0
BBU Pooling: gCoreIdxMap[0] = 4 is available!
BBU Pooling: gCoreIdxMap[1] = 5 is available!
BBU Pooling: gCoreIdxMap[2] = 6 is available!
BBU Pooling: gCoreIdxMap[3] = 7 is available!
BBU Pooling: gCoreIdxMap[4] = 8 is available!
BBU Pooling: gCoreIdxMap[5] = 9 is available!
BBU Pooling: gCoreIdxMap[6] = 10 is available!
BBU Pooling: gCoreIdxMap[7] = 11 is available!
BBU Pooling: gCoreIdxMap[8] = 12 is available!
BBU Pooling: gCoreIdxMap[9] = 28 is available!
BBU Pooling: gCoreIdxMap[10] = 29 is available!
BBU Pooling: gCoreIdxMap[11] = 30 is available!
BBU Pooling: gCoreIdxMap[12] = 31 is available!
BBU Pooling: gCoreIdxMap[13] = 32 is available!
BBU Pooling: gCoreIdxMap[14] = 33 is available!
BBU Pooling: gCoreIdxMap[15] = 34 is available!
BBU Pooling: gCoreIdxMap[16] = 35 is available!
BBU Pooling: gCoreIdxMap[17] = 36 is available!
phy_bbupool_init: Changing SrsCore Mask0 [(nil)] to [0x10000010]
phy_bbupool_init: Changing DlbeamCore Mask0 [(nil)] to [0x7e0]
Massive Mimo Config: nCarrierAggregationLevel[3],
nMassiveMimoSrsCoresMask[0x10000010] nTotalSrsCores[2]
   Setting aside core[4] for SRS
   Setting aside core[28] for SRS
Massive Mimo Config: nCarrierAggregationLevel[3], nMassiveMimoDlbeamCoresMask[0x7e0]
nTotalDlbeamCores[6]
   Setting aside core[5] for DL beam
   Setting aside core[6] for DL beam
   Setting aside core[7] for DL beam
   Setting aside core[8] for DL beam
   Setting aside core[9] for DL beam
   Setting aside core[10] for DL beam
BBU Pooling: taskId =  0 taskName =     DL_L1_CONFIG is registered
BBU Pooling: taskId =  1 taskName =   DL_L1_PDSCH_TB is registered
BBU Pooling: taskId =  2 taskName = DL_L1_PDSCH_SCRAMBLER is registered
BBU Pooling: taskId =  3 taskName = DL_L1_PDSCH_SYMBOL_TX is registered
BBU Pooling: taskId =  4 taskName = DL_L1_PDSCH_RS_GEN is registered
BBU Pooling: taskId =  5 taskName = DL_L1_CONTROL_CHANNELS is registered
BBU Pooling: taskId =  6 taskName =     UL_L1_CONFIG is registered
BBU Pooling: taskId =  7 taskName =  UL_L1_PUSCH_CE0 is registered
BBU Pooling: taskId =  8 taskName =  UL_L1_PUSCH_CE7 is registered
BBU Pooling: taskId =  9 taskName = UL_L1_PUSCH_MMSE0_PRE is registered
BBU Pooling: taskId = 10 taskName = UL_L1_PUSCH_MMSE7_PRE is registered
BBU Pooling: taskId = 11 taskName = UL_L1_PUSCH_MMSE0 is registered
BBU Pooling: taskId = 12 taskName = UL_L1_PUSCH_MMSE7 is registered
BBU Pooling: taskId = 13 taskName =  UL_L1_PUSCH_LLR is registered
BBU Pooling: taskId = 14 taskName = UL_L1_PUSCH_DECODE is registered
BBU Pooling: taskId = 15 taskName =   UL_L1_PUSCH_TB is registered
BBU Pooling: taskId = 16 taskName =      UL_L1_PUCCH is registered
BBU Pooling: taskId = 17 taskName =      UL_L1_PRACH is registered
BBU Pooling: taskId = 18 taskName =        UL_L1_SRS is registered
BBU Pooling: taskId = 19 taskName =       DL_L1_POST is registered
BBU Pooling: taskId = 20 taskName =       UL_L1_POST is registered
BBU Pooling: taskId = 21 taskName = DL_L1_BEAM_WEIGHT_GEN is registered
BBU Pooling: taskId = 22 taskName = DL_L1_BEAM_WEIGHT_TX is registered
BBU Pooling: taskId = 23 taskName = UL_L1_BEAM_WEIGHT_GEN is registered
BBU Pooling: taskId = 24 taskName = UL_L1_BEAM_WEIGHT_TX is registered
BBU Pooling: taskId = 25 taskName =     UL_L1_SRS_CE is registered
BBU Pooling: taskId = 26 taskName = UL_L1_SRS_REPORT is registered
BBU Pooling: taskId = 27 taskName = UL_L1_PUSCH_CE0_PRE is registered
BBU Pooling: taskId = 28 taskName = UL_L1_PUSCH_CE7_PRE is registered
BBU Pooling: next taskList of     DL_L1_CONFIG:    DL_L1_PDSCH_TB
DL_L1_PDSCH_RS_GEN    DL_L1_CONTROL_CHANNELS
BBU Pooling: next taskList of   DL_L1_PDSCH_TB:               N/A

BBU Pooling: next taskList of DL_L1_PDSCH_SCRAMBLER:  DL_L1_PDSCH_SYMBOL_TX
BBU Pooling: next taskList of DL_L1_PDSCH_SYMBOL_TX:        DL_L1_POST
BBU Pooling: next taskList of DL_L1_PDSCH_RS_GEN:  DL_L1_PDSCH_SYMBOL_TX
BBU Pooling: next taskList of DL_L1_CONTROL_CHANNELS:        DL_L1_POST
BBU Pooling: next taskList of     UL_L1_CONFIG:        UL_L1_POST
UL_L1_BEAM_WEIGHT_GEN
BBU Pooling: next taskList of  UL_L1_PUSCH_CE0:  UL_L1_PUSCH_MMSE0
UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of  UL_L1_PUSCH_CE7:  UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE0_PRE:  UL_L1_PUSCH_MMSE0
UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE7_PRE:  UL_L1_PUSCH_MMSE7
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE0:   UL_L1_PUSCH_LLR
BBU Pooling: next taskList of UL_L1_PUSCH_MMSE7:   UL_L1_PUSCH_LLR
BBU Pooling: next taskList of  UL_L1_PUSCH_LLR:  UL_L1_PUSCH_DECODE
BBU Pooling: next taskList of UL_L1_PUSCH_DECODE:               N/A

BBU Pooling: next taskList of   UL_L1_PUSCH_TB:        UL_L1_POST
BBU Pooling: next taskList of      UL_L1_PUCCH:        UL_L1_POST
BBU Pooling: next taskList of      UL_L1_PRACH:        UL_L1_POST
BBU Pooling: next taskList of        UL_L1_SRS:      UL_L1_SRS_CE
BBU Pooling: next taskList of       DL_L1_POST:               N/A

BBU Pooling: next taskList of       UL_L1_POST:               N/A

BBU Pooling: next taskList of DL_L1_BEAM_WEIGHT_GEN:  DL_L1_BEAM_WEIGHT_TX
BBU Pooling: next taskList of DL_L1_BEAM_WEIGHT_TX:        DL_L1_POST
BBU Pooling: next taskList of UL_L1_BEAM_WEIGHT_GEN:  UL_L1_BEAM_WEIGHT_TX
BBU Pooling: next taskList of UL_L1_BEAM_WEIGHT_TX:        UL_L1_POST
BBU Pooling: next taskList of     UL_L1_SRS_CE:  UL_L1_SRS_REPORT
BBU Pooling: next taskList of UL_L1_SRS_REPORT:              N/A

BBU Pooling: next taskList of UL_L1_PUSCH_CE0_PRE:   UL_L1_PUSCH_CE0
UL_L1_PUSCH_CE7
BBU Pooling: next taskList of UL_L1_PUSCH_CE7_PRE:   UL_L1_PUSCH_CE7
enter RtThread Launch
Allocated gpThreadWorker[coreIdx: 0][CoreNum: 4]: [0x7f738c000b70]
Allocated gpThreadWorker[coreIdx: 1][CoreNum: 5]: [0x7f738c000e20]
Allocated gpThreadWorker[coreIdx: 2][CoreNum: 6]: [0x7f738c0010d0]
Allocated gpThreadWorker[coreIdx: 3][CoreNum: 7]: [0x7f738c001380]
Allocated gpThreadWorker[coreIdx: 4][CoreNum: 8]: [0x7f738c001630]
Allocated gpThreadWorker[coreIdx: 5][CoreNum: 9]: [0x7f738c0018e0]
launching Thread 1 Queue 0 uCoreIdx 1 CoreId 5 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 0 Queue 0 uCoreIdx 0 CoreId 4 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 2 Queue 0 uCoreIdx 2 CoreId 6 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 3 Queue 0 uCoreIdx 3 CoreId 7 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 6][CoreNum: 10]: [0x7f738c001b90]
launching Thread 4 Queue 0 uCoreIdx 4 CoreId 8 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 5 Queue 0 uCoreIdx 5 CoreId 9 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 7][CoreNum: 11]: [0x7f738c001e40]
Allocated gpThreadWorker[coreIdx: 8][CoreNum: 12]: [0x7f738c0020f0]
bbupool_core_main: the server's coreNum = 48, the nCore = 18,nRtCoreMask =
0x1ff0001ff0, the nFeIfCore = 0,nFeIfCoreMask = 0x0
bbupool_core_main pthread_setaffinity_np succeed: coreId = 2, result = 0
Allocated gpThreadWorker[coreIdx: 9][CoreNum: 28]: [0x7f738c0023a0]
launching Thread 6 Queue 0 uCoreIdx 6 CoreId 10 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 10][CoreNum: 29]: [0x7f738c002650]
launching Thread 7 Queue 0 uCoreIdx 7 CoreId 11 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 11][CoreNum: 30]: [0x7f738c002900]
launching Thread 8 Queue 0 uCoreIdx 8 CoreId 12 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 9 Queue 0 uCoreIdx 9 CoreId 28 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 12][CoreNum: 31]: [0x7f738c002bb0]
Allocated gpThreadWorker[coreIdx: 13][CoreNum: 32]: [0x7f738c002e60]
launching Thread 10 Queue 0 uCoreIdx 10 CoreId 29 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 11 Queue 0 uCoreIdx 11 CoreId 30 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 14][CoreNum: 33]: [0x7f738c003110]
Allocated gpThreadWorker[coreIdx: 15][CoreNum: 34]: [0x7f738c0033c0]
launching Thread 12 Queue 0 uCoreIdx 12 CoreId 31 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 16][CoreNum: 35]: [0x7f738c003670]
launching Thread 13 Queue 0 uCoreIdx 13 CoreId 32 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

Allocated gpThreadWorker[coreIdx: 17][CoreNum: 36]: [0x7f738c003920]
18 thread associated with queue 0:coreIdx 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Leave RtThread Launch
launching Thread 14 Queue 0 uCoreIdx 14 CoreId 33 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 15 Queue 0 uCoreIdx 15 CoreId 34 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 16 Queue 0 uCoreIdx 16 CoreId 35 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

launching Thread 17 Queue 0 uCoreIdx 17 CoreId 36 Priority 94 Policy 1 nRtCoreSleep 1
nFriendCnt 0 nCurrentSfIdx -1

nr5g_gnb_mac2phy_api_proc_print_phy_init [0]:
    nCarrierIdx: 0
    nDMRSTypeAPos: 2
    nPhyCellId: 50
    nDLAbsFrePointA: 3500000
    nULAbsFrePointA: 3500000
    nDLBandwidth: 100
    nULBandwidth: 100
    nDLFftSize: 4096
    nULFftSize: 4096
    nSSBPwr: 0
    nSSBAbsFre: 0
    nSSBPeriod: 4
    nSSBSubcSpacing: 1
    nSSBSubcOffset: 0
    nSSBPrbOffset: 0
    nMIB[0]: 255
    nMIB[1]: 255
    nMIB[2]: 255
    nDLK0: 0
    nULK0: 0
    nSSBMask[0]: 0
    nSSBMask[1]: 0
    nNrOfTxAnt: 64
    nNrOfRxAnt: 64
    nNrOfDLPorts: 16
    nNrOfULPorts: 8
    nCarrierAggregationLevel: 2
    nFrameDuplexType: 1
    nSubcCommon: 1
    nTddPeriod: 10 (TDD)
    SlotConfig:
        Slot Sym 0 Sym 1 Sym 2 Sym 3 Sym 4 Sym 5 Sym 6 Sym 7 Sym 8 Sym 9 Sym10 Sym11
Sym12 Sym13
          0   DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
          1   DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
```

```
                    2    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
                    3    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     GD     GD
UL     UL
                    4    UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL
UL     UL
                    5    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
                    6    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
                    7    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
                    8    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     GD     GD
UL     UL
                    9    UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL
UL     UL

    nPrachConfIdx: 100
    nPrachSubcSpacing: 1
    nPrachZeroCorrConf: 1
    nPrachRestrictSet: 0
    nPrachRootSeqIdx: 0
    nPrachFreqStart: 100
    nPrachFdm: 1
    nPrachSsbRach: 0
    nPrachNrofRxRU: 4
    nCyclicPrefix: 0
    nGroupHopFlag: 0
    nSequenceHopFlag: 0
    nHoppingId: 0
    nUrllcCapable: 0
    nUrllcMiniSlotMask: 1 (0x00000001)
read_table: File table/common/pss_table.bin of size 381 read_size: 381
read_table: File table/common/sss_table.bin of size 128016 read_size: 128016
read_table: File table/common/srs_zc_36_plus.bin of size 905916 read_size: 905916
read_table: File table/common/pucch_zc_36_plus.bin of size 383040 read_size: 383040
read_table: File table/common/srs_wiener_sinc_comb2.bin of size 81216 read_size:
81216
read_table: File table/common/srs_wiener_sinc_comb4.bin of size 81216 read_size:
81216
BBU Pooling Info: maximum period length was configured, preMaxSF = 20480, postMasSF =
20480
set_slot_type SlotPattern:
    Slot:       0    1    2    3    4    5    6    7    8    9
        0      DL   DL   DL   SP   UL   DL   DL   DL   SP   UL

PHYDI-INIT[from 2] PhyInstance: 0
Processing MSG_TYPE_PHY_CONFIG_REQ: 1
nr5g_gnb_mac2phy_api_proc_print_phy_init [1]:
    nCarrierIdx: 1
    nDMRSTypeAPos: 2
    nPhyCellId: 50
    nDLAbsFrePointA: 3500000
    nULAbsFrePointA: 3500000
    nDLBandwidth: 100
    nULBandwidth: 100
    nDLFftSize: 4096
    nULFftSize: 4096
    nSSBPwr: 0
    nSSBAbsFre: 0
    nSSBPeriod: 4
    nSSBSubcSpacing: 1
    nSSBSubcOffset: 0
    nSSBPrbOffset: 0
    nMIB[0]: 255
    nMIB[1]: 255
    nMIB[2]: 255
    nDLK0: 0
    nULK0: 0
    nSSBMask[0]: 0
    nSSBMask[1]: 0
    nNrOfTxAnt: 64
    nNrOfRxAnt: 64
    nNrOfDLPorts: 16
    nNrOfULPorts: 8
    nCarrierAggregationLevel: 2
    nFrameDuplexType: 1
    nSubcCommon: 1
    nTddPeriod: 10 (TDD)
    SlotConfig:
        Slot Sym 0 Sym 1 Sym 2 Sym 3 Sym 4 Sym 5 Sym 6 Sym 7 Sym 8 Sym 9 Sym10 Sym11
Sym12 Sym13
           0    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           1    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           2    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           3    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    GD    GD
UL    UL
           4    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL
UL    UL
           5    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           6    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           7    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL
DL    DL
           8    DL    DL    DL    DL    DL    DL    DL    DL    DL    DL    GD    GD
UL    UL
           9    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL    UL
UL    UL

    nPrachConfIdx: 100
    nPrachSubcSpacing: 1
    nPrachZeroCorrConf: 1
    nPrachRestrictSet: 0
    nPrachRootSeqIdx: 0
    nPrachFreqStart: 100
    nPrachFdm: 1
    nPrachSsbRach: 0
    nPrachNrofRxRU: 4
    nCyclicPrefix: 0
    nGroupHopFlag: 0
    nSequenceHopFlag: 0
    nHoppingId: 0
    nUrllcCapable: 0
    nUrllcMiniSlotMask: 1 (0x00000001)
BBU Pooling Info: maximum period length was configured, preMaxSF = 20480, postMasSF =
20480
set_slot_type SlotPattern:
    Slot:       0    1    2    3    4    5    6    7    8    9
        0      DL   DL   DL   SP   UL   DL   DL   DL   SP   UL
```

```
PHYDI-INIT[from 2] PhyInstance: 1
Processing MSG_TYPE_PHY_CONFIG_REQ: 2
nr5g_gnb_mac2phy_api_proc_print_phy_init [2]:
    nCarrierIdx: 2
    nDMRSTypeAPos: 2
    nPhyCellId: 50
    nDLAbsFrePointA: 3500000
    nULAbsFrePointA: 3500000
    nDLBandwidth: 100
    nULBandwidth: 100
    nDLFftSize: 4096
    nULFftSize: 4096
    nSSBPwr: 0
    nSSBAbsFre: 0
    nSSBPeriod: 4
    nSSBSubcSpacing: 1
    nSSBSubcOffset: 0
    nSSBPrbOffset: 0
    nMIB[0]: 255
    nMIB[1]: 255
    nMIB[2]: 255
    nDLK0: 0
    nULK0: 0
    nSSBMask[0]: 0
    nSSBMask[1]: 0
    nNrOfTxAnt: 64
    nNrOfRxAnt: 64
    nNrOfDLPorts: 16
    nNrOfULPorts: 8
    nCarrierAggregationLevel: 2
    nFrameDuplexType: 1
    nSubcCommon: 1
    nTddPeriod: 10 (TDD)
    SlotConfig:
        Slot  Sym 0  Sym 1  Sym 2  Sym 3  Sym 4  Sym 5  Sym 6  Sym 7  Sym 8  Sym 9  Sym10  Sym11
Sym12  Sym13
           0    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           1    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           2    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           3    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     GD     GD
UL     UL
           4    UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL
UL     UL
           5    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           6    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           7    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     DL
DL     DL
           8    DL     DL     DL     DL     DL     DL     DL     DL     DL     DL     GD     GD
UL     UL
           9    UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL     UL
UL     UL

    nPrachConfIdx: 100
    nPrachSubcSpacing: 1
    nPrachZeroCorrConf: 1
    nPrachRestrictSet: 0
    nPrachRootSeqIdx: 0
    nPrachFreqStart: 100
    nPrachFdm: 1
    nPrachSsbRach: 0
    nPrachNrofRxRU: 4
    nCyclicPrefix: 0
    nGroupHopFlag: 0
    nSequenceHopFlag: 0
    nHoppingId: 0
    nUrllcCapable: 0
    nUrllcMiniSlotMask: 1 (0x00000001)
BBU Pooling Info: maximum period length was configured, preMaxSF = 20480, postMasSF =
20480
set_slot_type SlotPattern:
    Slot:       0    1    2    3    4    5    6    7    8    9
        0      DL   DL   DL   SP   UL   DL   DL   DL   SP   UL

PHYDI-INIT[from 2] PhyInstance: 2

---------------------------------------------------------
Global Variables:
---------------------------------------------------------
gCarrierAggLevel:                    3
gCarrierAggLevelInit:                3
gSupportedAVX2                       1
---------------------------------------------------------

Received MSG_TYPE_PHY_START_REQ: 0
Received MSG_TYPE_PHY_START_REQ: 1
Received MSG_TYPE_PHY_START_REQ: 2
Processing MSG_TYPE_PHY_START_REQ: 0
di_open port 0

xran_init_vfs_mapping: p 0 vf 0
xran_init_vfs_mapping: p 0 vf 1
XRAN_UP_VF: 0x0000
xran_timing_source_thread [CPU 22] [PID:  29437]
xran_open [CPU  2] [PID:  29437]
Waithing on Timing thread...
TTI interval 500 [us]
Start C-plane DL 71 us after TTI  [trigger on sym 2]
Start C-plane UL 200 us after TTI  [trigger on sym 6]
Start U-plane DL 196 us before OTA [offset  in sym -5]
Start U-plane UL 75 us OTA        [offset  in sym 3]
C-plane to U-plane delay 125 us after TTI
Start Sym timer 35714 ns
di_open port 1

xran_init_vfs_mapping: p 1 vf 2
xran_init_vfs_mapping: p 1 vf 3
Start C-plane DL 71 us after TTI  [trigger on sym 2]
Start C-plane UL 200 us after TTI  [trigger on sym 6]
Start U-plane DL 196 us before OTA [offset  in sym -5]
Start U-plane UL 75 us OTA        [offset  in sym 3]
C-plane to U-plane delay 125 us after TTI
Start Sym timer 35714 ns
xran_open [CPU  2] [PID:  29437]
Waithing on Timing thread...
di_open port 2

xran_init_vfs_mapping: p 2 vf 4
xran_init_vfs_mapping: p 2 vf 5
Start C-plane DL 71 us after TTI  [trigger on sym 2]
Start C-plane UL 200 us after TTI  [trigger on sym 6]
Start U-plane DL 196 us before OTA [offset  in sym -5]
Start U-plane UL 75 us OTA        [offset  in sym 3]
C-plane to U-plane delay 125 us after TTI
Start Sym timer 35714 ns
O-XU      0
HW        1
Num cores 4
Num ports 3
O-RU Cat  1
O-RU CC   3
O-RU eAxC 16
p:0 XRAN_JOB_TYPE_CP_DL worker id 1
p:0 XRAN_JOB_TYPE_CP_UL worker id 1
p:1 XRAN_JOB_TYPE_CP_DL worker id 1
p:1 XRAN_JOB_TYPE_CP_UL worker id 1
p:2 XRAN_JOB_TYPE_CP_DL worker id 1
p:2 XRAN_JOB_TYPE_CP_UL worker id 1
p:1 XRAN_JOB_TYPE_CP_DL worker id 2
p:1 XRAN_JOB_TYPE_CP_UL worker id 2
p:2 XRAN_JOB_TYPE_CP_DL worker id 2
p:2 XRAN_JOB_TYPE_CP_UL worker id 2
xran_generic_worker_thread [CPU 23] [PID:  29437]
spawn worker 0 core 23
xran_generic_worker_thread [CPU 24] [PID:  29437]
spawn worker 1 core 24
xran_generic_worker_thread [CPU 25] [PID:  29437]
spawn worker 2 core 25
xran_open [CPU  2] [PID:  29437]
Waithing on Timing thread...
----------------------------------------------------------------------------
mem_mgr_display_size:
    Num Memory Alloc:            38,294
    Total Memory Size:  20,049,968,118
----------------------------------------------------------------------------


PHYDI-START[from 2] PhyInstance: 0, Mode: 4, Count: 100207, Period: 0, NumSlotPerSfn:
20
PHYDI-START[from 2] PhyInstance: 1, Mode: 4, Count: 100207, Period: 0, NumSlotPerSfn:
20
PHYDI-START[from 2] PhyInstance: 2, Mode: 4, Count: 100207, Period: 0, NumSlotPerSfn:
20
Setting nMultiCellModeDelay: 40000
nr5g_gnb_urllc_register_call_backs: nTimerMode[0] nUrllcMiniSlotMask[0]
port [0] gnb_io_xran_start: gGnbIoXranStarted[0] CC 3 Ant 16 AntElm 64  [Cell:
nNrOfDLPorts 16 nNrOfULPorts 8]
port 0 has 1 CCs
port 0 cc_id 0 is phy id 0
XRAN front haul xran_mm_init
xran_sector_get_instances [0]: CC 0 handle 0x7f6fe7383280
Handle: 0xee1c8e0 Instance: 0x7f6fe7383280
gnb_io_xran_start [0]: CC 0 handle 0x7f6fe7383280
Sucess xran_mm_init Instance 0x7f6fe7383280
nSectorNum 1
ru_0_cc_0_idx_0: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 0] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 0] mb pool 0x44c493480
ru_0_cc_0_idx_1: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 1] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 1] mb pool 0x444381640
```

```
ru_0_cc_0_idx_2: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 2] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 2] mb pool 0x443dff2c0
ru_0_cc_0_idx_3: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 3] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 3] mb pool 0x443c5cf40
ru_0_cc_0_idx_4: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 4] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 4] mb pool 0x443ababc0
ru_0_cc_0_idx_5: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 5] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 5] mb pool 0x443538840
ru_0_cc_0_idx_6: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 6] nNumberOfBuffers 8960
nBufferSize 8192
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 6] mb pool 0x4433964c0
ru_0_cc_0_idx_7: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 7] nNumberOfBuffers 35840
nBufferSize 14432
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 7] mb pool 0x4431f4140
ru_0_cc_0_idx_8: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 8] nNumberOfBuffers
1146880 nBufferSize 32
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 8] mb pool 0x442ff1dc0
ru_0_cc_0_idx_9: [ handle 0x7f6fe7383280 0 0 ] [nPoolIndex 9] nNumberOfBuffers 35840
nBufferSize 12560
CC:[ handle 0x7f6fe7383280 ru 0 cc_idx 0 ] [nPoolIndex 9] mb pool 0x441e6fa40
port [0] gnb_io_xran_init_cp
port [0] init xran successfully
port [1] gnb_io_xran_start: gGnbIoXranStarted[0] CC 3 Ant 16 AntElm 64  [Cell:
nNrOfDLPorts 16 nNrOfULPorts 8]
port 1 has 1 CCs
port 1 cc_id 0 is phy id 1
XRAN front haul xran_mm_init
xran_sector_get_instances [1]: CC 0 handle 0x7f6fe7383380
Handle: 0xee1c940 Instance: 0x7f6fe7383380
gnb_io_xran_start [1]: CC 0 handle 0x7f6fe7383380
Sucess xran_mm_init Instance 0x7f6fe7383280
nSectorNum 1
ru_1_cc_0_idx_0: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 0] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 0] mb pool 0x2a1525740
ru_1_cc_0_idx_1: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 1] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 1] mb pool 0x299413900
ru_1_cc_0_idx_2: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 2] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 2] mb pool 0x28f1112c0
ru_1_cc_0_idx_3: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 3] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 3] mb pool 0x287f4fb80
ru_1_cc_0_idx_4: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 4] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 4] mb pool 0x27fe3dd40
ru_1_cc_0_idx_5: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 5] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 5] mb pool 0x275b3b700
ru_1_cc_0_idx_6: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 6] nNumberOfBuffers 8960
nBufferSize 8192
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 6] mb pool 0x26e979fc0
ru_1_cc_0_idx_7: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 7] nNumberOfBuffers 35840
nBufferSize 14432
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 7] mb pool 0x269ce9980
ru_1_cc_0_idx_8: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 8] nNumberOfBuffers
1146880 nBufferSize 32
```

```
O-DU: thread_run start time: 08/11/20 23:05:24.000000001 UTC [500]
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 8] mb pool 0x249d33b40
ru_1_cc_0_idx_9: [ handle 0x7f6fe7383380 1 0 ] [nPoolIndex 9] nNumberOfBuffers 35840
nBufferSize 12560
CC:[ handle 0x7f6fe7383380 ru 1 cc_idx 0 ] [nPoolIndex 9] mb pool 0x2215b1500
port [1] gnb_io_xran_init_cp
port [1] init xran successfully
port [2] gnb_io_xran_start: gGnbIoXranStarted[0] CC 3 Ant 16 AntElm 64  [Cell:
nNrOfDLPorts 16 nNrOfULPorts 8]
port 2 has 1 CCs
port 2 cc_id 0 is phy id 2
XRAN front haul xran_mm_init
xran_sector_get_instances [2]: CC 0 handle 0x7f6fe7383440
Handle: 0xee1c9a0 Instance: 0x7f6fe7383440
gnb_io_xran_start [2]: CC 0 handle 0x7f6fe7383440
Sucess xran_mm_init Instance 0x7f6fe7383280
nSectorNum 1
ru_2_cc_0_idx_0: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 0] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 0] mb pool 0x203b7bdc0
ru_2_cc_0_idx_1: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 1] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 1] mb pool 0x1fba69f80
ru_2_cc_0_idx_2: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 2] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 2] mb pool 0x1f1767940
ru_2_cc_0_idx_3: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 3] nNumberOfBuffers 8960
nBufferSize 14432
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 3] mb pool 0x1ea5a6200
ru_2_cc_0_idx_4: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 4] nNumberOfBuffers 286720
nBufferSize 32
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 4] mb pool 0x1e24943c0
ru_2_cc_0_idx_5: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 5] nNumberOfBuffers 8960
nBufferSize 12560
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 5] mb pool 0x1d8191d80
ru_2_cc_0_idx_6: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 6] nNumberOfBuffers 8960
nBufferSize 8192
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 6] mb pool 0x1d0fd0640
ru_2_cc_0_idx_7: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 7] nNumberOfBuffers 35840
nBufferSize 14432
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 7] mb pool 0x1cc340000
ru_2_cc_0_idx_8: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 8] nNumberOfBuffers
1146880 nBufferSize 32
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 8] mb pool 0x1ac38a1c0
ru_2_cc_0_idx_9: [ handle 0x7f6fe7383440 2 0 ] [nPoolIndex 9] nNumberOfBuffers 35840
nBufferSize 12560
CC:[ handle 0x7f6fe7383440 ru 2 cc_idx 0 ] [nPoolIndex 9] mb pool 0x183c07b80
port [2] gnb_io_xran_init_cp
port [2] init xran successfully
O-DU: XRAN start time: 08/11/20 23:05:24.384220762 UTC [500]
BBU Pooling: enter multicell Activate!
BBU Pooling Info: bbupool rt thread start on CoreIdx 14 coreId 33 at 118352443946329
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 1 coreId 5 at 118352443939667 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 11 coreId 30 at 118352443942535
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 8 coreId 12 at 118352443944575
at sf=0 with queue 0 successfully
BBU Pooling: active result: Q_id = 0,currenSf = 0, curCellNum = 0, activesfn = 4,
CellNumInActSfn = 3
```

```
BBU Pooling Info: bbupool rt thread start on CoreIdx 2 coreId 6 at 118352443929961 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 15 coreId 34 at 118352443933301
at sf=0 with queue 0 successfully
BBU Pooling: multiCell Activate sucessfully!
BBU Pooling Info: bbupool rt thread start on CoreIdx 13 coreId 32 at 118352443935245
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 4 coreId 8 at 118352443936745 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 6 coreId 10 at 118352443936883
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 3 coreId 7 at 118352443936747 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 12 coreId 31 at 118352443938019
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 5 coreId 9 at 118352443939937 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 9 coreId 28 at 118352443941217
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 16 coreId 35 at 118352443944465
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 17 coreId 36 at 118352443937701
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 0 coreId 4 at 118352443926969 at
sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 10 coreId 29 at 118352443928691
at sf=0 with queue 0 successfully
BBU Pooling Info: bbupool rt thread start on CoreIdx 7 coreId 11 at 118352443931713
at sf=0 with queue 0 successfully
phy_bbupool_rx_handler: PhyId[0] nSfIdx[4] frame,slot[0,5] gNumSlotPerSfn[20]
==== l1app Time: 5002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time: [  0.00..
0.00..  0.00] usces
==== [o-du0][rx 3807776 pps 761555 kbps 4744396][tx 10937607 pps 2187521 kbps
26031486] [on_time 3807776 early 0 late 0 corrupt 0 pkt_dupl 144 Total 3807776]
     Pusch[  439372   439372   439372   439372   439372   439372   439372   439372]
SRS[  292800]
==== [o-du1][rx 1469469 pps 293893 kbps 2684928][tx 3649817 pps 729963 kbps 9156812]
[on_time 1469469 early 0 late 0 corrupt 0 pkt_dupl 144 Total 1469469]
     Pusch[  146964   146956   146964   146956   146964   146956   146964   146956]
SRS[  293788]
==== [o-du2][rx 1469463 pps 293892 kbps 2684960][tx 3648883 pps 729776 kbps 9152795]
[on_time 1469463 early 0 late 0 corrupt 0 pkt_dupl 144 Total 1469463]
     Pusch[  146956   146956   146956   146956   146956   146956   146956   146956]
SRS[  293815]
-------------------------------------------------------------------------------------
------------------------------------------------------------------
      Cell         DL Tput           UL Tput          UL BLER
      0  (Kbps)          0          0 /          0      0.00%
      1  (Kbps)          0          0 /          0      0.00%
      2  (Kbps)          0          0 /          0      0.00%
-------------------------------------------------------------------------------------
------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:    0   4   2   4   4   2   3  13  17   0  13  15  14  16  14  17  15  14
9.28
     Xran Id:  22  23  24  25     Master Core Util:  85 %
-------------------------------------------------------------------------------------
------------------------------------------------------------------
==== l1app Time: 10002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time: [  0.00..
0.00..  0.00] usces
```

```
==== [o-du0][rx 5472406 pps 332926 kbps 4744396][tx 21871698 pps 2186818 kbps
26038405] [on_time 5472406 early 0 late 0 corrupt 0 pkt_dupl 144 Total 5472406]
     Pusch[  192084    192078    192078    192078    192078    192078    192078    192078]
SRS[  128000]
==== [o-du1][rx 2109680 pps 128042 kbps 2684917][tx 7297930 pps 729622 kbps 9156922]
[on_time 2109680 early 0 late 0 corrupt 0 pkt_dupl 144 Total 2109680]
     Pusch[   64026     64026     64026     64026     64026     64026     64026     64026]
SRS[  128004]
==== [o-du2][rx 2109682 pps 128043 kbps 2684993][tx 7296833 pps 729590 kbps 9156258]
[on_time 2109682 early 0 late 0 corrupt 0 pkt_dupl 144 Total 2109682]
     Pusch[   64026     64026     64026     64026     64026     64026     64026     64026]
SRS[  128011]
-------------------------------------------------------------------------------------
------------------------------------------------------------------
      Cell         DL Tput           UL Tput          UL BLER
      0 (Kbps)  6,894,368    576,420 /   576,492       0.00%
      1 (Kbps)          0          0 /         0       0.00%
      2 (Kbps)          0          0 /         0       0.00%
-------------------------------------------------------------------------------------
------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   15  30  34  29  26  28  26  46  50   0  40  40  43  42  44  42  48  50
35.17
     Xran Id:  22  23  24  25     Master Core Util:  95 %
-------------------------------------------------------------------------------------
------------------------------------------------------------------
==== l1app Time: 15003 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time: [  0.00..
0.00..  0.00] usces
==== [o-du0][rx 7136544 pps 332827 kbps 4744396][tx 32806663 pps 2186993 kbps
26042173] [on_time 7136544 early 0 late 0 corrupt 0 pkt_dupl 144 Total 7136544]
     Pusch[  192012    192018    192018    192018    192018    192018    192018    192018]
SRS[  128000]
==== [o-du1][rx 2749728 pps 128009 kbps 2684895][tx 10945622 pps 729538 kbps 9155645]
[on_time 2749728 early 0 late 0 corrupt 0 pkt_dupl 144 Total 2749728]
     Pusch[   64006     64006     64006     64006     64006     64006     64006     64006]
SRS[  128000]
==== [o-du2][rx 2749730 pps 128009 kbps 2684840][tx 10944272 pps 729487 kbps 9154660]
[on_time 2749730 early 0 late 0 corrupt 0 pkt_dupl 144 Total 2749730]
     Pusch[   64006     64006     64006     64006     64006     64006     64006     64006]
SRS[  128000]
-------------------------------------------------------------------------------------
------------------------------------------------------------------
      Cell         DL Tput           UL Tput          UL BLER
      0 (Kbps)  6,896,256    576,780 /   576,780       0.00%
      1 (Kbps)    539,740     65,260 /    65,260       0.00%
      2 (Kbps)          0          0 /         0       0.00%
-------------------------------------------------------------------------------------
------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   27  33  40  38  38  35  34  56  56  26  50  47  48  47  51  48  57  57
43.78
     Xran Id:  22  23  24  25     Master Core Util:  95 %
-------------------------------------------------------------------------------------
------------------------------------------------------------------
Setting MLogMask because nMLogDelay == 0
==== l1app Time: 20002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[480.00..500.23..516.00] usces
```

```
==== [o-du0][rx 8799776 pps 332646 kbps 4744396][tx 43740623 pps 2186792 kbps
26042944] [on_time 8799776 early 0 late 0 corrupt 0 pkt_dupl 144 Total 8799776]
     Pusch[  191904   191904   191904   191904   191904   191904   191904   191904]
SRS[  128000]
==== [o-du1][rx 3389472 pps 127948 kbps 2684982][tx 14591619 pps 729199 kbps 9154093]
[on_time 3389472 early 0 late 0 corrupt 0 pkt_dupl 144 Total 3389472]
     Pusch[   63968    63968    63968    63968    63968    63968    63968    63968]
SRS[  128000]
==== [o-du2][rx 3389474 pps 127948 kbps 2684873][tx 14589997 pps 729145 kbps 9152608]
[on_time 3389474 early 0 late 0 corrupt 0 pkt_dupl 144 Total 3389474]
     Pusch[   63968    63968    63968    63968    63968    63968    63968    63968]
SRS[  128000]
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
      Cell         DL Tput           UL Tput         UL BLER
      0 (Kbps)   6,896,256    576,780 /   576,780      0.00%
      1 (Kbps)     539,814     65,260 /    65,260      0.00%
      2 (Kbps)     539,814     65,260 /    65,260      0.00%
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   43  47  46  43  42  43  41  61  60  27  57  56  58  57  55  56  64  62
51.00
     Xran Id:  22  23  24  25     Master Core Util:  96 %
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
==== l1app Time: 25002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[442.00..500.12..562.00] usces
==== [o-du0][rx 10463824 pps 332809 kbps 4744396][tx 54675513 pps 2186978 kbps
26044150] [on_time 10463824 early 0 late 0 corrupt 0 pkt_dupl 144 Total 10463824]
     Pusch[  192006   192006   192006   192006   192006   192006   192006   192006]
SRS[  128000]
==== [o-du1][rx 4029487 pps 128003 kbps 2684928][tx 18237287 pps 729133 kbps 9150163]
[on_time 4029487 early 0 late 0 corrupt 0 pkt_dupl 144 Total 4029487]
     Pusch[   64002    64002    64002    64002    64002    64002    64002    64001]
SRS[  128000]
==== [o-du2][rx 4029474 pps 128000 kbps 2684873][tx 18235338 pps 729068 kbps 9148513]
[on_time 4029474 early 0 late 0 corrupt 0 pkt_dupl 144 Total 4029474]
     Pusch[   64000    64000    64000    64000    64000    64000    64000    64000]
SRS[  128000]
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
      Cell         DL Tput           UL Tput         UL BLER
      0 (Kbps)   6,896,256    576,492 /   576,492      0.00%
      1 (Kbps)     539,814     65,260 /    65,260      0.00%
      2 (Kbps)     539,814     65,260 /    65,260      0.00%
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   44  48  46  46  44  41  43  62  61  27  58  59  55  56  56  58  61  62
51.50
     Xran Id:  22  23  24  25     Master Core Util:  95 %
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
==== l1app Time: 30002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[435.00..500.12..562.00] usces
==== [o-du0][rx 12127888 pps 332812 kbps 4744396][tx 65610457 pps 2186988 kbps
26044065] [on_time 12127888 early 0 late 0 corrupt 0 pkt_dupl 144 Total 12127888]
```

```
     Pusch[  192012   192006   192012   192006   192010   192006   192006   192006]
SRS[  128000]
==== [o-du1][rx 4669504 pps 128003 kbps 2685058][tx 21883550 pps 729252 kbps 9152750]
[on_time 4669504 early 0 late 0 corrupt 0 pkt_dupl 144 Total 4669504]
     Pusch[   64002    64002    64002    64002    64002    64002    64002    64003]
SRS[  128000]
==== [o-du2][rx 4669498 pps 128004 kbps 2684993][tx 21881293 pps 729191 kbps 9151846]
[on_time 4669498 early 0 late 0 corrupt 0 pkt_dupl 144 Total 4669498]
     Pusch[   64004    64004    64004    64004    64002    64002    64002    64002]
SRS[  128000]
--------------------------------------------------------------------------------------
------------------------------------------------------------------
      Cell         DL Tput           UL Tput          UL BLER
      0 (Kbps)   6,896,256    577,069 /    577,069       0.00%
      1 (Kbps)     539,814     65,260 /     65,260       0.00%
      2 (Kbps)     539,814     65,260 /     65,260       0.00%
--------------------------------------------------------------------------------------
------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   44  47  45  47  43  43  42  63  63  27  56  56  56  55  58  55  65  62
51.50
     Xran Id:  22  23  24  25     Master Core Util:  95 %
--------------------------------------------------------------------------------------
------------------------------------------------------------------
==== l1app Time: 35002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[434.00..500.12..554.00] usces
==== [o-du0][rx 13792256 pps 332873 kbps 4744892][tx 76545521 pps 2187012 kbps
26042901] [on_time 13792256 early 0 late 0 corrupt 0 pkt_dupl 144 Total 13792256]
     Pusch[  192042   192048   192042   192048   192044   192048   192048   192048]
SRS[  128000]
==== [o-du1][rx 5309632 pps 128025 kbps 2685102][tx 25528867 pps 729063 kbps 9151639]
[on_time 5309632 early 0 late 0 corrupt 0 pkt_dupl 144 Total 5309632]
     Pusch[   64016    64016    64016    64016    64016    64016    64016    64016]
SRS[  128000]
==== [o-du2][rx 5309632 pps 128026 kbps 2685102][tx 25526238 pps 728989 kbps 9150147]
[on_time 5309632 early 0 late 0 corrupt 0 pkt_dupl 144 Total 5309632]
     Pusch[   64016    64016    64016    64016    64018    64018    64017    64017]
SRS[  128000]
--------------------------------------------------------------------------------------
------------------------------------------------------------------
      Cell         DL Tput           UL Tput          UL BLER
      0 (Kbps)   6,896,256    576,780 /    576,780       0.00%
      1 (Kbps)     539,814     65,260 /     65,260       0.00%
      2 (Kbps)     539,814     65,260 /     65,260       0.00%
--------------------------------------------------------------------------------------
------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   43  48  45  47  43  41  42  66  61  27  57  57  55  56  57  56  64  62
51.50
     Xran Id:  22  23  24  25     Master Core Util:  95 %
--------------------------------------------------------------------------------------
------------------------------------------------------------------
==== l1app Time: 40002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[440.00..500.12..553.00] usces
==== [o-du0][rx 15455740 pps 332696 kbps 4744396][tx 87479892 pps 2186874 kbps
26042995] [on_time 15455740 early 0 late 0 corrupt 0 pkt_dupl 144 Total 15455740]
     Pusch[  191940   191940   191940   191940   191940   191940   191940   191940]
SRS[  127964]
```

```
==== [o-du1][rx 5949408 pps 127955 kbps 2684764][tx 29174424 pps 729111 kbps 9150009]
[on_time 5949408 early 0 late 0 corrupt 0 pkt_dupl 144 Total 5949408]
     Pusch[   63980    63980    63980    63980    63980    63980    63980    63980]
SRS[  127936]
==== [o-du2][rx 5949410 pps 127955 kbps 2684840][tx 29171380 pps 729028 kbps 9148386]
[on_time 5949410 early 0 late 0 corrupt 0 pkt_dupl 144 Total 5949410]
     Pusch[   63980    63980    63980    63980    63980    63980    63981    63981]
SRS[  127936]
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
      Cell        DL Tput           UL Tput          UL BLER
      0 (Kbps)   6,896,256    576,780 /   576,780      0.00%
      1 (Kbps)     539,814     65,260 /    65,260      0.00%
      2 (Kbps)     539,814     65,260 /    65,260      0.00%
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   44  48  44  45  42  42  43  63  63  27  57  56  55  58  56  56  64  62
51.39
     Xran Id:  22  23  24  25     Master Core Util:  95 %
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
==== l1app Time: 45002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[436.00..500.12..556.00] usces
==== [o-du0][rx 17119776 pps 332807 kbps 4743900][tx 98415119 pps 2187045 kbps
26043843] [on_time 17119776 early 0 late 0 corrupt 0 pkt_dupl 144 Total 17119776]
     Pusch[  192000   192000   192000   192000   192000   192000   192000   192000]
SRS[  128036]
==== [o-du1][rx 6589472 pps 128012 kbps 2684753][tx 32820214 pps 729158 kbps 9154170]
[on_time 6589472 early 0 late 0 corrupt 0 pkt_dupl 144 Total 6589472]
     Pusch[   64000    64000    64000    64000    64000    64000    64000    64000]
SRS[  128064]
==== [o-du2][rx 6589474 pps 128012 kbps 2684753][tx 32816780 pps 729080 kbps 9152613]
[on_time 6589474 early 0 late 0 corrupt 0 pkt_dupl 144 Total 6589474]
     Pusch[   64000    64000    64000    64000    64000    64000    64000    64000]
SRS[  128064]
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
      Cell        DL Tput           UL Tput          UL BLER
      0 (Kbps)   6,896,256    576,780 /   576,780      0.00%
      1 (Kbps)     539,814     65,260 /    65,260      0.00%
      2 (Kbps)     539,814     65,260 /    65,260      0.00%
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   44  47  46  47  43  42  42  61  63  27  56  58  56  56  58  57  63  65
51.72
     Xran Id:  22  23  24  25     Master Core Util:  95 %
---------------------------------------------------------------------------------------
-------------------------------------------------------------------
==== l1app Time: 50002 ms NumCarrier: 3 NumBbuCores: 18. Tti2Tti Time:
[436.00..500.12..551.00] usces
==== [o-du0][rx 18783776 pps 332800 kbps 4744396][tx 109350065 pps 2186989 kbps
26043142] [on_time 18783776 early 0 late 0 corrupt 0 pkt_dupl 144 Total 18783776]
     Pusch[  192000   192000   192000   192000   192000   192000   192000   192000]
SRS[  128000]
==== [o-du1][rx 7229472 pps 128000 kbps 2684928][tx 36466505 pps 729258 kbps
18302595] [on_time 7229472 early 0 late 0 corrupt 0 pkt_dupl 144 Total 7229472]
```

```
     Pusch[   64000    64000    64000    64000    64000    64000    64000    64000]
SRS[  128000]
==== [o-du2][rx 7229474 pps 128000 kbps 2684895][tx 36462749 pps 729193 kbps 9148265]
[on_time 7229474 early 0 late 0 corrupt 0 pkt_dupl 144 Total 7229474]
     Pusch[   64000    64000    64000    64000    64000    64000    64000    64000]
SRS[  128000]
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
      Cell        DL Tput          UL Tput          UL BLER
      0 (Kbps)  6,896,256    576,492 /    576,492      0.00%
      1 (Kbps)    539,814     65,260 /     65,260      0.00%
      2 (Kbps)    539,814     65,260 /     65,260      0.00%
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
Core Utilization [18 BBU core(s)]:
     Core Id:   4   5   6   7   8   9  10  11  12  28  29  30  31  32  33  34  35  36
Avg
     Util %:   43  47  45  47  43  41  41  62  63  27  57  55  57  56  55  57  62  66
51.33
     Xran Id:  22  23  24  25      Master Core Util:  95 %
-------------------------------------------------------------------------------------
-------------------------------------------------------------------
```

7. To execute testmac with O-DU functionality according to O-RAN Fronthaul specification, enter:

```
[root@xran flexran] cd ./bin/nr5g/gnb/testmac
```

8. To execute test case type:

```
./l2.sh --testfile=./cascade_lake-sp/csxsp_mu1_100mhz_mmimo_hton_xran.cfg
```

where output corresponding to Test MAC:

```
[root@icelake-scs1-1 testmac]# ./l2.sh --testfile=./icelake-
sp/icxsp_mu1_100mhz_mmimo_64x64_hton_xran.cfg
kernel.sched_rt_runtime_us = -1
kernel.shmmax = 2147483648
kernel.shmall = 2147483648
Note: Forwarding request to 'systemctl disable irqbalance.service'.
start 5GNR Test MAC
=========================
5GNR Testmac Application
=========================
testmac_cfg_set_cfg_filename: Coult not find string 'cfgfile' in command line. Using
default File: testmac_cfg.xml


---------------------------
TestMacCfg.xml Version: 20.08
---------------------------

 --version=20.08
 --wls_dev_name=wls0
 --wlsMemorySize=0x3F600000
 --dpdkIovaMode=0
 --PhyStartMode=1
 --PhyStartPeriod=40
 --PhyStartCount=0
 --MlogSubframes=128
 --MlogCores=3
 --MlogSize=2048
 --latencyTest=0
 --wlsRxThread=1, 90, 0
 --systemThread=0, 0, 0
 --runThread=0, 89, 0
 --urllcThread=16, 90, 0

wls_dev_filename: wls0
sys_reg_signal_handler:[err] signal handler in NULL
sys_reg_signal_handler:[err] signal handler in NULL
timer_set_tsc_freq_from_clock: System clock (rdtsc) resolution 1496523032 [Hz]
                               Ticks per usec 1496
MLogOpen: filename(testmac-mlog.bin) mlogSubframes (128), mlogCores(3),
mlogSize(2048) mlog_mask (-1)
    mlogSubframes (128), mlogCores(3), mlogSize(2048)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1496526140 [Hz]
        Ticks per us 1496
    MLog Storage: 0x7f821905d100 -> 0x7f821911d920 [ 788512 bytes ]
    localMLogFreqReg: 1496. Storing: 1496
    Mlog Open successful

Calling rte_eal_init: testmac -c1 --proc-type=auto --file-prefix wls0 --iova-mode=pa
EAL: Detected 48 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Auto-detected process type: SECONDARY
EAL: Multi-process socket /var/run/dpdk/wls0/mp_socket_29473_6b9e031eaf8b
EAL: Selected IOVA mode 'PA'
EAL: Probing VFIO support...
EAL: PCI device 0000:01:00.0 on NUMA socket 0
EAL:   probe driver: 8086:1533 net_e1000_igb
EAL: PCI device 0000:18:00.0 on NUMA socket 0
EAL:   probe driver: 8086:1563 net_ixgbe
EAL: PCI device 0000:18:00.1 on NUMA socket 0
EAL:   probe driver: 8086:1563 net_ixgbe
EAL: PCI device 0000:8c:00.0 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:8c:00.1 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:90:00.0 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
EAL: PCI device 0000:90:00.1 on NUMA socket 0
EAL:   probe driver: 8086:d58 net_i40e
wls_lib: Open wls0 (DPDK memzone)
wls_lib: WLS_Open 0x43f600000
wls_lib: link: 1 <-> 0
wls_lib: Mode 1
wls_lib: WLS shared management memzone: wls0
wls_lib: hugePageSize on the system is 1073741824
wls_lib: WLS_Alloc [1063256064] bytes
wls_lib: Connecting to remote peer ...
wls_lib: Connected to remote peer
wls_mac_create_mem_array: pMemArray[0xf354350] pMemArrayMemory[0x400000000]
totalSize[1063256064] nBlockSize[262144] numBlocks[4056]
WLS_EnqueueBlock [1]
WLS inited ok [383]


=====================================================================================
======================
TESTMAC VERSION
=====================================================================================
======================

$Version: #DIRTY# $ (x86)
IMG-date: Aug  5 2020
```

```
IMG-time: 18:32:53
==================================================================================
======================


==================================================================================
======================
Testmac threads in application
==================================================================================
======================
testmac_run_thread:       [PID:  29477] binding on [CPU  0] [PRIO: 89] [POLICY:  1]
wls_mac_rx_task:          [PID:  29476] binding on [CPU  1] [PRIO: 90] [POLICY:  1]
==================================================================================
======================

testmac_set_phy_start: mode[1], period[40], count[0]

testmac_run_load_files:
Loading DL Config Files:
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/dl/testmac_dl_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/dl/testmac_dl_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/dl/testmac_dl_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/dl/testmac_dl_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/dl/testmac_dl_mu3_100mhz.cfg
Loading UL Config Files:
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu1_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu1_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu1_40mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/ul/testmac_ul_mu3_100mhz.cfg
Loading FD Config Files:
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu0_5mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu0_10mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu0_20mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu1_40mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu1_100mhz.cfg
    testmac_run_parse_file Parsing config file:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/fd/testmac_fd_mu3_100mhz.cfg

TESTMAC DL TESTS:
    Numerology[0] Bandwidth[5]
```

```
          1001  1002  1003  1004  1005  1006  1007  1008
    Numerology[0] Bandwidth[10]
          1001  1002  1003  1004  1005  1006  1007  1008
    Numerology[0] Bandwidth[20]
          1001  1002  1003  1004  1005  1006  1007  1008
    Numerology[1] Bandwidth[100]
          1200  1201  1202  1203  1204  1205  1206  1207  1210  1211
          1212  1213  1214  1215  1216  1217  1218  1219  1220  1221
          1222  1223  1224  1225  1226  1227  1228  1229  1230  1241
          1242  1243  1244  1245  1250  1251  1252  1260  1261  1262
          1263  1264  1265  1266  1267  1268  1269  1270  1271  1272
          1300  1301  1302  1303  1304  1305  1402  1404  1408  1416
          1500  1501  1502  1503  1504  1505  1506  2213  2214  2215
          2217  2218  2219  2223  2224  2225  2227  2228  2229  2500
          2501  2502  2503  2504  3213  3214  3215  3217  3218  3219
          3223  3224  3225  3227  3228  3229
    Numerology[3] Bandwidth[100]
          1001  1002  1003  1005  1006  1007  1008  1009  1010  1011
          1012  1013  1014  1015  1016  1017  1018  1019  1030  1031
          1032  1033  2001  2002  2003  2030  2033  3001  3002  3003
          3030

TESTMAC UL TESTS:
    Numerology[0] Bandwidth[5]
          1001  1002  1003  1069  1070  1071  1072  1073  1074  1075
          1076  1077
    Numerology[0] Bandwidth[10]
          1001  1002  1069  1070  1071  1072  1073  1074  1075  1076
          1077
    Numerology[0] Bandwidth[20]
          1001  1002  1003  1004  1005  1006  1007  1008  1069  1070
          1071  1072  1073  1074  1075  1076  1077
    Numerology[1] Bandwidth[10]
          1069  1070  1071  1072  1073  1074  1075  1076  1077
    Numerology[1] Bandwidth[20]
          1069  1070  1071  1072  1073  1074  1075  1076  1077
    Numerology[1] Bandwidth[40]
          1069  1070  1071  1072  1073  1074  1075  1076  1077
    Numerology[1] Bandwidth[100]
          1010  1030  1031  1032  1033  1034  1035  1036  1037  1038
          1039  1040  1041  1042  1043  1070  1071  1072  1073  1074
          1080  1081  1082  1083  1084  1085  1086  1087  1091  1092
          1093  1094  1095  1096  1100  1101  1102  1103  1104  1105
          1106  1107  1108  1110  1111  1113  1114  1115  1116  1117
          1118  1119  1120  1121  1122  1123  1124  1130  1131  1132
          1133  1134  1135  1136  1137  1138  1139  1140  1141  1142
          1143  1150  1152  1153  1154  1155  1156  1157  1159  1160
          1161  1162  1163  1164  1165  1166  1167  1168  1169  1170
          1171  1172  1173  1200  1201  1202  1203  1204  1205  1206
          1207  1208  1209  1210  1211  1212  1213  1214  1215  1216
          1217  1218  1219  1220  1221  1222  1230  1231  1232  1233
          1234  1235  1236  1237  1402  1404  1408  1416  1420  1421
          1422  1423  1424  1425  1426  1427  1428  1429  1430  1431
          1432  1433  1434  1435  1436  1437  1438  1500  1503  1504
          1505  1506  1507  1508  1512  1513  1514  1515  1516  1540
          1541  1542  1563  1564  1565  1566  1567  1568  1569  1570
          1571  1572  1573  1574  1575  1576  1577  1600  1601  1602
          1603  1604  1605  1606  1607  1608  1609  1610  1611  1612
          1613  1614  1615  1616  1617  1618  1619  1620  1621  1622
          1623  1624  1625  1626  1627  1628  1629  1630  1631  1632
          1633  1634  1635  1636  1637  1638  1639  1640  1641  1642
          1700  1701  1702  1969  1970  1971  1972  1973  1974  1975
          1976  1977  2236  2237  3236  3237
    Numerology[3] Bandwidth[100]
          1001  1002  1003  1004  1005  1006  1007  1010  1011  1012
          1013  1014  1015  1020  1021  1022  1023  1024  1025  1026
          1027  1028  1029  1030  1031  1032  1033  1034  1035  1036
          1037  1040  1041  1042  1043  1044  1045  1046  1050  1051
          1052  1053  1054  1059  1060  1061  1062  1063  1064  1065
          1066  1067  1070  1071  1073  1074  1081  1082  1083  1084
          1085  1086  2001  2002  2003  3001  3002  3003

TESTMAC FD TESTS:
    Numerology[0] Bandwidth[5]
          1001  6001  8001 10001 12001
    Numerology[0] Bandwidth[10]
          1001  2001  4001  6001  8001 10001 12001  1002  2002  4002
          6002  8002 10002 12002  1003
    Numerology[0] Bandwidth[20]
          1002  1004  1012  1014  1015  1016  1017  1018  1020  1021
          1022  1023  1024  1025  1030  1031  1032  1033  1200  1201
          1202  1206  1207  1208  1209  1210  1211  1212  1220  1221
          1222  1223  1224  1225  1226  1227  1228
    Numerology[1] Bandwidth[40]
          1001  1002  1003
    Numerology[1] Bandwidth[100]
          1001  1002  1200  1201  1202  1203  1204  1205  1206  1207
          1208  1209  1210  1300  1301  1302  1303  1304  1305  1306
          1307  1308  1350  1351  1352  1353  1354  1355  1356  1357
          1358  1359  1370  1371  1372  1373  1374  1375  1376  1377
          1378  1401  1402  1403  1404  1405  1406  1411  1412  1490
          1494  1500  1501  1502  1503  1504  1510  1511  1512  1513
          1514  1515  1520  1521  1522  1523  1524  1525  1526  1527
          1528  1529  1530  1531  1532  1540  1541  1700  1701  1702
          2520  2521  2522  2523  2524  2525  2526  2527  2528  2529
          2530  2531  2532  3524  3525  3526  3527  3528  3529  3530
          3531  3532  4524  4525  4526  4527  4528  4529  4530  4531
          4532
    Numerology[3] Bandwidth[100]
          1001  1002  1004  1005  1006  1007  1008  1009  1010  1011
          1012  1013  1014  1015  1061  1062  1063  1064  1065  1080
          1081  1082  2001  3001
    testmac_run_parse_file Parsing config file: ./icelake-
sp/icxsp_mu1_100mhz_mmimo_64x64_hton_xran.cfg
testmac_set_phy_start: mode[4], period[0], count[100200]
    Adding setoption pdsch_split [numTests: 0] [nCellMask: 0x00000001] [nOption: 4]
[pMacOptions: 260 / 0x00000104]
    Adding setoption pdsch_dl_weight_split [numTests: 0] [nCellMask: 0x00000001]
[nOption: 4] [pMacOptions: 260 / 0x00000104]
    Adding setoption pusch_chan_est_split [numTests: 0] [nCellMask: 0x00000001]
[nOption: 2] [pMacOptions: 258 / 0x00000102]
    Adding setoption pusch_mmse_split [numTests: 0] [nCellMask: 0x00000001] [nOption:
4] [pMacOptions: 260 / 0x00000104]
    Adding setoption pusch_llr_rx_split [numTests: 0] [nCellMask: 0x00000001]
[nOption: 2] [pMacOptions: 258 / 0x00000102]
    Adding setoption pusch_ul_weight_split [numTests: 0] [nCellMask: 0x00000001]
[nOption: 2] [pMacOptions: 258 / 0x00000102]
    Adding setoption timer_multi_cell [numTests: 0] [nCellMask: 0xffffffff] [nOption:
10000] [pMacOptions: 10000 / 0x00002710]
    Adding setoption fec_dec_num_iter [numTests: 0] [nCellMask: 0xffffffff] [nOption:
3] [pMacOptions: -253 / 0xffffff03]
    Adding SetCoreMask[numTests: 0][setCoreCnt: 0]. CoreMask[137170526192 /
0x0000001ff0001ff0]
```

```
    Adding SetDlbeamCoreMask[numTests: 0][setCoreCnt: 0]. CoreMask[2016 /
0x00000000000007e0]
    Adding SetSrsCoreMask[numTests: 0][setCoreCnt: 0]. CoreMask[268435472 /
0x0000000010000010]
Setting Testmac System Core: 2
Setting Testmac Run Core: 2
Setting Testmac Wls Core: 3
    Adding Test[3370]. NumCarr[3], Current Directory:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/
        Carrier[0]: ConfigFile: fd/mu1_100mhz/376/fd_testconfig_tst376.cfg
        Carrier[1]: ConfigFile: fd/mu1_100mhz/377/fd_testconfig_tst377.cfg
        Carrier[2]: ConfigFile: fd/mu1_100mhz/377/fd_testconfig_tst377.cfg

testmac_set_multi_cell_timer: 10000




-------------------------------------------------------------------------------------
---
Running Test[3370]. NumCarr[3], Current Directory:
/home/vzakharc/master/../master_aux/flexran_l1_5g_test/
Carrier[0]: ConfigFile: fd/mu1_100mhz/376/fd_testconfig_tst376.cfg
Carrier[1]: ConfigFile: fd/mu1_100mhz/377/fd_testconfig_tst377.cfg
Carrier[2]: ConfigFile: fd/mu1_100mhz/377/fd_testconfig_tst377.cfg
TESTMAC>welcome to application console

MLogRestart
MLogOpen: filename(testmac-mlog.bin) mlogSubframes (128), mlogCores(3),
mlogSize(2048) mlog_mask (-1)
    mlogSubframes (128), mlogCores(3), mlogSize(2048)
    localMLogTimerInit
        System clock (rdtsc)  resolution 1496525908 [Hz]
        Ticks per us 1496
    MLog Storage: 0x7f8208000900 -> 0x7f82080c1120 [ 788512 bytes ]
    localMLogFreqReg: 1496. Storing: 1496
    Mlog Open successful

testmac_mac2phy_set_num_cells: Setting Max Cells: 3
testmac_config_parse: test_num[3370] test_type[2] numcarrier[3]
Queueing MSG_TYPE_PHY_UL_IQ_SAMPLES(0)
Queueing MSG_TYPE_PHY_UL_IQ_SAMPLES(1)
Queueing MSG_TYPE_PHY_UL_IQ_SAMPLES(2)
Received MSG_TYPE_PHY_UL_IQ_SAMPLES(0)
Queueing MSG_TYPE_PHY_CONFIG_REQ(0)
Received MSG_TYPE_PHY_UL_IQ_SAMPLES(1)
Queueing MSG_TYPE_PHY_CONFIG_REQ(1)
Received MSG_TYPE_PHY_UL_IQ_SAMPLES(2)
Queueing MSG_TYPE_PHY_CONFIG_REQ(2) and sending list
Received MSG_TYPE_PHY_CONFIG_RESP(0)
Queueing MSG_TYPE_PHY_START_REQ(0)
Received MSG_TYPE_PHY_CONFIG_RESP(1)
Queueing MSG_TYPE_PHY_START_REQ(1)
Received MSG_TYPE_PHY_CONFIG_RESP(2)
Queueing MSG_TYPE_PHY_START_REQ(2) and sending list
Received MSG_TYPE_PHY_START_RESP(0)
Received MSG_TYPE_PHY_START_RESP(1)
Received MSG_TYPE_PHY_START_RESP(2)
==== testmac Time: 5000 ms NumCarrier: 3 Total Proc Time: [  0.00..  6.30.. 19.00]
usces====
    Core Utilization [Core: 3] [Util %:  0.42%]
```

```
==== testmac Time: 10000 ms NumCarrier: 3 Total Proc Time: [  6.00..116.80..206.00]
usces====
    Core Utilization [Core: 3] [Util %: 27.86%]
==== testmac Time: 20000 ms NumCarrier: 3 Total Proc Time: [ 10.00..156.33..260.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.31%]
==== testmac Time: 25000 ms NumCarrier: 3 Total Proc Time: [ 11.00..156.33..260.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.30%]
==== testmac Time: 30000 ms NumCarrier: 3 Total Proc Time: [ 11.00..156.44..256.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.32%]
==== testmac Time: 35000 ms NumCarrier: 3 Total Proc Time: [ 11.00..156.42..258.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.32%]
==== testmac Time: 40000 ms NumCarrier: 3 Total Proc Time: [ 11.00..156.45..258.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.33%]
==== testmac Time: 45000 ms NumCarrier: 3 Total Proc Time: [ 11.00..156.40..282.00]
usces====
    Core Utilization [Core: 3] [Util %: 32.32%]

TESTMAC>==== testmac Time: 50000 ms NumCarrier: 3 Total Proc Time: [
11.00..156.39..260.00] usces====
    Core Utilization [Core: 3] [Util %: 32.31%]
Received MSG_TYPE_PHY_STOP_RESP(0)
Queueing MSG_TYPE_PHY_SHUTDOWN_REQ(0)
Received MSG_TYPE_PHY_STOP_RESP(1)
Queueing MSG_TYPE_PHY_SHUTDOWN_REQ(1)
Received MSG_TYPE_PHY_STOP_RESP(2)
Queueing MSG_TYPE_PHY_SHUTDOWN_REQ(2) and sending list
Received MSG_TYPE_PHY_SHUTDOWN_RESP(2)
Received MSG_TYPE_PHY_SHUTDOWN_RESP(0)
Received MSG_TYPE_PHY_SHUTDOWN_RESP(1)
MLogPrint: ext_filename((null).bin)
    Opening MLog File: testmac-mlog-c0.bin
    MLog file testmac-mlog-c0.bin closed
    Mlog Print successful
Test[FD_mu1_100mhz_3370] Completed
wls_mac_free_list_all:
        nTotalBlocks[4056] nAllocBlocks[1010] nFreeBlocks[3046]
        nTotalAllocCnt[4538427] nTotalFreeCnt[4537417] Diff[1010]
        nDlBufAllocCnt[3609068] nDlBufFreeCnt[3609068] Diff[0]
        nUlBufAllocCnt[929359] nUlBufFreeCnt[928349] Diff[1010]

All Tests Completed, Total run 1 Tests, PASS 1 Tests, and FAIL 0 Tests
```

§

Appendix B PTP Configuration

B.1 PTP Synchronization

Precision Time Protocol (PTP) provides an efficient way to synchronize time on the network nodes. This protocol uses Master-Slave architecture. Grandmaster Clock (Master) is a reference clock for the other nodes, which adapt their clocks to the master.

Using Physical Hardware Clock (PHC) from the Grandmaster Clock, NIC port precision timestamp packets can be served for other network nodes. Slave nodes adjust their PHC to the master following the IEEE 1588 specification.

There are existing implementations of PTP protocol that are widely used in the industry. One of them is PTP for Linux, which is a set of tools providing necessary PTP functionality. There is no need to re-implement the 1588 protocol because PTP for Linux is precise and efficient enough to be used out of the box.

To meet O-RAN requirements, two tools from PTP for Linux package are required: `ptp4l` and `phc2sys`.

B.1.1 PTP for Linux* Requirements

PTP for Linux* introduces some software and hardware requirements. The machine on which the tools will be run needs to use at least a 3.10 Kernel version (built-in PTP support). Several Kernel options need to be enabled in Kernel configuration:

- `CONFIG_PPS`
- `CONFIG_NETWORK_PHY_TIMESTAMPING`
- `PTP_1588_CLOCK`

Be sure that the Kernel is compiled with these options.

For the best precision, PTP uses hardware timestamping. NIC has its own clock, called Physical Hardware Clock (PHC), to read current time just a moment before the packet is sent to minimalize the delays added by the Kernel processing the packet. Not every NIC supports that feature. To confirm that currently attached NIC support Hardware Timestamps, use ethtool with the command:

```
ethtool -T eth0
```

Where the `eth0` is the potential PHC port. The output from the command should say that there is Hardware Timestamps support.

To set up PTP for Linux*:

1. Download source code:

```
git clone http://git.code.sf.net/p/linuxptp/code linuxptp
git checkout v2.0
```

NOTE: Apply patch (this is required to work around an issue with some of the GM PTP packet sizes.)

```
diff --git a/msg.c b/msg.c
old mode 100644
new mode 100755
index d1619d4..40d1538
--- a/msg.c
+++ b/msg.c
@@ -399,9 +399,11 @@ int msg_post_recv(struct ptp_message *m, int cnt)
        port_id_post_recv(&m->pdelay_resp.requestingPortIdentity);
        break;
    case FOLLOW_UP:
+       cnt -= 4;
```

```
        timestamp_post_recv(m, &m->follow_up.preciseOriginTimestamp);
    break;
    case DELAY_RESP:
+       cnt -= 4;
        timestamp_post_recv(m, &m->delay_resp.receiveTimestamp);
        port_id_post_recv(&m->delay_resp.requestingPortIdentity);
        break;
```

2. Build and install `ptp4l`.

```
# make && make install
```

3. Modify `configs/default.cfg` to control frequency of Sync interval to 0.0625 s.

```
logSyncInterval     -4
```

B.1.2 ptp4l

This tool handles all PTP traffic on the provided NIC port and updated PHC. It also determines the Grandmaster Clock and tracks synchronization status. This tool can be run as a daemon or as a regular Linux* application. When the synchronization is reached, it gives output on the screen for precision tracking. The configuration file of ptp4l contains many options that can be set to get the best synchronization precision. Although, even with `default.cfg` the synchronization quality is excellent.

To start the synchronization process run:

```
cd linuxptp
./ptp4l -f ./configs/default.cfg -2 -i <if_name> -m
```

The output below shows what the output on non-master node should look like when synchronization is started. This means that PHC on this machine is synchronized to the master PHC.

```
ptp4l[1434165.358]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1434165.358]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[1434166.384]: port 1: new foreign master fcaf6a.fffe.029708-1
ptp4l[1434170.352]: selected best master clock fcaf6a.fffe.029708
ptp4l[1434170.352]: updating UTC offset to 37
ptp4l[1434170.352]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[1434171.763]: master offset      -5873 s0 freq  -18397 path delay      2778
ptp4l[1434172.763]: master offset      -6088 s2 freq  -18612 path delay      2778
ptp4l[1434172.763]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[1434173.763]: master offset      -5886 s2 freq  -24498 path delay      2732
ptp4l[1434174.763]: master offset        221 s2 freq  -20157 path delay      2728
ptp4l[1434175.763]: master offset       1911 s2 freq  -18401 path delay      2724
ptp4l[1434176.763]: master offset       1774 s2 freq  -17964 path delay      2728
ptp4l[1434177.763]: master offset       1198 s2 freq  -18008 path delay      2728
ptp4l[1434178.763]: master offset        746 s2 freq  -18101 path delay      2755
ptp4l[1434179.763]: master offset        218 s2 freq  -18405 path delay      2792
ptp4l[1434180.763]: master offset        103 s2 freq  -18454 path delay      2792
ptp4l[1434181.763]: master offset        -13 s2 freq  -18540 path delay      2813
ptp4l[1434182.763]: master offset          9 s2 freq  -18521 path delay      2813
ptp4l[1434183.763]: master offset         11 s2 freq  -18517 path delay      2813
```

B.1.3 phc2sys

The PHC clock is independent from the system clock. Synchronizing only PHC does not make the system clock exactly the same as the master. The xRAN library requires use of the system clock to determine a common point in time on two machines (O-DU and RU) to start transmission at the same moment and keep time frames defined by O-RAN Fronthaul specification.

This application keeps the system clock updated to PHC. It makes it possible to use POSIX timers as a time reference in xRAN application.

Run `phc2sys` with the command:

```
cd linuxptp
./phc2sys -s enp25s0f0 -w -m -R 8
```

Command output will look like:

```
ptp4l[1434165.342]: selected /dev/ptp4 as PTP
phc2sys[1434344.651]: CLOCK_REALTIME phc offset       450 s2 freq  -39119 delay   1354
phc2sys[1434344.776]: CLOCK_REALTIME phc offset       499 s2 freq  -38620 delay   1344
phc2sys[1434344.902]: CLOCK_REALTIME phc offset       485 s2 freq  -38484 delay   1347
phc2sys[1434345.027]: CLOCK_REALTIME phc offset       476 s2 freq  -38348 delay   1346
phc2sys[1434345.153]: CLOCK_REALTIME phc offset       392 s2 freq  -38289 delay   1340
phc2sys[1434345.278]: CLOCK_REALTIME phc offset       319 s2 freq  -38244 delay   1340
phc2sys[1434345.404]: CLOCK_REALTIME phc offset       278 s2 freq  -38190 delay   1349
phc2sys[1434345.529]: CLOCK_REALTIME phc offset       221 s2 freq  -38163 delay   1343
phc2sys[1434345.654]: CLOCK_REALTIME phc offset        97 s2 freq  -38221 delay   1342
phc2sys[1434345.780]: CLOCK_REALTIME phc offset        67 s2 freq  -38222 delay   1344
phc2sys[1434345.905]: CLOCK_REALTIME phc offset        68 s2 freq  -38201 delay   1341
phc2sys[1434346.031]: CLOCK_REALTIME phc offset       104 s2 freq  -38144 delay   1340
phc2sys[1434346.156]: CLOCK_REALTIME phc offset        58 s2 freq  -38159 delay   1340
phc2sys[1434346.281]: CLOCK_REALTIME phc offset        12 s2 freq  -38188 delay   1343
phc2sys[1434346.407]: CLOCK_REALTIME phc offset       -36 s2 freq  -38232 delay   1342
phc2sys[1434346.532]: CLOCK_REALTIME phc offset      -103 s2 freq  -38310 delay   1348
```

B.1.4 Configuration C3

```
Configuration C3 27 can be simulated for O-DU using a separate server acting as
Fronthaul Network and O-RU at the same time. O-RU server can be configured to relay PTP
and act as PTP master for O-DU. Settings below can be used to instantiate this scenario.
The difference is that on the O-DU side, the Fronthaul port can be used as the source of
PTP as well as for U-plane and C-plane traffic.
```

1. Follow the steps in Appendix B.1.1, PTP for Linux* Requirements to install PTP on the O-RU server.
2. Copy `configs/default.cfg` to `configs/default_slave.cfg` and modify the copied file as below:

```
diff --git a/configs/default.cfg b/configs/default.cfg
old mode 100644
new mode 100755
index e23dfd7..f1ecaf1
--- a/configs/default.cfg
+++ b/configs/default.cfg
@@ -3,26 +3,26 @@
 # Default Data Set
 #
 twoStepFlag            1
-slaveOnly              0
+slaveOnly              1
 priority1              128
-priority2              128
+priority2              255
 domainNumber           0
 #utc_offset            37
-clockClass             248
+clockClass             255
 clockAccuracy          0xFE
 offsetScaledLogVariance        0xFFFF
 free_running           0
 freq_est_interval      1
 dscp_event             0
 dscp_general           0
-dataset_comparison     ieee1588
```

```
+dataset_comparison     G.8275.x
 G.8275.defaultDS.localPriority 128
 maxStepsRemoved                255
 #
 # Port Data Set
 #
 logAnnounceInterval    1
-logSyncInterval                0
+logSyncInterval                -4
 operLogSyncInterval    0
 logMinDelayReqInterval 0
 logMinPdelayReqInterval        0
@@ -37,7 +37,7 @@ G.8275.portDS.localPriority   128
 asCapable               auto
 BMCA                    ptp
 inhibit_announce        0
-inhibit_pdelay_req      0
+#inhibit_pdelay_req      0
 ignore_source_id        0
 #
 # Run time options
```

3. Start slave port toward PTP GM:

```
./ptp4l -f ./configs/default_slave.cfg -2 -i enp25s0f0 -m
```

Example of output:

```
./ptp4l -f ./configs/default_slave.cfg -2 -i enp25s0f0 -m
ptp4l[3904470.256]: selected /dev/ptp6 as PTP clock
ptp4l[3904470.274]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[3904470.275]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[3904471.085]: port 1: new foreign master fcaf6a.fffe.029708-1
ptp4l[3904475.053]: selected best master clock fcaf6a.fffe.029708
ptp4l[3904475.053]: updating UTC offset to 37
ptp4l[3904475.053]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[3904477.029]: master offset        196 s0 freq  -18570 path delay      1109
ptp4l[3904478.029]: master offset        212 s2 freq  -18554 path delay      1109
ptp4l[3904478.029]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[3904479.029]: master offset         86 s2 freq  -18468 path delay      1109
ptp4l[3904480.029]: master offset         23 s2 freq  -18505 path delay      1124
ptp4l[3904481.029]: master offset          3 s2 freq  -18518 path delay      1132
ptp4l[3904482.029]: master offset       -169 s2 freq  -18689 path delay      1141
```

4. Synchronize local timer clock on O-RU for sample application

```
./phc2sys -s enp25s0f0 -w -m -R 8
```

Example of output:

```
./phc2sys -s enp25s0f0 -w -m -R 8
phc2sys[3904510.892]: CLOCK_REALTIME phc offset   343 s0 freq  -38967 delay   1530
phc2sys[3904511.017]: CLOCK_REALTIME phc offset   368 s2 freq  -38767 delay   1537
phc2sys[3904511.142]: CLOCK_REALTIME phc offset   339 s2 freq  -38428 delay   1534
phc2sys[3904511.267]: CLOCK_REALTIME phc offset   298 s2 freq  -38368 delay   1532
phc2sys[3904511.392]: CLOCK_REALTIME phc offset   239 s2 freq  -38337 delay   1534
phc2sys[3904511.518]: CLOCK_REALTIME phc offset   145 s2 freq  -38360 delay   1530
phc2sys[3904511.643]: CLOCK_REALTIME phc offset   106 s2 freq  -38355 delay   1527
phc2sys[3904511.768]: CLOCK_REALTIME phc offset   -30 s2 freq  -38459 delay   1534
phc2sys[3904511.893]: CLOCK_REALTIME phc offset   -92 s2 freq  -38530 delay   1530
phc2sys[3904512.018]: CLOCK_REALTIME phc offset  -173 s2 freq  -38639 delay   1528
phc2sys[3904512.143]: CLOCK_REALTIME phc offset  -246 s2 freq  -38764 delay   1530
phc2sys[3904512.268]: CLOCK_REALTIME phc offset  -300 s2 freq  -38892 delay   1532
```

5. Modify `configs/default.cfg` as shown below to run PTP master on Fronthaul of O-RU.

```
diff --git a/configs/default.cfg b/configs/default.cfg
old mode 100644
new mode 100755
index e23dfd7..c9e9d4c
--- a/configs/default.cfg
+++ b/configs/default.cfg
@@ -15,14 +15,14 @@ free_running                0
 freq_est_interval       1
 dscp_event              0
 dscp_general            0
-dataset_comparison      ieee1588
+dataset_comparison      G.8275.x
 G.8275.defaultDS.localPriority 128
 maxStepsRemoved                255
 #
 # Port Data Set
 #
 logAnnounceInterval     1
-logSyncInterval                0
+logSyncInterval                -4
 operLogSyncInterval     0
 logMinDelayReqInterval 0
 logMinPdelayReqInterval        0
@@ -37,7 +37,7 @@ G.8275.portDS.localPriority   128
 asCapable               auto
 BMCA                    ptp
 inhibit_announce        0
-inhibit_pdelay_req      0
+#inhibit_pdelay_req      0
 ignore_source_id        0
 #
 # Run time options
```

6. Start PTP master toward O-DU:

```
./ptp4l -f ./configs/default.cfg -2 -i enp175s0f1 -m
```

Example of output:

```
./ptp4l -f ./configs/default.cfg -2 -i enp175s0f1 -m
ptp4l[3903857.249]: selected /dev/ptp3 as PTP clock
ptp4l[3903857.266]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[3903857.267]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[3903863.734]: port 1: LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[3903863.734]: selected local clock 3cfdfe.fffe.bd005d as best master
ptp4l[3903863.734]: assuming the grand master role
```

7. Synchronize local NIC PTP master clock to local NIC PTP slave clock.

```
./phc2sys -c enp175s0f1 -s enp25s0f0 -w -m -R 8
```

Example of output:

```
./phc2sys -c enp175s0f1 -s enp25s0f0 -w -m -R 8
phc2sys[3904600.332]: enp175s0f1 phc offset      2042 s0 freq   -2445 delay   4525
phc2sys[3904600.458]: enp175s0f1 phc offset      2070 s2 freq   -2223 delay   4506
phc2sys[3904600.584]: enp175s0f1 phc offset      2125 s2 freq     -98 delay   4505
phc2sys[3904600.710]: enp175s0f1 phc offset      1847 s2 freq    +262 delay   4518
phc2sys[3904600.836]: enp175s0f1 phc offset      1500 s2 freq    +469 delay   4515
phc2sys[3904600.961]: enp175s0f1 phc offset      1146 s2 freq    +565 delay   4547
phc2sys[3904601.086]: enp175s0f1 phc offset       877 s2 freq    +640 delay   4542
phc2sys[3904601.212]: enp175s0f1 phc offset       517 s2 freq    +543 delay   4517
phc2sys[3904601.337]: enp175s0f1 phc offset       189 s2 freq    +370 delay   4510
phc2sys[3904601.462]: enp175s0f1 phc offset      -125 s2 freq    +113 delay   4554
phc2sys[3904601.587]: enp175s0f1 phc offset      -412 s2 freq    -212 delay   4513
```

```
phc2sys[3904601.712]: enp175s0f1 phc offset      -693 s2 freq    -617 delay   4519
phc2sys[3904601.837]: enp175s0f1 phc offset      -878 s2 freq   -1009 delay   4515
phc2sys[3904601.962]: enp175s0f1 phc offset      -965 s2 freq   -1360 delay   4518
phc2sys[3904602.088]: enp175s0f1 phc offset     -1048 s2 freq   -1732 delay   4510
phc2sys[3904602.213]: enp175s0f1 phc offset     -1087 s2 freq   -2086 delay   4531
phc2sys[3904602.338]: enp175s0f1 phc offset     -1014 s2 freq   -2339 delay   4528
phc2sys[3904602.463]: enp175s0f1 phc offset     -1009 s2 freq   -2638 delay   4531
```

8. On O-DU Install PTP for Linux tools from source code the same way as on O-RU above but no need to apply the patch for `msg.c`
9. Start slave port toward PTP master from O-RU using the same `default_slave.cfg` as on O-RU (see above)

```
./ptp4l -f ./configs/default_slave.cfg -2 -i enp181s0f0 -m
```

Example of output:

```
./ptp4l -f ./configs/default_slave.cfg -2 -i enp181s0f0 -m
ptp4l[809092.918]: selected /dev/ptp6 as PTP clock
ptp4l[809092.934]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[809092.934]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[809092.949]: port 1: new foreign master 3cfdfe.fffe.bd005d-1
ptp4l[809096.949]: selected best master clock 3cfdfe.fffe.bd005d
ptp4l[809096.950]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[809098.363]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[809099.051]: rms 38643 max 77557 freq  +719 +/- 1326 delay  1905 +/-   0
ptp4l[809100.051]: rms 1134 max 1935 freq  -103 +/- 680 delay  1891 +/-   4
ptp4l[809101.051]: rms  453 max  855 freq  +341 +/- 642 delay  1888 +/-   0
ptp4l[809102.052]: rms  491 max  772 freq +1120 +/- 752 delay  1702 +/-   0
ptp4l[809103.052]: rms  423 max  654 freq +1352 +/- 653 delay  1888 +/-   0
ptp4l[809104.052]: rms  412 max  579 freq +1001 +/- 672 delay  1702 +/-   0
ptp4l[809105.053]: rms  441 max  672 freq  +807 +/- 709 delay  1826 +/-  88
ptp4l[809106.053]: rms  422 max  607 freq +1353 +/- 636 delay  1702 +/-   0
ptp4l[809107.054]: rms  401 max  466 freq  +946 +/- 646 delay  1702 +/-   0
ptp4l[809108.055]: rms  401 max  502 freq  +912 +/- 659
```

10. Synchronize local clock on O-DU for sample application or l1 application.

```
./phc2sys -s enp181s0f0 -w -m -R 8
```

Example of output:

```
./phc2sys -s enp181s0f0 -w -m -R 8
phc2sys[809127.123]: CLOCK_REALTIME phc offset    675 s0 freq  -37379 delay   1646
phc2sys[809127.249]: CLOCK_REALTIME phc offset    696 s2 freq  -37212 delay   1654
phc2sys[809127.374]: CLOCK_REALTIME phc offset    630 s2 freq  -36582 delay   1648
phc2sys[809127.500]: CLOCK_REALTIME phc offset    461 s2 freq  -36562 delay   1642
phc2sys[809127.625]: CLOCK_REALTIME phc offset    374 s2 freq  -36510 delay   1643
phc2sys[809127.751]: CLOCK_REALTIME phc offset    122 s2 freq  -36650 delay   1649
phc2sys[809127.876]: CLOCK_REALTIME phc offset     34 s2 freq  -36702 delay   1650
phc2sys[809128.002]: CLOCK_REALTIME phc offset   -112 s2 freq  -36837 delay   1645
phc2sys[809128.127]: CLOCK_REALTIME phc offset   -160 s2 freq  -36919 delay   1643
phc2sys[809128.252]: CLOCK_REALTIME phc offset   -270 s2 freq  -37077 delay   1657
phc2sys[809128.378]: CLOCK_REALTIME phc offset   -285 s2 freq  -37173 delay   1644
phc2sys[809128.503]: CLOCK_REALTIME phc offset   -349 s2 freq  -37322 delay   1644
phc2sys[809128.629]: CLOCK_REALTIME phc offset   -402 s2 freq  -37480 delay   1641
phc2sys[809128.754]: CLOCK_REALTIME phc offset   -377 s2 freq  -37576 delay   1648
phc2sys[809128.879]: CLOCK_REALTIME phc offset   -467 s2 freq  -37779 delay   1650
phc2sys[809129.005]: CLOCK_REALTIME phc offset   -408 s2 freq  -37860 delay   1648
phc2sys[809129.130]: CLOCK_REALTIME phc offset   -480 s2 freq  -38054 delay   1655
phc2sys[809129.256]: CLOCK_REALTIME phc offset   -350 s2 freq  -38068 delay   1650
```

B.1.5 Support in xRAN Library

The xRAN library provides an API to check whether PTP for Linux is running correctly. There is a function called `xran_is_synchronized()`. It checks if `ptp4l` and `phc2sys` are running in the system by making PMC tool requests for the current port state and comparing it with the expected value. This verification should be done before initialization.

NOTE: "SLAVE" is the only expected value in this release; only a non-master scenario is supported currently.

§

Appendix C eCPRI DDP Profile for Columbiaville (Experimental Feature)

C.1 Introduction

The Intel® Ethernet 800 Series is the next generation of Intel® Ethernet Controllers and Network Adapters. The Intel® Ethernet 800 Series is designed with an enhanced programmable pipeline, allowing deeper and more diverse protocol header processing. This on-chip capability is called Dynamic Device Personalization (DDP). In the Intel® Ethernet 800 Series, a DDP profile is loaded dynamically on driver load per device.

A general-purpose DDP package is automatically installed with all supported Intel® Ethernet 800 Series drivers on Windows*, ESX*, FreeBSD*, and Linux* operating systems, including those provided by the Data Plane Development Kit (DPDK). This general-purpose DDP package is known as the OS-default package.

For more information on DDP technology in the Intel® Ethernet 800 Series products and the OS-default package, refer to the Intel® Ethernet Controller E810 Dynamic Device Personalization (DDP) Technology Guide, published here: https://cdrdv2.intel.com/v1/dl/getContent/617015.

This document describes an optional DDP package targeted towards the needs of Wireless and Edge (Wireless Edge) customers. This Wireless Edge DDP package (v1.3.22.101) adds support for eCPRI protocols in addition to the protocols in the OS-default package. The Wireless Edge DDP package is supported by DPDK.

Starting from DPDK 21.02 drivers and in the future will also be supported by the Intel® Ethernet 800 Series ice driver. on Linux operating systems. The Wireless DDP Package can be loaded on all Intel® Ethernet 800 Series devices, or different packages can be selected via serial number per device.

C.2 Software/Firmware Requirements

The specific DDP package requires certain firmware and DPDK versions and Intel® Ethernet 800 Series firmware/NVM versions. Support for eCPRI DDP profile included starting from Columbiaville (CVL)release 2.4 or later. The required DPDK version contains the support of loading the specific Wireless Edge DDP package.

- Intel® Ethernet 800 Series Linux Driver (ice) — 1.4.0 (or later)
- Wireless Edge DDP Package version (ice_wireless_edge) — 1.3.22.101
- Intel® Ethernet 800 Series firmware version — 1.5.4.2 (or later)
- Intel® Ethernet 800 Series NVM version — 2.4 (or later)
- DPDK version— 21.02 (or later)

NOTE: For FlexRAN release 21.03, corresponding support of CVL 2.4 driver pack and DPDK 21.02 is “experimental” and subject to additional testing and potential changes.

C.3 DDP Package Setup

The Intel® Ethernet 800 Series Comms DDP package supports only Linux-based operating systems currently.

Currently, the eCPRI is fully supported only by DPDK 21.02. It can be loaded either by DPDK or the Intel® Ethernet 800 Series Linux base driver.

C.3.1 Wireless Edge DDP Package

For details on how to set up DPDK, refer to Intel® Ethernet Controller E810 Data Plane Development Kit (DPDK) Configuration Guide (Doc ID: 633514).

There are two methods where DDP package can be loaded and used under DPDK (see Section C.3.2 and Section C.3.2). For both methods, the user must obtain the ice_wireless_edge-1.3.22.101.pkg or later from Intel (please contact your Intel representative for more information)

C.3.2 Option 1: *ice* Linux Base Driver

The first option is to have the ice Linux base driver load the package.

The *ice* Linux base driver looks for the symbolic link *intel/ice/ddp/ice.pkg* under the default firmware search path, checking the following folders in order:

- */lib/firmware/updates/*
- */lib/firmware/*

To install the Comms package, copy the extracted .pkg file and its symbolic link to */lib/firmware/updates/intel/ice/ddp* as follows, and reload the ice driver:

```
# cp /usr/tmp/ice_wireless_edge-1.3.22.101.pkg /lib/firmware/updates/intel/ice/ddp/
# ln -sf /lib/firmware/updates/intel/ice/ddp/ice_wireless_edge-1.3.22.101.pkg
/lib/firmware/updates/intel/ice/ddp/ice.pkg
# modprobe -r irdma
# modprobe -r ice
# modprobe ice
```

The kernel message log (*dmesg*) indicates status of package loading in the system. If the driver successfully finds and loads the DDP package, *dmesg* indicates that the DDP package successfully loaded. If not, the driver transitions to safe mode.

Once the driver loads the package, the user can unbind the *ice* driver from a desired port on the device so that DPDK can utilize the port.

The following example unbinds Port 0 and Port 1 of device on Bus 6, Device 0. Then, the port is bound to either igb_uio or vfio-pci.

```
# ifdown <interface>
# dpdk-devbind -u 06:00.0
# dpdk-devbind -u 06:00.1
# dpdk-devbind -b igb_uio 06:00.0 06:00.1
```

C.3.3 Option 2: DPDK Driver Only

The second method is if the system does not have the *ice* driver installed. In this case, the user can download the DDP package from the Intel download center and extract the zip file to obtain the package (*.pkg*) file. Similar to the Linux base driver, the DPDK driver looks for the *intel/ddp/ice.pkg* symbolic link in the kernel default firmware search path */lib/firmware/updates and /lib/firmware/*.

Copy the extracted DDP *.pkg* file and its symbolic link to */lib/firmware/intel/ice/ddp*, as follows.

```
# cp /usr/tmp/ice_wireless_edge-1.3.22.101 /lib/firmware/intel/ice/ddp/
# cp /usr/tmp/ice.pkg /lib/firmware/intel/ice/ddp/
```

When DPDK driver loads, it looks for *ice.pkg* to load. If the file exists, the driver downloads it into the device. If not, the driver transitions into safe mode.

C.3.4 Loading DDP Package to a Specific Intel® Ethernet 800 Series Device

On a host system running with multiple Intel® Ethernet 800 Series devices, there is sometimes a need to load a specific DDP package on a selected device while loading a different package on the remaining devices.

The Intel® Ethernet 800 Series Linux base driver and DPDK driver can both load a specific DDP package to a selected adapter based on the device's serial number. The driver does this by looking for a specific symbolic link package filename containing the selected device's serial number.

The following example illustrates how a user can load a specific package (e.g., *ice_wireless_edge-1.3.22.101*) on the device of Bus 6.

1. Find device serial number.

   To view bus, device, and function of all Intel® Ethernet 800 Series Network Adapters in the system:

```
# lspci | grep -i Ethernet | grep -i Intel
06:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev
01)
06:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev
01)
82:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP (rev
01)
82:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP (rev
01)
```

```
82:00.2 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP (rev
01)
82:00.3 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP (rev
01)
```

Use the **lspci** command to obtain the selected device serial number:

```
# lspci -vv -s 06:00.0 | grep -i Serial
Capabilities: [150 v1] Device Serial Number 35-11-a0-ff-ff-ca-05-68
```

Or, fully parsed without punctuation:

```
# lspci -vv -s 06:00.0 |grep Serial |awk '{print $7}'|sed s/-//g 3511a0ffffca0568
```

2. Rename the package file with the device serial number in the name.

   Copy the specific package over to /lib/firmware/updates/intel/ice/ddp (or /lib/firmware/intel/ice/ ddp) and create a symbolic link with the serial number linking to the package, as shown. The specific symbolic link filename starts with "ice-" followed by the device serial in lower case without dash ('-').

```
# ln -s /lib/firmware/updates/intel/ice/ddp/ice_wireless_edge-1.3.22.101.pkg
/lib/firmware/updates/intel/ice/ddp/ice-3511a0ffffca0568.pkg
```

3. If using Linux kernel driver (*ice*), reload the base driver (not required if using only DPDK driver).

```
# rmmod ice
# modprobe ice
```

   The driver loads the specific package to the selected device and the OS-default package to the remaining Intel® Ethernet 800 Series devices in the system.

4. Verify.

C.3.5 For kernel driver:

Example of output of successful load of Wireless Edge Package to all devices:

```
# dmesg | grep -i "ddp \| safe"
[606960.921404] ice 0000:18:00.0: The DDP package was successfully loaded: ICE
Wireless Edge Package version 1.3.22.101
[606961.672999] ice 0000:18:00.1: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
[606962.439067] ice 0000:18:00.2: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
[606963.198305] ice 0000:18:00.3: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
[606964.252076] ice 0000:51:00.0: The DDP package was successfully loaded: ICE
Wireless Edge Package version 1.3.22.101
[606965.017082] ice 0000:51:00.1: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
[606965.802115] ice 0000:51:00.2: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
[606966.576517] ice 0000:51:00.3: DDP package already present on device: ICE Wireless
Edge Package version 1.3.22.101
```

C.3.6 If using only DPDK driver:

Verify using DPDK's **testpmd** application to indicate the status and version of the loaded DDP package.

Example of eCPRI config with dpdk-testpmd

16 O-RAN eCPRI IQ streams mapped to 16 independent HW queues each.

```
#./dpdk-testpmd -l 22-25 -n 4 -a 0000:af:01.0 -- -i  --rxq=16 --txq=16 --cmdline-
file=/home/flexran_xran/ddp.txt

cat /home/flexran_xran/ddp.txt
port stop 0
port config mtu 0 9600
port config 0 rx_offload vlan_strip on
port start 0
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0000 / end
actions queue index 0 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0001 / end
actions queue index 1 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0002 / end
actions queue index 2 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0003 / end
actions queue index 3 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0004 / end
actions queue index 4 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0005 / end
actions queue index 5 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0006 / end
actions queue index 6 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0007 / end
actions queue index 7 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0008 / end
actions queue index 8 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x0009 / end
actions queue index 9 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000a / end
actions queue index 10 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000b / end
actions queue index 11 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000c / end
actions queue index 12 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000d / end
actions queue index 13 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000e / end
actions queue index 14 / mark / end
flow create 0 ingress pattern eth / ecpri common type iq_data pc_id is 0x000f / end
actions queue index 15 / mark / end
set fwd rxonly
start
show fwd stats all
```

C.4 O-RAN Front haul eCPRI

Intel® Ethernet 800 Series DDP capabilities support several functionalities important for the O-RAN FH.

- RSS for packet steering based on ecpriMessage
- RSS for packet steering based on ecpriRtcid/ecpriPcid
- Queue mapping based on ecpriRtcid/ecpriPcid

- Queue mapping based on ecpriMessage

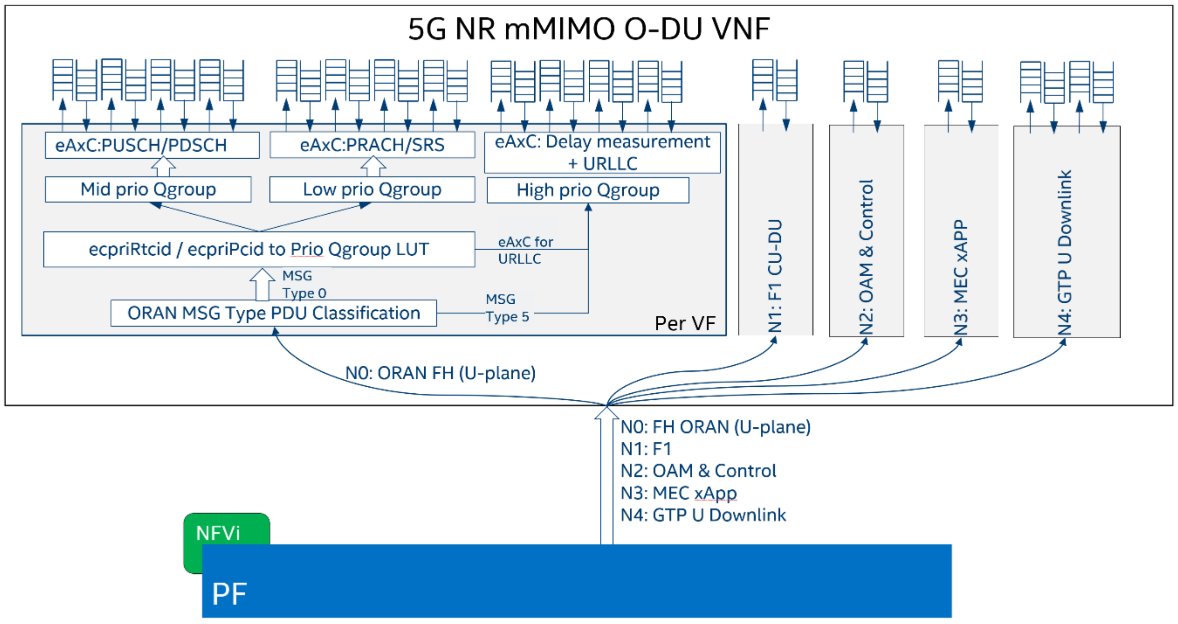


Figure 30. O-RAN FH VNF

Table 13. Patterns & Input Sets for Flow Director and RSS (DPDK 21.02)

Pattern	Input Set
ETH / VLAN / eCPRI	ecpriMessage | ecpriRtcid/ecpriPcid
ETH / VLAN /IPv4(6)/UDP/eCPRI	ecpriMessage | ecpriRtcid/ecpriPcid (*)

Note: * IP/UDP is not used with FlexRAN

C.4.1 Limitations

DPDK 21.02 allows up to 1024 queues per VF and RSS across up to 64 receive queues.

C.5 RTE Flow API

The DPDK Generic flow API (rte_flow) will be used to the configure the Intel® Ethernet 800 Series to match specific ingress traffic and forward it to specified queues.

For further information, please refer to section 11 of the DPDK Programmers guide.

The specific ingress traffic is identified by a matching pattern which is composed of one or more Pattern items (represented by struct rte_flow_item). Once a match has been determined one or more associated Actions (represented by struct rte_flow_action) will be performed.

A number of flow rules can be combined such that one rule directs traffic to a queue group based on *ecpriMessage/ecpriRtcid/ecpriPcid* etc. and a second rule distributes matching packets within that queue group using RSS.

The following subset of the RTE Flow API functions can be used to validate, create and destroy RTE Flow rules.

C.5.1 RTE Flow Rule Validation

A RTE Flow rule is created via a call to the function *rte_flow_validate*. This can be used to check the rule for correctness and whether it would be accepted by the device given sufficient resources.

```
int     rte_flow_validate(uint16_t port_id,
                         const struct rte_flow_attr *attr,
                         const struct rte_flow_item pattern[],
                         const struct rte_flow_action *actions[]
                         struct rte_flow_error *error);
```

port_id	:	port identifier of Ethernet device
attr	:	flow rule attributes(ingress/egress)
pattern	:	pattern specification (list terminated by the END pattern item).
action	:	associated actions (list terminated by the END action).
error	:	perform verbose error reporting if not NULL.

0 is returned upon success, negative errno otherwise.

C.5.2 RTE Flow Rule Creation

A RTE Flow rule is created via a call to the function *rte_flow_create*.

```
struct rte_flow * rte_flow_create(uint16_t port_id,
                                  const struct rte_flow_attr *attr,
                                  const struct rte_flow_item pattern[],
                                  const struct rte_flow_action *actions[]
                                  struct rte_flow_error *error);
```

port_id	:	port identifier of Ethernet device
attr	:	flow rule attributes(ingress/egress)
pattern	:	pattern specification (list terminated by the END pattern item).
action	:	associated actions (list terminated by the END action).
error	:	perform verbose error reporting if not NULL.

A valid handle is returned upon success, NULL otherwise.

C.5.3 RTE Flow Rule Destruction

A RTE Flow rule is destroyed via a call to the function *rte_flow_destroy*.

```
int      rte_flow_destroy(uint16_t port_id,
                   struct rte_flow *flow,
                   struct rte_flow_error *error);
```

port_id	:	port identifier of Ethernet device
flow	:	flow rule handle to destroy.
error	:	perform verbose error reporting if not NULL.

0 is returned upon success, negative errno otherwise.

C.5.4 RTE Flow Flush

All flow rule handles associated with a port can be released using *rte_flow_flush*. They are released as with successive calls to function *rte_flow_destroy* .

```
int      rte_flow_flush(uint16_t port_id,
                          struct rte_flow_error *error);
```

port_id	:	port identifier of Ethernet device
error	:	perform verbose error reporting if not NULL.

0 is returned upon success, negative errno otherwise.

C.5.5 RTE Flow Query

A RTE Flow rule is queried via a call to the function *rte_flow_query* .

```
int rte_flow_query(uint16_t port_id,
                   struct rte_flow *flow,
                   const struct rte_flow_action *action,
                   void *data,
                   struct rte_flow_error *error);
```

port_id	:	port identifier of Ethernet device
flow	:	flow rule handle to query
action	:	action to query, this must match prototype from flow rule.
data	:	pointer to storage for the associated query data type
error	:	perform verbose error reporting if not NULL.

0 is returned upon success, negative errno otherwise.

C.6 RTE Flow Rules

A flow rule is the combination of attributes with a matching pattern and a list of actions. Each flow rules consists of:

- **Attributes (represented by *struct rte_flow_attr*):** properties of a flow rule such as its direction (ingress or egress) and priority.
- **Pattern Items (represented by *struct rte_flow_item*):** is part of a matching pattern that either matches specific packet data or traffic properties.
- **Matching pattern:** traffic properties to look for, a combination of any number of items.
- **Actions (represented by *struct rte_flow_action*):** operations to perform whenever a packet is matched by a pattern.

C.6.1 Attributes

Flow rule patterns apply to inbound and/or outbound traffic. For the purposes described in later sections the rules apply to ingress only. For further information, please refer to section 11 of the DPDK Programmers guide .

```
struct rte_flow_attr {
        uint32_t group;
        uint32_t priority;
        uint32_t ingress:1;
        uint32_t egress:1;
        uint32_t transfer:1;
        uint32_t reserved:29;
};
```

C.6.2 Pattern items

For the purposes described in later sections Pattern items are primarily for matching protocol headers and packet data, usually associated with a specification structure. These must be stacked in the same order as the protocol layers to match inside packets, starting from the lowest.

Item specification structures are used to match specific values among protocol fields (or item properties).

Up to three structures of the same type can be set for a given item:

- **spec:** values to match (e.g. a given IPv4 address).
- **last:** upper bound for an inclusive range with corresponding fields in spec.
- **mask:** bit-mask applied to both spec and last whose purpose is to distinguish the values to take into account and/or partially mask them out (e.g. in order to match an IPv4 address prefix).

Table 14. Example RTE FLOW Item Types

Item Type*	Description	Specification Structure
END	End marker for item lists	None
VOID	Used as a placeholder for convenience	None
ETH	Matches an Ethernet header	rte_flow_item_eth
VLAN	Matches an 802.1Q/ad VLAN tag.	rte_flow_item_vlan
IPV4	Matches an IPv4 header	rte_flow_item_ipv4

IPV6	Matches an IPv6 header	rte_flow_item_ipv6
ICMP	Matches an ICMP header.	rte_flow_item_icmp
UDP	Matches an UDP header.	rte_flow_item_udp
TCP	Matches a TCP header.	rte_flow_item_tcp
SCTP	Matches a SCTP header.	rte_flow_item_sctp
VXLAN	Matches a VXLAN header.	rte_flow_item_vxlan
NVGRE	Matches a NVGRE header.	rte_flow_item_nvgre
ECPRI	Matches ECPRI Header	rte_flow_item_ecpri

```
RTE_FLOW_ITEM_TYPE_ETH

struct rte_flow_item_eth {
        struct rte_ether_addr dst; /**< Destination MAC. */
        struct rte_ether_addr src; /**< Source MAC. > */
        rte_be16_t type; /**< EtherType or TPID.> */
};

struct rte_ether_addr {
        uint8_t addr_bytes[RTE_ETHER_ADDR_LEN]; /**< Addr bytes in tx order */
}
```

```
RTE_FLOW_ITEM_TYPE_IPV4

struct rte_flow_item_ipv4 {
        struct rte_ipv4_hdr hdr; /**< IPv4 header definition. */
};

struct rte_ipv4_hdr {
        uint8_t  version_ihl;           /**< version and header length */
        uint8_t  type_of_service;       /**< type of service */
        rte_be16_t total_length;        /**< length of packet */
        rte_be16_t packet_id;           /**< packet ID */
        rte_be16_t fragment_offset;     /**< fragmentation offset */
        uint8_t  time_to_live;          /**< time to live */
        uint8_t  next_proto_id;         /**< protocol ID */
        rte_be16_t hdr_checksum;        /**< header checksum */
        rte_be32_t src_addr;            /**< source address */
        rte_be32_t dst_addr;            /**< destination address */
}

RTE_FLOW_ITEM_TYPE_UDP

struct rte_flow_item_udp {
        struct rte_udp_hdr hdr; /**< UDP header definition. */
};

struct rte_udp_hdr {
        rte_be16_t src_port;    /**< UDP source port. */
        rte_be16_t dst_port;    /**< UDP destination port. */
        rte_be16_t dgram_len;   /**< UDP datagram length */
        rte_be16_t dgram_cksum; /**< UDP datagram checksum */
}
```

```
RTE_FLOW_ITEM_TYPE_ECPRI

struct rte_flow_item_ecpri {
        struct rte_ecpri_combined_msg_hdr hdr;
};

struct rte_ecpri_combined_msg_hdr {
        struct rte_ecpri_common_hdr common;
        union {
                struct rte_ecpri_msg_iq_data type0;
                struct rte_ecpri_msg_bit_seq type1;
                struct rte_ecpri_msg_rtc_ctrl type2;
                struct rte_ecpri_msg_bit_seq type3;
                struct rte_ecpri_msg_rm_access type4;
                struct rte_ecpri_msg_delay_measure type5;
                struct rte_ecpri_msg_remote_reset type6;
                struct rte_ecpri_msg_event_ind type7;
                rte_be32_t dummy[3];
        };
};
struct rte_ecpri_common_hdr {
        union {
                rte_be32_t u32;  /**< 4B common header in BE */
                struct {
#if RTE_BYTE_ORDER == RTE_LITTLE_ENDIAN
                        uint32_t size:16; /**< Payload Size */
                        uint32_t type:8; /**< Message Type */
                        uint32_t c:1; /**< Concatenation Indicator */
                        uint32_t res:3; /**< Reserved */
                        uint32_t revision:4; /**< Protocol Revision */
#elif RTE_BYTE_ORDER == RTE_BIG_ENDIAN
                        uint32_t revision:4; /**< Protocol Revision */
                        uint32_t res:3; /**< Reserved */
                        uint32_t c:1;  /**< Concatenation Indicator */
                        uint32_t type:8; /**< Message Type */
                        uint32_t size:16; /**< Payload Size */
#endif
                };
        };
};
/**
 * eCPRI Message Header of Type #0: IQ Data
 */
struct rte_ecpri_msg_iq_data {
        rte_be16_t pc_id;                           /**< Physical channel ID */
        rte_be16_t seq_id;                          /**< Sequence ID */
};

/**
 * eCPRI Message Header of Type #1: Bit Sequence
 */
struct rte_ecpri_msg_bit_seq {
        rte_be16_t pc_id;                           /**< Physical channel ID */
        rte_be16_t seq_id;                          /**< Sequence ID */
};

/**
 * eCPRI Message Header of Type #2: Real-Time Control Data
 */
struct rte_ecpri_msg_rtc_ctrl {
        rte_be16_t rtc_id;                              /**< Real-Time Control Data ID */
        rte_be16_t seq_id;                              /**< Sequence ID */
};

/**
 * eCPRI Message Header of Type #3: Generic Data Transfer
 */
struct rte_ecpri_msg_gen_data {
        rte_be32_t pc_id;                               /**< Physical channel ID */
        rte_be32_t seq_id;                              /**< Sequence ID */
};

/**
 * eCPRI Message Header of Type #4: Remote Memory Access
 */
RTE_STD_C11
struct rte_ecpri_msg_rm_access {
#if RTE_BYTE_ORDER == RTE_LITTLE_ENDIAN
        uint32_t ele_id:16;                             /**< Element ID */
        uint32_t rr:4;                                  /**< Req/Resp */
        uint32_t rw:4;                                  /**< Read/Write */
        uint32_t rma_id:8;                              /**< Remote Memory Access ID */
#elif RTE_BYTE_ORDER == RTE_BIG_ENDIAN
        uint32_t rma_id:8;                              /**< Remote Memory Access ID */
        uint32_t rw:4;                                  /**< Read/Write */
        uint32_t rr:4;                                  /**< Req/Resp */
        uint32_t ele_id:16;                             /**< Element ID */
#endif
        uint8_t addr[6];            /**< 48-bits address */
        rte_be16_t length;                              /**< number of bytes */
};

/**
 * eCPRI Message Header of Type #5: One-Way Delay Measurement
 */
struct rte_ecpri_msg_delay_measure {
        uint8_t msr_id;                                 /**< Measurement ID */
        uint8_t act_type;                               /**< Action Type */
};

/**
 * eCPRI Message Header of Type #6: Remote Reset
 */
struct rte_ecpri_msg_remote_reset {
        rte_be16_t rst_id;                              /**< Reset ID */
        uint8_t rst_op;                                 /**< Reset Code Op */
};

/**
 * eCPRI Message Header of Type #7: Event Indication
 */
struct rte_ecpri_msg_event_ind {
        uint8_t evt_id;                                 /**< Event ID */
        uint8_t evt_type;                               /**< Event Type */
        uint8_t seq;                                    /**< Sequence Number */
        uint8_t number;                                 /**< Number of Faults/Notif */
};
```

C.6.3 Matching Patterns

A matching pattern is formed by stacking items starting from the lowest protocol layer to match. Patterns are terminated by END pattern item.

Actions

Each possible action is represented by a type. An action can have an associated configuration object. Actions are terminated by the END action.

Table 15. RTE FLOW Actions

Action*	Description	Configuration Structure
END	End marker for action lists	none
VOID	Used as a placeholder for convenience	none
PASSTHRU	Leaves traffic up for additional processing by subsequent flow rules; makes a flow rule non-terminating.	none
MARK	Attaches an integer value to packets and sets PKT_RX_FDIR and PKT_RX_FDIR_ID mbuf flags	rte_flow_action_mark
QUEUE	Assigns packets to a given queue index	rte_flow_action_queue
DROP	Drops packets	none
COUNT	Enables Counters for this flow rule	rte_flow_action_count
RSS	Similar to QUEUE, except RSS is additionally performed on packets to spread them among several queues according to the provided parameters.	rte_flow_action_rss
VF	Directs matching traffic to a given virtual function of the current device	rte_flow_action_vf

C.6.4 Route to specific Queue id based on ecpriRtcid/ecpriPcid

An RTE Flow Rule will be created to match an eCPRI packet with a specific pc_id value and route it to specified queues.

Pattern Items

Table 16. Pattern Items to match eCPRI packet with a Specific Physical Channel ID (pc_id)

Index	Item	Spec	Mask
0	Ethernet	0	0

1	eCPRI	hdr.common.type = RTE_ECPRI_MSG_TYPE_IQ_DATA;
		hdr.type0.pc_id = pc_id;
			hdr.common.type = 0xff;
			hdr.type0.pc_id = 0xffff;

2	END	0	0

The following code sets up the *RTE_FLOW_ITEM_TYPE_ETH* and *RTE_FLOW_ITEM_TYPE_ECPRI* Pattern Items.

The *RTE_FLOW_ITEM_TYPE_ECPRI* Pattern is configured to match on the pc_id value (in this case 8 converted to Big Endian byte order).

```
uint8_t pc_id_be = 0x0800;
#define MAX_PATTERN_NUM            3
struct rte_flow_item pattern[MAX_PATTERN_NUM];
struct rte_flow_action action[MAX_ACTION_NUM];
struct rte_flow_item_ecpri ecpri_spec;
struct rte_flow_item_ecpri ecpri_mask;

/* Ethernet */
patterns[0].type = RTE_FLOW_ITEM_TYPE_ETH;
patterns[0].spec = 0;
patterns[0].mask = 0;

/* ECPRI */
ecpri_spec.hdr.common.type = RTE_ECPRI_MSG_TYPE_IQ_DATA;
ecpri_spec.hdr.type0.pc_id = pc_id_be;

ecpri_mask.hdr.common.type  = 0xff;
ecpri_mask.hdr.type0.pc_id     = 0xffff;
ecpri_spec.hdr.common.u32    = rte_cpu_to_be_32(ecpri_spec.hdr.common.u32);
pattern[1].type = RTE_FLOW_ITEM_TYPE_ECPRI;
pattern[1].spec = &ecpri_spec;
pattern[1].mask = &ecpri_mask;

/* END the pattern array */
patterns[2].type = RTE_FLOW_ITEM_TYPE_END
```

Action

Table 17. QUEUE action for given queue id

Index	Action	Fields	Description	Value
0	QUEUE	index	queue indices to use	Must be 0,1,2,3, etc
1	END			

The following code sets up the action *RTE_FLOW_ACTION_TYPE_QUEUE* and calls the *rte_flow_create* function to create the RTE Flow rule.

```
#define MAX_ACTION_NUM                  2
uint16_t rx_q = 4;
struct rte_flow_action_queue queue = { .index = rx_q };

struct rte_flow *handle;
struct rte_flow_error err;
struct rte_flow_action actions[MAX_ACTION_NUM];
struct rte_flow_attr attributes = {.ingress = 1 };

action[0].type = RTE_FLOW_ACTION_TYPE_QUEUE;
action[0].conf = &queue;
action[1].type = RTE_FLOW_ACTION_TYPE_END;

handle = rte_flow_create (port_id, &attributes, patterns, actions, &err);
```