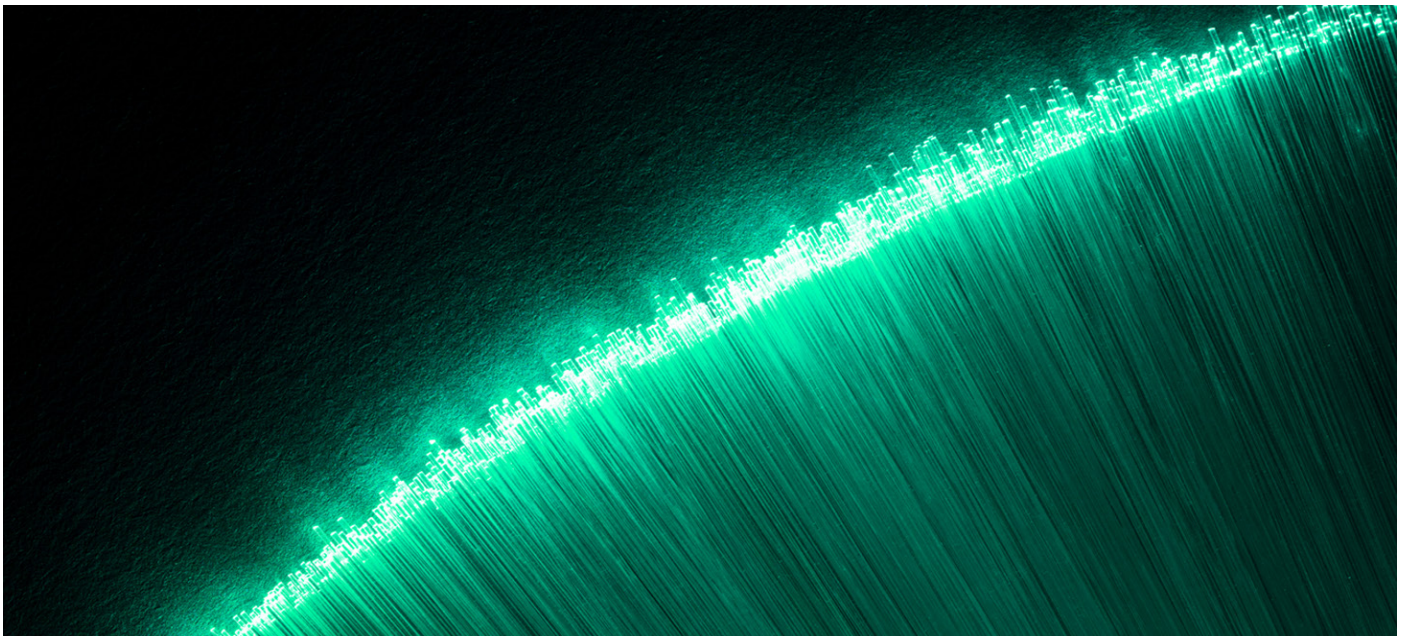
Hewlett Packard Enterprise

Technical white paper

HPE PROLIANT DL110 GEN10 PLUS SERVER READINESS FOR OPEN RAN USE CASE

CONTENTS



INTRODUCTION

The radio access network (RAN) connects cellular devices to a mobile network operator's core network. It is deployed as a large, geographically distributed assembly of nodes called base stations. Base stations include a base band unit (BBU) and a radio unit (RU). BBU processes baseband signals in the digital domain while RU generally implements digital/analog conversions as well as processes RF analog signals coming to/from an antenna.

The first fundamental step of bringing more openness and flexibility into RAN architecture is disaggregating the monolithic baseband processing of a base station into separate distinct layers, communicating via well-defined interfaces. That disaggregation can be implemented in a number of ways, with 3GPP TR 38.801 defining up to eight potential splits between centralized and distributed portions of baseband.

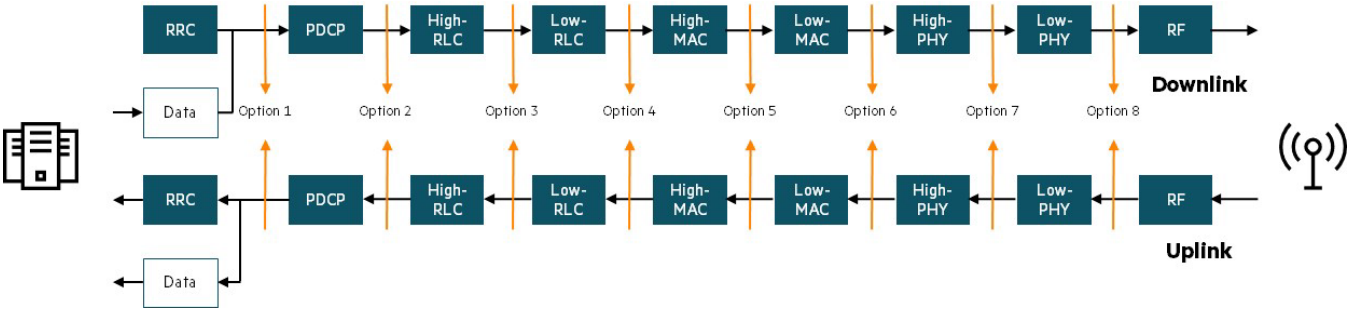


FIGURE 1. 3GPP TR 38.801 split options for baseband

Recent advancements in general-purpose compute as well as telco-driven industry initiatives for open, standard interfaces within the RAN domain have led to open virtualized RAN (vRAN). It is based on the following technological principles:

- Fully decoupled software running on abstracted general-purpose hardware, with a best-of-breed approach
- Functional components implemented as abstracted software interacting via open standardized interfaces
- General-purpose edge compute infrastructure supporting multiple use cases and workloads, with vRAN workload as one of the tenants
- Cloud-native capabilities in deployment, lifecycle management, scaling, and redundancy of vRAN workload and underlying infrastructure

O-RAN Alliance defines centralized unit (CU) as a logical node hosting RRC, SDAP, and PDCP protocols and distributed unit (DU) as a logical node hosting RLC/MAC/high-PHY functions based on a lower layer functional split. A radio unit (RU) is defined as a logical node hosting low-PHY functions and RF processing.

In scenarios where DU is virtualized, layer 1 forward error correction (FEC) can be offloaded to an accelerator card. In addition, SR-IOV and DPDK enabled NIC cards are used to achieve accelerated packet processing needed to support high bandwidth fronthaul networks.

FEC resolves data transmission errors over unreliable or noisy communication channels. FEC technology detects and corrects a limited number of errors without the need for retransmission. For 4G LTE networks, FEC is employed during turbo channel coding while for 5G networks it is used during low-density parity-check (LDPC) channel coding. Both channel coding mechanisms add parity bits to the message. These redundant bits enable the receiving station to detect and fix bits that are corrupted during transmission. However, FEC is very compute intensive.

FEC is a common function that is not differentiated across vendor implementations and hence can be easily offloaded. Accelerating FEC provides benefits by freeing up CPU resources while improving encoding/decoding throughput and latency.

This white paper summarizes the results of validation tests performed by Hewlett Packard Enterprise on Intel® vRAN Accelerator ACC100 Adapter in HPE ProLiant DL110 Gen10 Plus—Telco server.

TEST SETUP

Hardware setup

For this test, HPE ProLiant DL110 Gen10 Plus server was configured with Intel Ethernet Network Adapter E810-XXVDA4T and Intel vRAN Accelerator ACC100 Adapter.

The E810-XXVDA4T supports IEEE 1588 Precision Time Protocol (PTP) v2, Synchronous Ethernet (SyncE), and integrated GNSS, for high-precision timing synchronization. This card also supports SR-IOV and DPDK.

The Intel vRAN Accelerator ACC100 Adapter offloads FEC processing. It is an eASIC device that requires low power compared to FPGA or ASIC device and supports PCI Gen 3 x 16 interface.

Hardware BOM for this setup is as follows:

TABLE 1. Hardware bill of materials

Quantity	Part no.	Product description
1	P39478-B21	HPE ProLiant DL110 Gen10 Plus Front Cabled Telco Configure-to-order Server
1	P37603-B21	Intel® Xeon® Gold 6338N 2.2GHz 32-core 185W Processor for HPE
8	P06031-B21	HPE 16GB (1x16GB) Dual Rank x8 DDR4-3200 CAS-22-22-22 Registered Smart Memory Kit
2	P19890-B21	HPE 480GB SATA 6G Read Intensive M.2 2280 5300P SSD
1	P41636-B21	Intel Ethernet Network Adapter E810-XXVDA4T, 0/25Gb 4-port SFP28 Adapter for HPE
1	NA	Intel vRAN Accelerator ACC100 Adapter
1	P43150-B21	HPE ProLiant DL110 Gen10 Plus 700W Flex Slot -48VDC Hot Plug Low Halogen Power Supply Kit

Cloud platform setup

After power on, Virtualization—Max Performance and additional UEFI settings were configured, and cloud platform was deployed on HPE ProLiant DL110 in a single-node cluster mode.

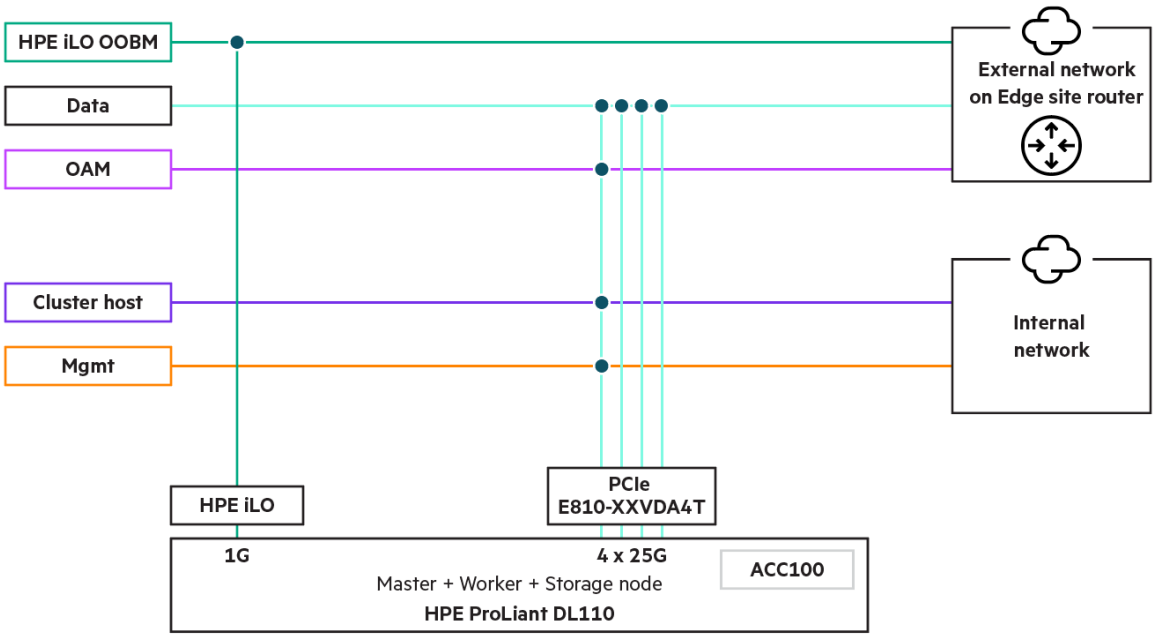


FIGURE 2. Cloud platform deployment diagram with network connections

ACCELERATED PACKET PROCESSING

In order to get enhanced packet processing, Huge pages and Isolated CPU cores were configured. SR-IOV & DPDK enabled Intel 810 NIC is used to achieve higher throughput from NIC interfaces for communication between Kubernetes containers running on different hosts or between containers and external networks.

SR-IOV device plug-in discovers and advertises SR-IOV network virtual functions (VFs) in a Kubernetes host. A range of VLANs referred to as Data network is configured before provisioning the SR-IOV interfaces. An SR-IOV interface can support multiple VFs for VFIO-based containers.

The following test procedure was used:

1. Deploy Isolated CPU policy and configure 1G Huge pages.
2. Provision SR-IOV interface with virtual functions.
3. Instantiate Pod and verify SR-IOV VF assignment to the container application.
4. CPU manager policy is configured as “static”, and Topology manager policy as “restricted”.
5. The VFs associated with the SR-IOV Pods will be visible in the node, with VLAN tags associated.

DPDK’s Ethernet Device framework is used for packet processing acceleration of Ethernet network cards supporting SR-IOV and DPDK.

FEC OFFLOAD TEST IMPLEMENTATION

Intel’s FlexRAN™ reference software contains libraries for LTE and 5G NR Layer 1 workload acceleration. The FlexRAN SDK provides a set of building blocks for RAN developers to build 5G NR VNFs such as vDU. FlexRAN artifacts include APIs, source code, and build and test environment parameters.

DPDK’s Ethernet device framework is used for packet processing acceleration of Ethernet network cards supporting SR-IOV and DPDK. Similarly, DPDK wireless baseband device library, DPDK BBDEV, provides a common programming framework that abstracts hardware accelerators. ACC100 supports the DPDK BBDEV API for FEC offload. Physical devices are discovered during DPDK initialization and virtual functions are created and assigned to applications.

A Pod was instantiated in cloud-native platform to run FEC tests.

The following test procedure was used to run FEC tests:

1. Configure virtual functions and apply telco-grade settings
2. Instantiate Pod for container workload
3. Download DPDK, FlexRAN artifacts, and DPDK patch
4. Install System Studio for Intel C++ compiler (ICC)
5. Compile FlexRAN SDK
6. Apply DPDK patch and compile DPDK in the Pod
7. Run FEC tests

test-bbdev was used to run FEC offload tests. It is a DPDK utility for measuring parameters. Tests were run to measure FEC throughput, FEC latency, and FEC offload costs.

test-bbdev takes a number of parameters that are described as follows:

- -n NUM_OPS

  Specifies the number of operations to process on the device

- -b BURST_SIZE

  Specifies operations enqueue/dequeue burst size

- -l NUM_LCORES

  Specifies the number of lcores to run

- -c TEST_CASE

  Defines test case to run

- -v TEST_VECTOR

  Specifies the path to test vector file

Run performance tests

```
# Environment path setup for test-bbdev testing tool and test vectors
cd /opt/dpdk/app/test-bbdev/
PCI_ADDR="-a <PCI ID of FEC VF>"
COREMASK="0x<COREMASK>"
```

bbdev LDCP decode/encode throughput tests

```
python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 8 -c throughput -v
./test_vectors/ldpc_dec_v6563.data

python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 8 -c throughput -v
./test_vectors/ldpc_enc_v2570_lbrm.data
```

- These commands perform full operation of enqueue and dequeue.
- These commands run in poll mode.
- These commands measure the achieved throughput on a subset of CPU cores.
- Results are printed in million operations per second and million bits per second.

TABLE 2. Throughput tests

Test vector file	Output	Operations
ldpc_dec_v6563.data	Total throughput for 8 cores: 1196350.9 MOPS, 10011.064 Mbps @ max. 6 iterations	dev: 0000:4c:00.1, nb_queues: 24, burst size: 80, num ops: 100, num_lcores: 8, op type: RTE_BBDEV_OP_LDPC_DEC, itr mode: PMD mode, GHz: 2.2
ldpc_enc_v2570_lbrm.data	Total throughput for 8 cores: 4936799.9 MOPS, 41706.086 Mbps	dev: 0000:4c:00.1, nb_queues: 24, burst size: 80, num ops: 100, num_lcores: 8, op type: RTE_BBDEV_OP_LDPC_ENC, itr mode: PMD mode, GHz: 2.2

bbdev LDCP decode/encode latency tests

```
python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 1 -c latency -v
./test_vectors/ldpc_dec_v7813.data

python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 1 -c latency -v
./test_vectors/ldpc_enc_v7813.data
```

- These commands measure the time consumed from the first enqueue until the first appearance of a dequeued result.
- This measurement represents the full latency of a bbdev operation (encode or decode) to start.

TABLE 3. Latency tests

Test vector file	Output	Operations
ldpc_dec_v7813.data	Operation latency: Avg.: 31275 cycles, 14.2159 us Min.: 16586 cycles, 7.53909 us Max.: 45964 cycles, 20.8927 us	dev:0000:4c:00.1, burst size: 80, num ops: 100, op type: RTE_BBDEV_OP_LDPC_DEC
ldpc_enc_v7813.data	Operation latency: Avg.: 21281 cycles, 9.67318 us Min.: 10136 cycles, 4.60727 us Max.: 32426 cycles, 14.7391 us	dev:0000:4c:00.1, burst size: 80, num ops: 100, op type: RTE_BBDEV_OP_LDPC_ENC

bbdev LDPC decode/encode offload cost tests

```
python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 1 -c offload -v
./test_vectors/ldpc_dec_HARQ_26449_1.loopback_r

python3 ./test-bbdev.py -e="-c ${COREMASK} ${PCI_ADDR}" -n 100 -b 80 -l 1 -c offload -v
./test_vectors/ldpc_enc_v11835.data
```

- These commands measure the CPU cycles consumed from the receipt of a user enqueue until it is put on the device queue.
- The test measures:
  - Enqueue offload cost: Software only enqueue offload cost—the cycle count and time (us) from the point the enqueue API is called until the point the operation is put on the accelerator queue.
  - Enqueue accelerator offload cost: The cycle count and time (us) from the point the operation is put on the accelerator queue until the return from enqueue.
  - Dequeue offload cost: Software dequeue cost, the cycle count and time (us) consumed to dequeue one operation.
  - Empty queue dequeue offload cost: The cycle count and time (us) consumed to dequeue from an empty queue.

TABLE 4. Offload cost tests

Test vector file	Output	Operations
ldpc_dec_HARQ_26449_1.loopback_r	Enqueue driver offload cost latency: Avg.: 5843 cycles, 2.65591 us Min.: 2458 cycles, 1.11727 us Max.: 9228 cycles, 4.19455 us Enqueue accelerator offload cost latency: Avg.: 318 cycles, 0.144545 us Min.: 318 cycles, 0.144545 us Max.: 318 cycles, 0.144545 us Dequeue offload cost latency—one op: Avg.: 1418 cycles, 0.644545 us Min.: 542 cycles, 0.246364 us Max.: 2294 cycles, 1.04273 us Empty dequeue offload: Avg.: 219 cycles, 0.0995455 us Min.: 94 cycles, 0.0427273 us Max.: 344 cycles, 0.156364 us	dev: 0000:4c:00.1, burst size: 80, num ops: 100, op type: RTE_BBDEV_OP_LDPC_DEC
ldpc_enc_v11835.data	Enqueue driver offload cost latency: Avg.: 11591 cycles, 5.26864 us Min.: 1176 cycles, 0.534545 us Max.: 22006 cycles, 10.0027 us Enqueue accelerator offload cost latency: Avg.: 357 cycles, 0.162273 us Min.: 308 cycles, 0.14 us Max.: 406 cycles, 0.184545 us Dequeue offload cost latency—one op: Avg.: 2306 cycles, 1.04818 us Min.: 270 cycles, 0.122727 us Max.: 4342 cycles, 1.97364 us Empty dequeue offload: Avg.: 209 cycles, 0.095 us Min.: 84 cycles, 0.0381818 us Max.: 334 cycles, 0.151818 us	dev: 0000:4c:00.1, burst size: 80, num ops: 100, op type: RTE_BBDEV_OP_LDPC_ENC

CONCLUSION

Test results in this white paper clearly show the readiness of HPE ProLiant DL110 Gen10 Plus server for virtualized DU applications where FEC is offloaded to the Intel vRAN Accelerator ACC100 Adapter. Indeed, a number of vRAN software vendors are already using Intel ACC100 on HPE infrastructure to meet high-performance requirements for challenging Open RAN use cases.

Open RAN provides mobile operators with a broad ecosystem of vendors offering interoperable network products, as well as opportunities to reduce TCO, accelerate time to market, and introduce new innovations in the market. Open RAN solutions on general-purpose compute can be implemented without the use of hardware accelerators when the compute processing requirements are low. For larger compute intensive deployments, using FEC HW accelerators such as ACC100 help to both increase performance capacity and lower overall power usage. Intel also provides power savings through power management techniques using C1/C6 power management and CPU core pooling schedulers that can remove cores from the pool during low CPU utilization periods.

ABBREVIATIONS AND REFERENCE

Abbreviations

Abbreviations	Terminologies
LDPC	Low-density parity-check
SR-IOV	Single root I/O virtualization
DPDK	Data Plane Development Kit
VF	Virtual function

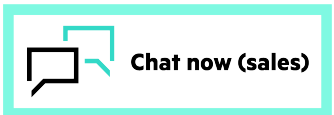
Reference

vRAN 2.0 on HPE infrastructure

LEARN MORE AT

hpe.com/telco/cloud

Make the right purchase decision.
Contact our presales specialists.



© Copyright 2022 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Intel and Intel Xeon Gold are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. All third-party marks are property of their respective owners.

a50005156ENW, Rev. 1

Hewlett Packard Enterprise