White Paper

Communications Service Providers

Intel-HPE Verified Reference Configuration for Virtualized Radio Access Networks on the HPE ProLiant DL110

Table of Contents

1 Objectives of Intel-HPE VRC for vRAN on HPE ProLiant DL110

Virtualization of telco networking functions on general compute platforms is a well proven architectural trend enabling maximum flexibility, sustainable technological evolution, and superior TCO in the telco industry. By migrating telco network infrastructure to distributed cloud platforms, the telco industry is leveraging the economies of scale that has already been seen in public cloud infrastructure.

While this trend has been mainly limited to telco network functions in core networks, recently cellular Radio Access Network (RAN) began to open up for virtualized architectures, enabled by OpenRAN architectural evolution as well as ever growing efficiency of general compute platforms for real time signal processing.

HPE and Intel are collaborating to transform the economics of RAN by introducing workload optimized general compute platforms meeting the performance and efficiency needed by modern, highly loaded 5G NR RAN. Delivering on this goal, HPE launched the HPE ProLiant DL110 general compute platform built on latest 3rd Gen Intel® Xeon® Scalable processors and optimized for the needs of production vRAN.

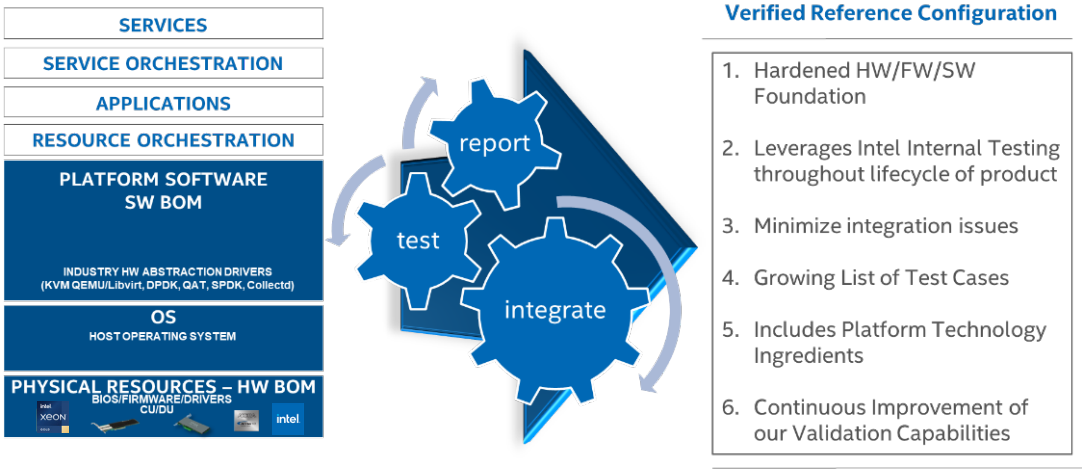
OpenRAN supports virtualized RAN with Hardware and Software disaggregation, allowing telcos to select multiple vendors in the solution stack that must interoperate together in a production vRAN deployment. HPE and Intel recognized the benefit of having pre-validated hardware, firmware and software components built on a general purpose compute platform when bundling ingredients in vRAN solutions.

In order to speed up the production deployment of vRAN, HPE and Intel collaborated within the **Verified Reference Configuration for vRAN** program to create a verified hardware, firmware and software recipe, pre-verified with vRAN real time processing workloads to ensure performance and efficiency. This VRC can serve as a known working server platform and associated firmware/SW dependencies up to the FlexRAN™ L1 software stack, that would be a foundation to be combined with an ISV L2 and L3 SW stack for a complete vRAN deployable solution.

The objective of this document is to describe the scope of **Verified Reference Configuration for vRAN** program and the resulting highly-optimized and pre-verified hardware, firmware and software of compute solution underpinning vRAN workloads, all based on HPE ProLiant DL110, which includes the 3rd generation Intel® Xeon® processor.

2 Scope of Intel-HPE VRC for vRAN on the HPE ProLiant DL110

Intel is committed to continuous improvement of its verification capabilities and introduces new customer use cases to produce the hardened hardware, firmware, and software foundation that is verified as a reference solution. Customers can start to build their production workload for large scale deployment on this known working foundation. As a result, this reduces early integration issues, simplifies evaluations, and accelerates time to market.



The test methodologies involved Intel internal testing to minimize integration issues exposure to customers. Intel verification reference configuration for vRAN covers the following verification:

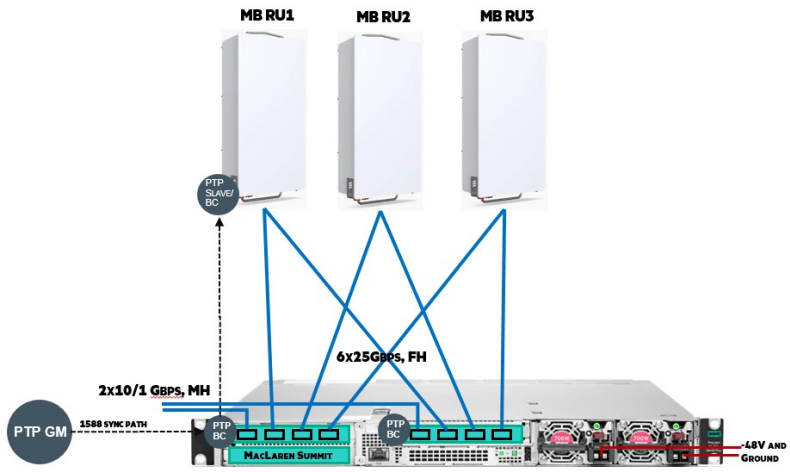
- FlexRAN™ workload virtualized
- Networking
- DPDK
- Virtualized – Single Root IO Virtualization (SRIOV) and VirtIO/vhost
- Platform level testing
- Service Assurance

An Intel Verified Configuration is carefully and thoroughly verified by Intel to help ensure the platform configuration in terms of hardware components, BIOS recommendation, and the software solution stack is proven, stable, and reliable for the vRAN real time workload.

The next section defines the solution components from hardware ingredients, platform, BIOS recommendation, and operating systems tuning for the Intel verified reference configuration for RAN on vRAN workload-optimized HPE ProLiant DL110 general purpose compute platform.

3 RAN scenarios targeted by VRC for vRAN on the HPE ProLiant DL110

The HPE ProLiant DL110 platform was specifically optimized to run vRAN real time processing workloads at scale, across Distributed and Centralized (DRAN, CRAN) deployment scenarios running both Narrow Band and Midband 5G NR configurations in production at scale.



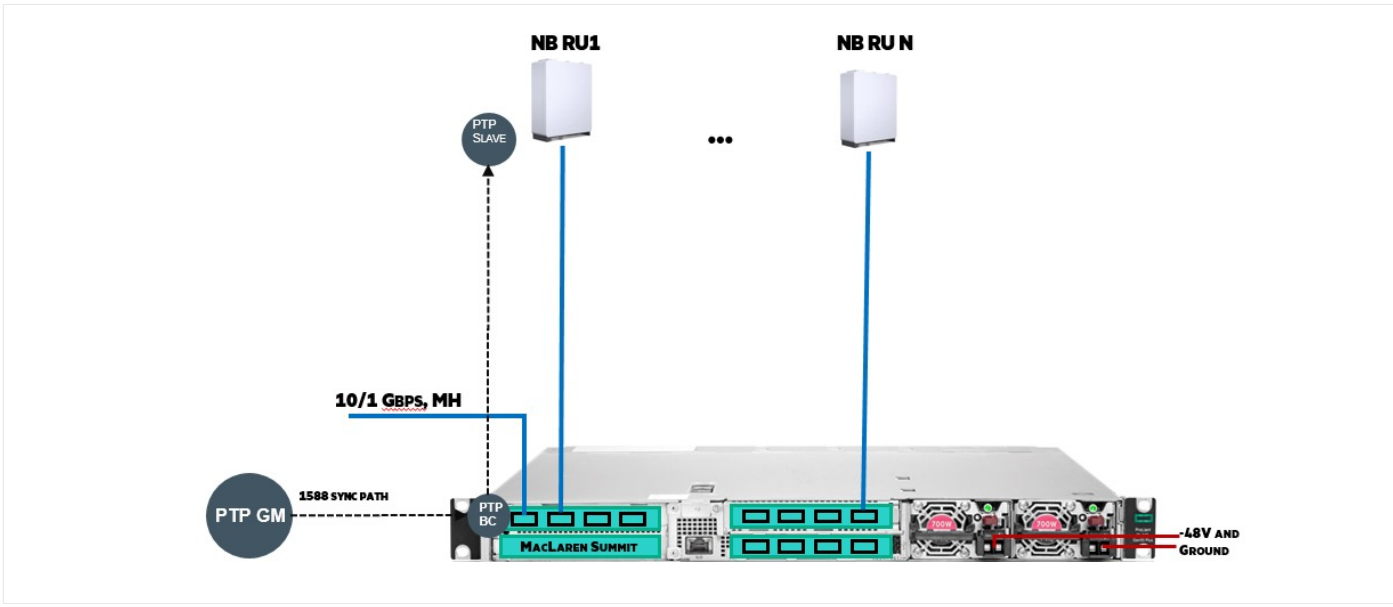


Figure 2. Narrow Band deployment example enabled by Verified Reference Configuration

Intel-HPE VRC for vRAN targets both Narrow Band (up to 12x20MHz cell carriers MIMO 4x4) and Midband mMIMO (up to 3x100MHz 64T64R mMIMO) 5G NR deployments, as illustrated on Figure 1 and 2.

The above diagrams are examples of vRAN architectures enabled by Intel-HPE VRC for vRAN on the HPE ProLiant DL110 – the same Verified Reference Architecture on the HPE ProLiant DL110 can underpin many more flavors of RAN sites in terms of RF configurations, fronthaul structure, synchronization, and midhaul architecture.

4 VRC details for vRAN on HPE ProLiant DL110

The objective of this section is to describe Verified Reference Configuration for vRAN on the HPE ProLiant DL110, including validated hardware Bill of Materials and firmware components.

4.1 Hardware, firmware of VRC for vRAN on HPE ProLiant DL110

The tables below describe the pre-verified hardware and firmware components resulting from Intel-HPE Verified Reference Configuration program for vRAN on the HPE ProLiant DL110.

The hardware Bill of Materials for Verified Reference Configuration for vRAN on the HPE ProLiant DL110 is described in Table 1.

P39478-B21	HPE ProLiant DL110 Gen10+ Front Telco Cbl CTO Svr	1	
BD505A	HPE iLO Adv 1-svr Lic 3yr Support	1	
P43150-B21	HPE ProLiant DL110 Gen10+ 700W FS -48VDC PS Kit	1	*
P37603-B21	Intel® Xeon® Gold 6338N Processor 32 cores, 2.2 Ghz, 185W	1	
P06031-B21	HPE 16GB 2Rx8 PC4-3200AA-R Smart Kit	8	
P40513-B21	HPE 480GB NVMe RI M.2 22110 MV SSD	2	
R8G90C	Intel vRAN Dedicated Accelerator ACC100, HHHL x16 PCIe 3.0 40W LDPC/Turbo	1	
P08458-B21	Intel Ethernet Network Adapter E810-XXVDA4 10/25GbE 4p SFP28 (Salem Channel)	2	**

* - number of PSUs and AC vs DC distribution can be changed to fit concrete deployment scenario
** - number or flavor of Intel E810 NIC cards can be adjusted to fit concrete fronthaul and synchronization architecture

HPE ProLiant DL110 firmware and software solution stacks for Verified Reference Configuration details can be provided upon request through your field representative.

4.2 Configuration of UEFI and OS within VRC for vRAN on HPE ProLiant DL110

Proper configuration of UEFI, kernel and operating system is important to achieve deterministic low latency system behavior essential for real time baseband processing workloads of vRAN, all while keeping power consumption at bay.

As a guiding strategy in optimizing UEFI profile for required behavior under real time baseband processing vRAN workload, Intel-HPE VRC program leveraged general recommendations for BIOS setting for FlexRAN workload for 3rd generation Intel® Xeon® Scalable processors. This BIOS tuning strategy has been translated to proper UEFI configuration profile of HPE ProLiant DL110. (Contact us for further details.)

5 VRC for vRAN on the HPE ProLiant DL110, Testing Setup

Before testing the actual vRAN workload, a foundational set of tests has been executed within VRC for vRAN on HPE ProLiant DL110, establishing solid baseline.

5.1 Foundational tests

The following foundational tests are recommended and have been executed within VRC for vRAN on HPE ProLiant DL110:

- **MLC**

The first application is the Memory Latency Checker which can be downloaded from https://software.intel.com/en-us/articles/intelr-memory-latency-checker.

Download the latest version and execute this application, unzip the tarball package and go into Linux folder and execute

./mlc

- **Jitter**

The jitter application aims to measure the variability of latency in the execution of a user space dummy loop with dummy operation, more information about this tool is available at https://wiki.fd.io/view/Pma_tools/jitter

Use the following steps to download the tool, build, and execute the tool targeting an idle core:

- git clone https://gerrit.fd.io/r/pma_tools
- cd pma_tools/jitter
- make
- ./jitter –c 2 –i 200

Review Inst_Jitter column, using Deterministic Performance in a BIOS setting, the jitter should not exceed 10k.

- **Cyclictest**

vRAN DU/CU solution requires vRAN system latency and responsiveness to be very low (e.g. below 35us over prolonged period of time). Cyclictest application is used to measure this latency.

Following setup of cyclictest is recommended:

-VM Kernel Command Line:

```
crashkernel=auto rhgb quiet LANG=en_GB.UTF-8 clock=pit console=tty0 console=ttyS0 biosdevname=0 net.ifnames=0 no_timer_
check clocksource=tsc tsc=perfect intel_pstate=disable selinux=0 enforcing=0 nmi_watchdog=0 softlockup_panic=0 isolcpus=1-9
nohz_full=1-9 default_hugepagesz=1G hugepagesz=1G hugepages=15 rcu_nocbs=1-9 kthread_cpus=0 irqaffinity=0 rcu_nocb_poll
```

-Configuration in VM:

```
#To set ktimer a very high priority#
for i in $(seq 0 9); do mypid=`pgrep -o ksoftirqd\/$i`;chrt -f -p 2 $mypid; done
for i in $(seq 0 9); do mypid=`pgrep -o ktimersoftd\/$i`;chrt -f -p 3 $mypid; done
for i in $(seq 0 9); do mypid=`pgrep -o rcuc\/$i`;chrt -f -p 4 $mypid; done


#To disable ksm(A memory sharing mechanism), will not consume CPU resources on scanning and merging same pages.#
echo 0 > /sys/kernel/mm/ksm/merge_across_nodes
echo 2 > /sys/kernel/mm/ksm/run
```

```
#To make RT process occupy 100% runtime in the period#
echo -1 > /proc/sys/kernel/sched_rt_period_us
echo -1 > /proc/sys/kernel/sched_rt_runtime_us

#To disable timer_migration, will not migrate timers away from idle CPUs to busy CPUs, can reduce CPU consumption#
echo 0 > /proc/sys/kernel/timer_migration

- Start cyclictest for 24 hours:
   taskset -c 9 ./cyclictest -S -p99 -n -m -d0 -A 7e
```

After 24hrs test, the max latency should be under 35 as shown below:

```
[root@wolfpass tools]# taskset -c 9 ./cyclictest -S -p99 -n -m -d0 -A 7e
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.00 0.01 0.05 1/289 1368

T: 0 (13326) P:99 I:1000 C:84963372 Min:      8 Act:   12 Avg:   10 Max:      22
T: 1 (13327) P:99 I:1000 C:84963373 Min:      8 Act:   11 Avg:   10 Max:      25
T: 2 (13328) P:99 I:1000 C:84963368 Min:      8 Act:   11 Avg:   10 Max:      29
T: 3 (13329) P:99 I:1000 C:84963365 Min:      9 Act:   11 Avg:   11 Max:      28
T: 4 (13330) P:99 I:1000 C:84963364 Min:      8 Act:   11 Avg:   10 Max:      25
T: 5 (13331) P:99 I:1000 C:84963364 Min:      8 Act:   10 Avg:   10 Max:      29
```

The above set of foundational tests ensures low latency deterministic behavior of the HPE ProLiant DL110 system and provides a foundation upon further verification of vRAN workload rests.

5.2 PHY Acceleration tests

Production grade high capacity RAN solution benefits from offloading FEC of higher PHY baseband processing to dedicated silicon, packaged as PCIe card.

The recommended architectural approach within Intel-HPE VRC for vRAN on the HPE ProLiant DL110 is to offload 5G NR LDPC and/or 4G LTE Turbo coding/decoding functionality into the Intel vRAN Dedicated Accelerator ACC100 Adapter. The accelerator is implemented using Intel eASIC technology, which provides lower unit cost and power consumption than Application-Specific Integrated Circuits (ASICs), while also offering faster time-to-market and smaller footprint than prior solutions.

From software standpoint, acceleration leverages DPDK BBDEV extension as an open API between vDU application and Intel ACC100 Adapter.

Verifying the functional capabilities and the performance of Intel ACC100 adapter together with DPDK BBDEV API is an important step in the Verified Reference Configuration for vRAN on the HPE ProLiant DL110:

- **BBDEV-test**

The dpdk-test-bbdev tool is a Data Plane Development Kit (DPDK) utility that allows measuring functional capability parameters of PMDs available in the BBDEV framework. Tests for execution are: latency, throughput, validation, BLER (block error rate), and sanity tests. Execution of tests can be customized using various parameters passed to a python running script.

There are 6 main test cases to be executed using bbdev-test tool:

1. Validation tests [-c validation]

- Performs full operation of enqueue and dequeue
- Compares the dequeued data buffer with expected values in the test vector (TV) being used
- Fails if any dequeued value does not match the data in the TV

2. Offload Cost measurement [-c offload]

- Measures the CPU cycles consumed from the receipt of a user enqueue until it is put on the device queue

3. Latency measurement [-c latency]

- Measures 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 execute

4. Poll-mode Throughput measurement [-c throughput]

- Performs full operation of enqueue and dequeue
- Executes in poll mode
- Measures the achieved throughput on a subset or all available CPU cores
- Dequeued data is not validated against expected values stored in TV
- Results are printed in million operations per second and million bits per second

5. BLER measurement [-c bler]

- Performs full operation of enqueue and dequeue
- Measures the achieved throughput on a subset or all available CPU cores
- Computed BLER (Block Error Rate, ratio of blocks not decoded at a given Signal to Noise Ratio (SNR)) in % based on the total number of operations.

6. Interrupt-mode Throughput [-c interrupt]

- Similar to Throughput test case, but using interrupts. No polling.

Scripts:

```
python3 ./test-bbdev.py -e="-c 0xff0 -w${PCI_ADDR}" -n 60 -b 24 -l 4 -c validation -v ./test_vectors/* >> test_report
python3 ./test-bbdev.py -e="-c 0xff0 -w${PCI_ADDR}" -n 80 -b 64 -l 1 -c latency -v ./test_vectors/* >> test_report
python3 ./test-bbdev.py -e="-c 0xff0 -w${PCI_ADDR}" -n 100 -b 80 -l 1 -c throughput -v ./test_vectors/* >> test_report
python3 ./test-bbdev.py -e="-c 0xff0 -w${PCI_ADDR}" -n 50 -b 32 -l 1 -c offload -v ./test_vectors/* >> test_report
python3 ./test-bbdev.py -e="-c 0xff0 -w${PCI_ADDR}" -n 70 -b 50 -l 1 -c bler -v ./test_vectors/l1* >> test_report
python3 ./test-bbdev.py -e="-c 0xff -w${PCI_ADDR}" -n 10 -b 10 -l 1 -c interrupt -v ./test_vectors/l1* >> test_report
```

Ensure that there is the result from the print log does not show any failing results in each test. Some tests are skipped because they are not supported by the device which does not impact vRAN functionality or performance.

- **DPDK-test-bbdev app**

DPDK-test-bbdev-app is a test tool to enable execution of only the uplink (UL) and downlink (DL) Low Density Parity check Code (LDPC) operations in a loop for any given cell configuration on the LDPC hardware (without the rest of the L1 pipeline). This can be used notably for HW performance and power measurements. It is included in the FlexRAN™ software release.

5.3 vRAN workload tests with Intel FlexRAN™ reference architecture

Intel's FlexRAN reference architecture is a flexible and scalable open RAN reference implementation on Intel Architecture based solution. FlexRAN reference architecture refers to both the reference instance as well as the baseband software and supports both bare metal as well as virtualized to cloud RAN topologies. FlexRAN software optimizes the use of Intel® Xeon® Scalable processors and add-in hardware acceleration to fully realize a high-performant vRAN base station on a general-purpose compute platform.

The FlexRAN reference architecture has been widely adopted by a diverse ecosystem of RAN players and has been used within the Intel-HPE VRC for vRAN on the HPE ProLiant DL110 as the vRAN workload representation for functional and performance tests.

Within the testing setup, verification of consistent performance for representative (Instead of concrete) RAN configurations (e.g., 3x100MHz mMIMO 64x64) were executed on HPE ProLiant DL110 using Testmac, a standalone testing tool without any external dependency on third party software or hardware components.

Testmac is a standalone testing without any external dependency to third party software or hardware components. The L1 application is run in real-time mode and source of time (for TTI / symbol boundaries) is the base on Intel® Xeon® processor's internal time stamp counter.

The samples are read from reference files (from test config) and loaded into system memory (DDR) and L1 application reads from and writes to memory. The cycle count is used as a form of performance, where time stamps are logged through every event in the pipeline and stored.

In this mode:

- There are no external dependencies in terms of hardware or third-party software components.
- The L1 application is run in real-time mode and source of time (for TTI / Symbol boundaries) is the Intel® Xeon Scalable processor's internal time stamp counter.
- The IQ samples are read from Reference Files (from test config) and loaded into DDR and L1 application reads from / writes to memory.
- This mode is used for Cycle count Performance Benchmarking where time stamps are logged through every event in the pipeline and stored.

6 VRC for vRAN on HPE ProLiant DL110, Testing Results Overview

Some of the key foundational testing results and functional tests for the pre-integrated platform are provided below:

- **Cyclictest executed in Guest VM**

  Maximum observed latency not exceeding 35usec throughout 24hours period – below target threshold of 35usec

```
policy: fifo: loadavg: 0.13 0.16 0.17 1/789 581095

T: 0 (22400) P:99 I:1000 C:84341356 Min:      5 Act:   14 Avg:   14 Max:      35
T: 1 (22401) P:99 I:1000 C:84341356 Min:      5 Act:   16 Avg:   14 Max:      31
T: 2 (22402) P:99 I:1000 C:84341356 Min:      5 Act:   14 Avg:   14 Max:      32
T: 3 (22403) P:99 I:1000 C:84341352 Min:      5 Act:   14 Avg:   14 Max:      30
T: 4 (22404) P:99 I:1000 C:84341352 Min:      5 Act:   13 Avg:   14 Max:      30
T: 5 (22405) P:99 I:1000 C:84341352 Min:      5 Act:   16 Avg:   14 Max:      30
```

- **BBDEV test**

  100% pass rate for the key 6 metrics of acceleration

```
python3 ./test-bbdev.py -e="-c 0xff0 -w 11:00.0" -n 60 -b 24 -l 4 -c validation -v ./test_vectors/* >> test_report1
[root@localhost test-bbdev]# cat test_report1 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:64
[root@localhost test-bbdev]# cat test_report1 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
```

Result:
Validation: Pass 64, Fail 0

```
python3 ./test-bbdev.py -e="-c 0xff0 -w 11:00.0" -n 80 -b 64 -l 1 -c latency -v ./test_vectors/* >> test_report2
[root@localhost test-bbdev]# cat test_report2 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
[root@localhost test-bbdev]# cat test_report2 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:32
```

Latency: Pass 32 Fail 0

```
python3 ./test-bbdev.py -e="-c 0xff0 -w 11:00.0" -n 100 -b 80 -l 1 -c throughput -v ./test_vectors/* >> test_report3
[root@localhost test-bbdev]# cat test_report3 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:32
[root@localhost test-bbdev]# cat test_report3 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
```

Throughput: Pass 32 Fail 0

```
python3 ./test-bbdev.py -e="-c 0xff0 -w 11:00.0" -n 50 -b 32 -l 1 -c offload -v ./test_vectors/* >> test_report4
[root@localhost test-bbdev]# cat test_report4 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:64
[root@localhost test-bbdev]# cat test_report4 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
```

Offload: Pass 64 Fail 0

```
python3 ./test-bbdev.py -e="-c 0xff0 -w 11:00.0" -n 70 -b 50 -l 1 -c bler -v ./test_vectors/l* >> test_report5
[root@localhost test-bbdev]# cat test_report5 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:16
[root@localhost test-bbdev]# cat test_report5 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
```

BLER: Pass 16 Fail 0

```
python3 ./test-bbdev.py -e="-c 0xff -w 11:00.0" -n 10 -b 10 -l 1 -c interrupt -v ./test_vectors/l* >> test_report6
[root@localhost test-bbdev]# cat test_report6 | grep -v Debug | grep Passed | awk  '{sum += $5} END {print "P:" sum}'
P:23
[root@localhost test-bbdev]# cat test_report6 | grep -v Debug | grep Failed | awk  '{sum += $5} END {print "F:" sum}'
F:0
```

Interrupt: Pass 23 Fail 0

- **BBDEV app**

  Stable results of BBDEV application, no exceptions in the output logs.

```
     DUL OFFSET (us):    1000   22.00   22.31   23.00
BBDEV Status 6624000 6624000
==== llapp Time: 1000 TTI 9000 9003 199315====
------------------------------------------------------------
     Cell        DL Tput    UL Tput
     0 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     1 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     2 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     3 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     4 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     5 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     6 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     7 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     8 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
     9 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
    10 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
    11 (Kbptti)    342,240     56,288  UEs 16  ReTx 10%
------------------------------------------------------------
     Stats Func:        Count     Min     Avg     Max
        DL PREP (us):    1000   11.00   13.50   16.00
     DL ENQUEUE (us):    1000    8.00    9.00    9.00
        UL PREP (us):    1000   10.00   10.02   13.00
     UL ENQUEUE (us):    1000   18.00   18.13   19.00
        UL POLL (us):    1000   54.00  153.71  251.00
        DL POLL (us):    1000    0.00   83.56  179.00
------------------------------------------------------------
     TOTAL TASK (us):    1000  1000000.00    0.00    0.00
------------------------------------------------------------
     UL HW LAT  (us):    1000  227.00  237.33  251.00     9.50 Iters
     DL HW LAT  (us):    1000  176.00  189.14  202.00
     DUL OFFSET (us):    1000   22.00   22.25   24.00
BBDEV Status 7452000 7452000
Test Completed after 10001 TTIs 200000 200000
```

- **FlexRAN Massive MIMO testing**

  The HPE ProLiant DL110 has been verified for MidBand mMIMO configuration, 3x100MHz 64x64

  Results can be provided upon request.

CONFIGURATION	USE CASE: MMIMO 100MHZ 64T64R
Test Config / Test Case	HPE ProLiant DL110 system w/ Intel® Xeon® Gold 6338N Processor + Intel vRAN Dedicated Accelerator ACC100/FD_3389
LDPC Offload	Intel vRAN Dedicated Accelerator ACC100
Number of Cells	3

- **Platform integration functional verification**

  As a foundational effort, Intel verified with integration functional testing of the HPE ProLiant DL110 compute platform, it's hardware and firmware components, drivers, and the FlexRAN software. Table 4 summarizes 100% pass rate of functional tests for the VRC recipe of components:

	TESTS ATTEMPTED	TESTS PASSED	TESTS FAILED	PASS RATE, %
DPDK_Columbiaville_100G	20	20	0	100%
Platform	8	8	0	100%
NIC_Columbiaville_100G	24	24	0	100%
Service Assurance	38	38	0	100%
SR-IOV	10	10	0	100%
FlexRAN software	11	11	0	100%
virtio_vhost	3	3	0	100%
BMC	8	8	0	100%
Networking	1	1	0	100%

7 Conclusion

Virtualized RAN architectures running on general purpose compute platform provide unprecedented flexibility, openness, and associated TCO improvement to the world of cellular access networks.

One of the challenges to reaping full benefits of vRAN deployments is the complexity and effort associated with integrating mutliple vendor ingredients into a high performance production deployable vRAN solution.

HPE and Intel are fully committed to simplify this challenge. Based on strong technological collaboration, HPE and Intel delivered Verified Reference Configuration for vRAN on the industry leading workload optimized HPE ProLiant DL110 platform. All the hardware, firmware and software components are pre-integrated and pre-verified into high performance efficient vRAN solution addressing the most challenging RAN configurations of today.

RAN RF architectures, however, are a moving target – continuous evolution in 3GPP standards and capabilities of radio interface, addition of new frequency bands, modulations and evolution of antenna techniques require respective evolution of underlying general compute solution underpinning vRAN workloads.

Therefore the Intel-HPE VRC collaboration is not a one-time integration activity. It is a recurring program targeting to continuously integrate and improve the vRAN solution on HPE ProLiant DL110 platform and it's roadmap evolution incorporating latest and greatest silicon technology to grow address vRAN scenarios beyond today's Narrow Band and Midband deployments, incorporating evolution of radio interface, fronthaul, and synchronization architectures.

The objective of long-term VRC collaboration is to stay ahead of the curve of RAN RF architectural evolution and continuously deliver improvements to the pre-integrated and pre-verified general-purpose compute solution for any production RAN deployment scenario.

0422/TM/H09/PDF ♻ Please Recycle 348670-001US