Xilinx Answer 71453
QDMA Performance Report

Important Note: This downloadable PDF of an Answer Record is provided to enhance its usability and readability. It is important to note that Answer Records are Web-based content that are frequently updated as new information becomes available. You are reminded to visit the Xilinx Technical Support Website and review (Xilinx Answer 71453) for the latest version of this Answer.

Overview

Xilinx QDMA (Queue Direct Memory Access) Subsystem for PCI Express® (PCIe®) is a high-performance DMA for use with the PCI Express® 3.x Integrated Block(s) which can work with AXI Memory Mapped or Streaming interfaces and uses multiple queues optimized for both high bandwidth and high packet count data transfers. (Please refer to QDMA Subsystem for PCI Express v4.0 - PG302 for additional details).

Xilinx provides two reference drivers for QDMA IP

- Linux Kernel driver (Linux Driver)
- DPDK Poll Mode driver (DPDK Driver)

This performance report provides the measurement of the DMA bandwidth of the QDMA IP using the reference Linux and DPDK drivers. This report provides the measured DMA bandwidth with different DMA configurations that can be extrapolated to a target application.

The reference design is targeted at a PCIe Gen 3 x16 design on a Xilinx QDMA 4.0 VU9P device on a VCU1525 board. The reference design can also be ported to other Xilinx cards.

***Note**: The QDMA DPDK Driver and Linux Driver are hosted at https://github.com/Xilinx/dma_ip_drivers/, under the QDMA directory. For known issues and other information on QDMA IP, see (Xilinx Answer 70927).*

Audience

The pre-requisite for understanding this document is that the user has gone through the following from (Xilinx Answer 70928):

- QDMA Subsystem for PCI Express v4.0 - PG302,
- QDMA Linux Kernel Reference Driver User Guide and
- DPDK Driver User Guide

System Overview

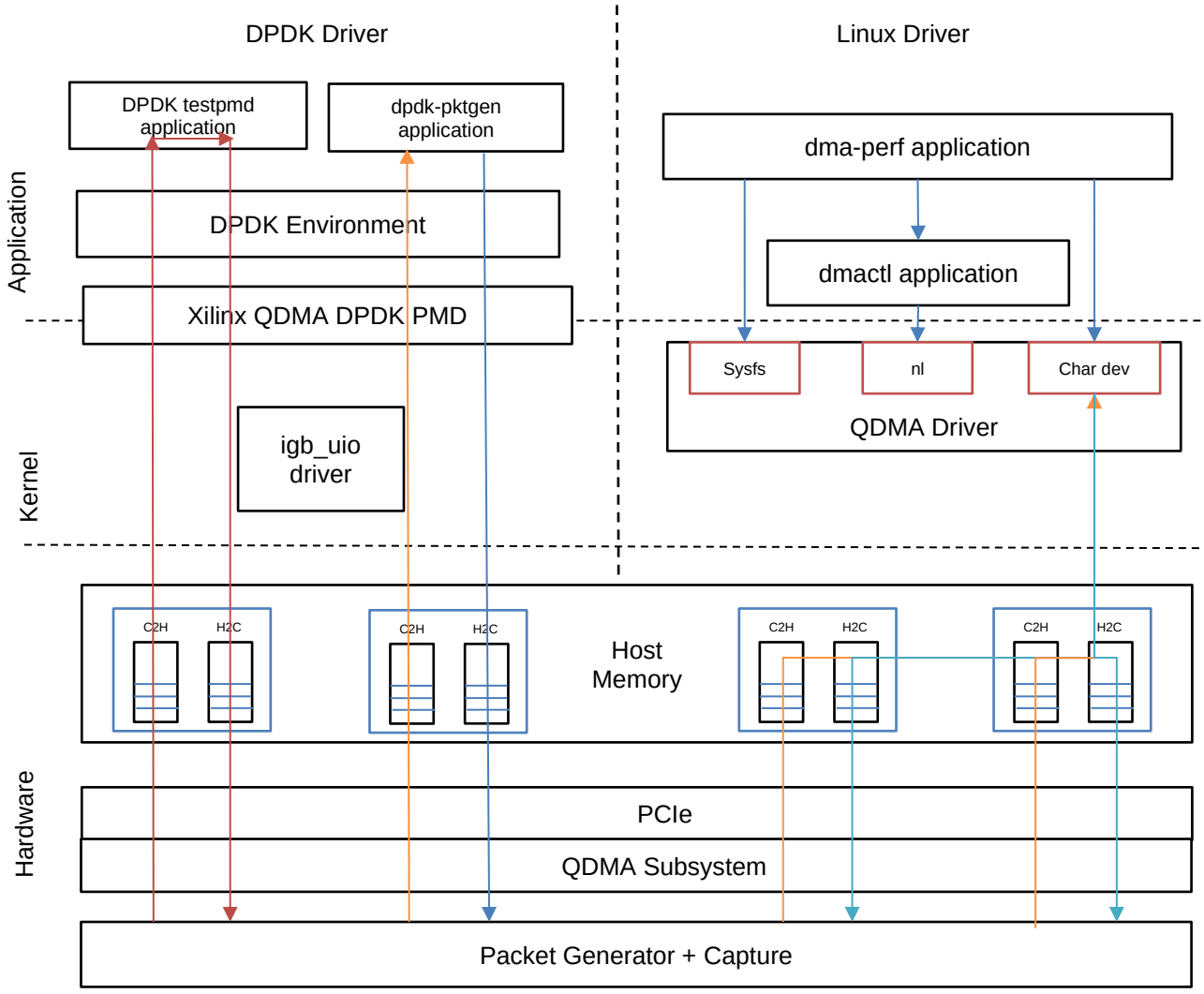


Figure 1: System Diagram

Hardware

Xilinx provides sample reference designs for Streaming (ST) mode and Memory Mapped (MM) mode.

ST performance reference design consists of an AXI Stream-only packet generator in the C2H direction and performance / latency measurement tools in both C2H and H2C directions. The reference design will generate a known data pattern (timestamp) and send a user-specified packet length on the C2H direction when there is an available descriptor. This data pattern can be looped back into the H2C direction by the application and measured for performance and latency. Please refer to the Example Design section in QDMA Subsystem for PCI Express v4.0 - PG302 for information on how to configure the packet generator and read the data collected by the measurement counters through the AXI Lite Master BAR (BAR# 2).

For MM mode, BRAM and DDR based reference designs are provided. For more information on reference designs refer to QDMA Subsystem for PCI Express v4.0 - PG302.

For details regarding register maps and the limitations of the design, please refer to the Reference Design RTL.

Software

Linux Kernel Reference Device Driver

The Xilinx Linux kernel reference driver v2020.1.2.3 is used for collecting the performance numbers.

Xilinx-developed custom tool “dma-perf” is used to collect the performance metrics for unidirectional and bidirectional traffic.

The QDMA Linux kernel reference driver is a PCIe device driver, it manages the QDMA queues in the HW. The driver creates a character device for each queue pair configured,

Standard I/O tools such as ‘fio’ can be used for performing I/O operations using the char device interface.

However, most of the tools are limited to sending / receiving 1 packet at a time and wait for the processing of the packet to complete, so they are not able to keep the driver/ HW busy enough for performance measurement. Although fio also supports asynchronous interfaces, it does not continuously submit I/O requests while polling for the completion parallelly.

To overcome this limitation, Xilinx developed the dma-perf tool. It leverages the asynchronous functionality provided by the libaio library. Using libaio, an application can submit I/O request to the driver and the driver returns the control to the caller immediately (i.e., non-blocking). The completion notification is sent separately, so the application can then poll for the completion and free the buffer upon receiving the completion.

For more information on the dma-perf tools please refer to the QDMA Linux kernel reference driver user guide hosted at https://xilinx.github.io/dma_ip_drivers/master/linux-kernel/html/index.html

DPDK Poll Mode Driver

The Xilinx reference QDMA DPDK 2020.1.1 driver is based on DPDK v19.11. The DPDK driver is tested by binding the PCIe functions with the igb_uio kernel driver.

The dpdk-pktgen application is used to perform uni-directional performance measurement and the testpmd application is used for the Bi-directional forwarding performance measurement.

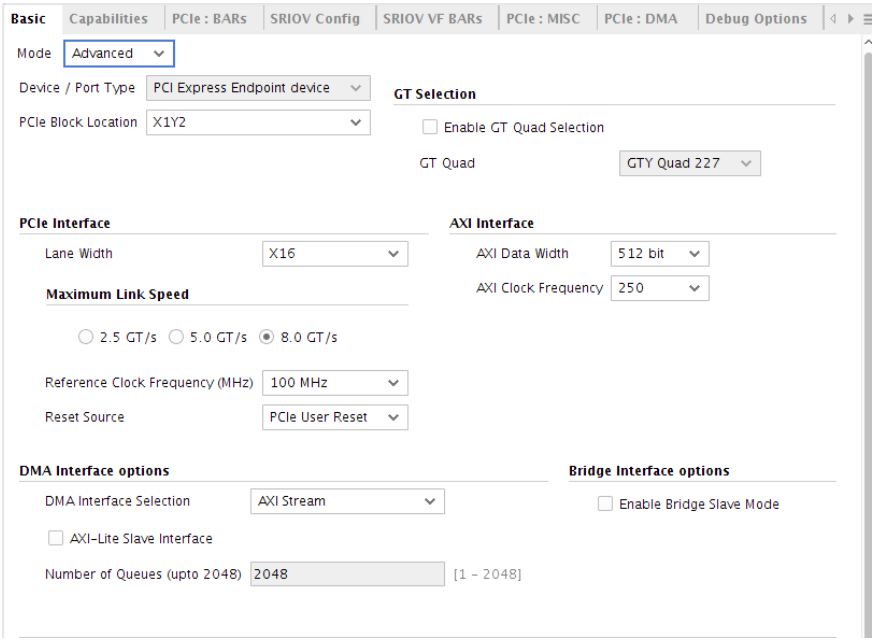
Generating The Reference Design

The Reference Design bitfile used in this Performance report is available for immediate download into a VCU1525 design. For users who are using a different card, the Reference Design can be generated by following these steps:

Create a Vivado project and add/configure a QDMA IP with the following settings – All options not mentioned below can be left at their default settings:

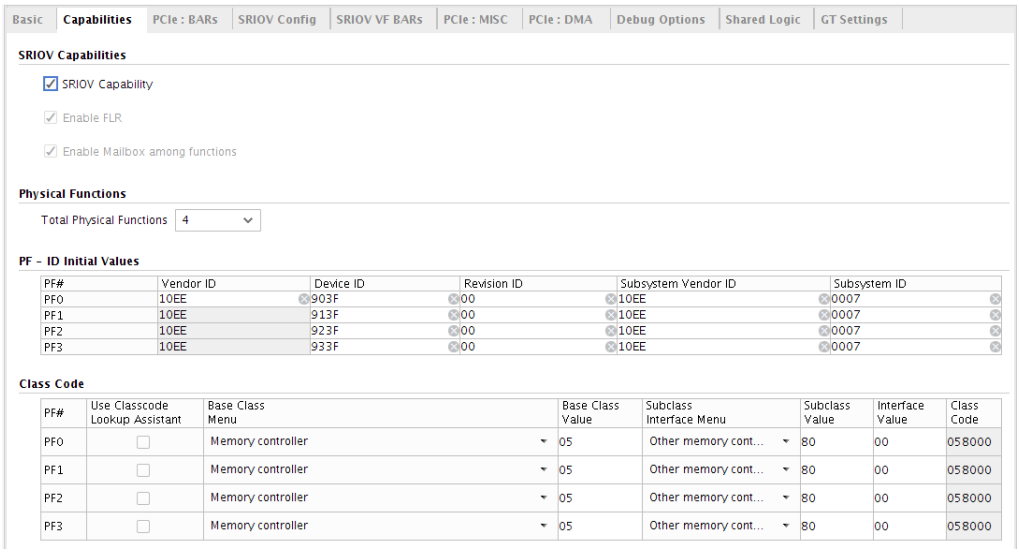
Basic Tab:

- **Mode:** Advanced
  - Lane Width & Link Speed: X16 Gen3 (8.0 GT/s)
  - DMA Interface Selection: AXI Stream



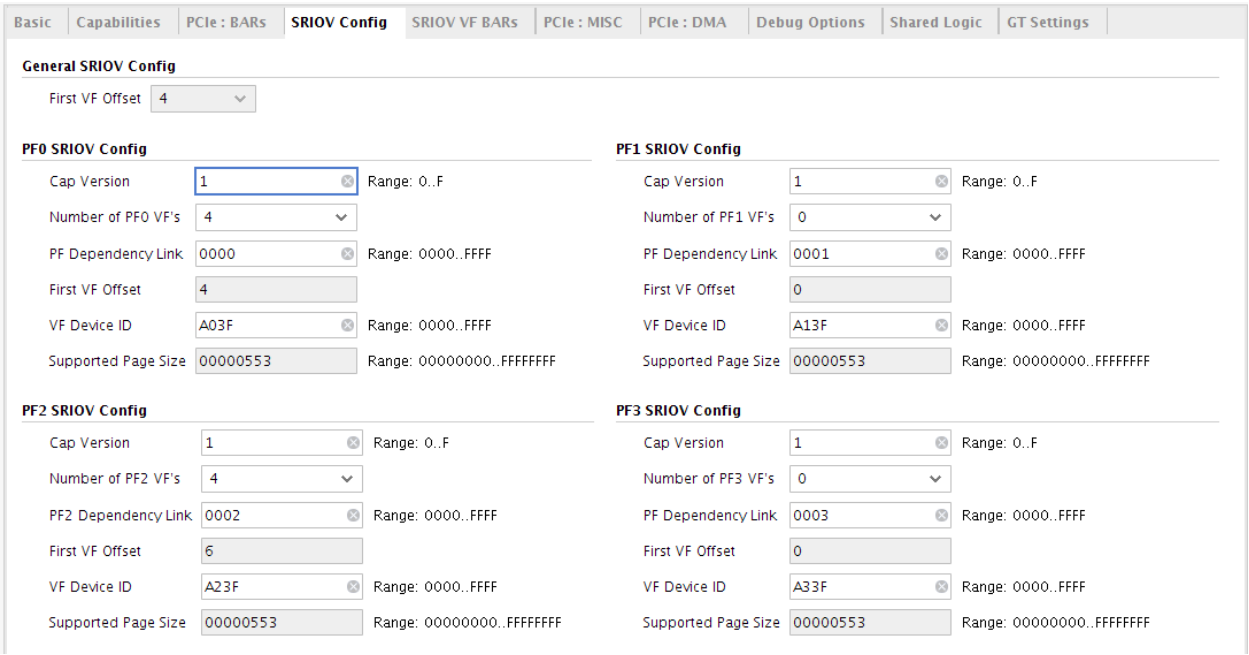
Capabilities Tab:

- Enable SRIOV Capability
- Total Physical Functions: 4



SRIOV Config:

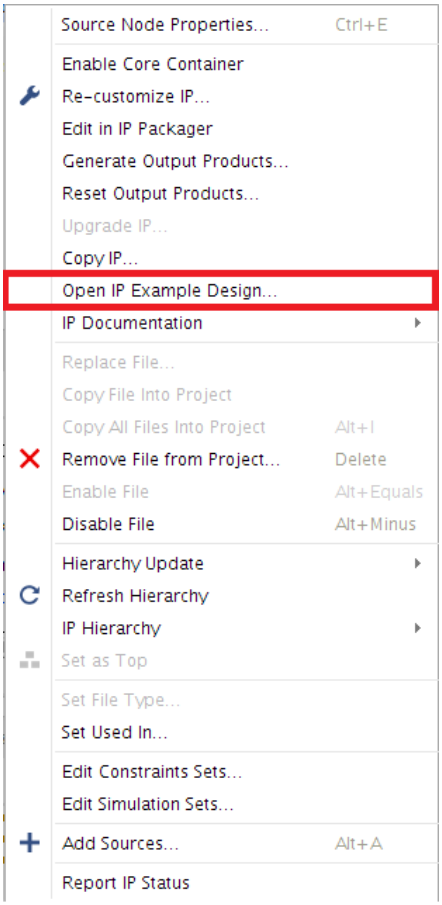
- Number of PF0 VFs: 4
- Number of PF2 VFs: 4



Note: The Reference Design used in this report is an SRIOV capable design with 4PFs and 252VFs. It is not mandatory to enable this feature to use the reference design or achieve the performance reported in this document.

1. Run the following command in the Tcl console to enable Performance Reference Design:
   set_property CONFIG.performance_exdes {true} [get_ips <QDMA_ip_name>]

2. Right click the QDMA IP and choose “Open IP Example Design”



Measurement

For DPDK driver performance analysis, the below performance measurements are taken with dpdk-pktgen and testpmd DPDK applications on PF-0 and VF-0 for this report.

- ST Mode DMA Performance of PF-0 on Host (i.e. Run DPDK performance application on the PF-0 in the Host):
  - C2H only DMA performance using the dpdk-pktgen application
  - H2C only DMA performance using the dpdk-pktgen application
  - Bi-directional (forwarding) DMA performance using the testpmd application
- ST Mode DMA Performance of VF-0 on VM (i.e. Run DPDK application on the PF-0 in the Host which is mainly used for mailbox communication and run DPDK performance application on the VF-0 in the VM):
  - C2H only DMA performance using the dpdk-pktgen application
  - H2C only DMA performance using the dpdk-pktgen application
  - Bi-directional (forwarding) DMA performance using the testpmd application

For Linux Kernel Reference Driver performance analysis, the below performance measurements are taken with the dma-perf tool on PF-0 for host tests and 1 VF created on PF0 for VM tests with driver in the auto (i.e., interrupt aggregation + poll) mode for this report.

- ST Mode DMA Performance:
  - ST-C2H only
  - ST-H2C only
  - ST-H2C & ST-C2H bi-directional
- MM Mode DMA Performance With BRAM Design:
  - MM-C2H only
  - MM-H2C only
  - MM-H2C & MM-C2H bi-directional
- MM Mode DMA Performance With DDR Design:
  - MM-C2H only
  - MM-H2C only
  - MM-H2C & MM-C2H bi-directional

DMA Overheads

The PCIe bandwidth utilization is higher than DMA bandwidth as this number excludes PCIe protocol overheads. In addition to PCIe overhead, DMA will have its own overhead to communicate with the driver as listed below.

- CIDX update by driver affects C2H and forwarding performance
- PIDX update by driver affects H2C and forwarding performance
- 16B H2C descriptor affects H2C and forwarding performance
- 8B C2H descriptor affects C2H and forwarding performance
- C2H completion can be 8B or 16B or 32B or 64B sent for every packet to pass the meta-data.
- Status descriptor writes affect both C2H and H2C performance
- Memory controller overhead in Memory-mapped mode.

When possible, QDMA reduces various TLP overheads by coalescing reads and writes. QDMA is highly customizable, the overheads can be reduced by customizing the solution to be specific to an application.

DMA Bandwidth Performance Measurement

The packets per second (PPS) numbers reported by the application are noted and the DMA bandwidth performance is calculated as below:

- ***DMA Bandwidth Performance = PPS * DMA Packet size in bytes * 8***

For NIC use case the performance can be extrapolated as follows:

- ***Ethernet Performance = PPS * (DMA Packet size in bytes + Preamble bytes + Inter-frame gap bytes + FCS) * 8***

Every Ethernet packet includes Preamble of 8 bytes, Inter-frame Gap of 12 bytes, FCS of 4 bytes and the DMA packet size can be 4 bytes less than the network packet size as the FCS 4 bytes can be stripped off by the MAC and as a result are not DMA'ed.

Latency Measurement

Latency measurement is calculated using the performance counters provided by the Traffic Generator Reference Design. The reference design maintains the minimum and average latency counters. It determines the time taken for a packet to traverse from the C2H path to the H2C path via the testpmd application using a 64-bit timestamp embedded in the packet.

Test Environment

The test setup is as outlined in **Figure 1**. Table 1 lists the system settings used for the performance measurement.

Item	Description
CPU	Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz
Vendor	Supermicro
Model	X11SPi-TF
No of Sockets	1
Number of Physical Cores	28
Threads Per Core	2
No of logical Cores	56
RAM	48GB (6 x 8GB)
DUT	Xilinx VU9P device based VCU1525 board
PCIe Setting	MPS=256, MRRS=512, Extended Tag Enabled, Relaxed Ordering Enabled
Additional settings (Recommended)	Fully populated memory channels Use the CPU slot on which PCIe slot is used
Other Setting (Recommended)	Tx queue depth = 2048, Rx queue depth = 2048, Packet buffer size = 4096, Burst size = 64
Operating System	Ubuntu 18.04.4, Kernel 4.15.0-112-generic
Linux Driver Specifc Settings	
Boot Setting	"intel_idle.max_cstate=0 processor.max_cstate=0 intel_pstate=disable rcu_nocb_poll audit=0 pci=realloc nosoftlockup iommu=pt intel_iommu=on"
BIOS Version	2.1
BIOS Settings	ACS, IOMMU, ARI enabled. Power Technology is set to Custom, Power Performance Tuning – OS controls EPB
VM Settings for Linux VF Performance	VM RAM = 32G, Number of cores for VM = 32
DPDK driver specific settings:	

Boot Setting	default_hugepagesz=1GB hugepagesz=1G hugepages=30 iommu=pt intel_iommu=on pci=realloc transparent_hugepages=never isolcpus=0-16 rcu_nocbs=0-16 nohz=on nohz_full=0-16 numa_balancing=disable nmi_watchdog=1 audit=0 nosoftlockup hpet=disable tsc=reliable selinux=0
DPDK Version	19.11
Performance setting	Disable iptables/ip6tables service Disable irqbalance service Disable cpuspeed service Point scaling-governor to performance

Table 1: Performance System Settings

There could be some variation in performance with other systems based on system settings.

The following are additional DPDK Performance tuning settings that could be used.

PCIe MRRS

The Performance System Settings listed in Table 1 have PCIe MRRS (Max Read Request) of 512, with which all performance data has been collected. However, depending on the system in use (motherboard, PCIe root ports and its capability, etc.), system specific tuning might be needed. For example, if line-rate is not observed for a packet size of 4K on a system, then PCIe MRRS could be tuned to 2048.

CPU isolation and additional kernel settings

If the system in use runs additional apps, then it is always good to set aside CPU cores for the dpdk workload.
The below additional kernel command-line parameters could be added to GRUB boot settings:

```
isolcpus=1-<n> nohz_full=1-<n> rcu_nocbs=1-<n>
```

where:
`isolcpus` isolates cpus 1 to <n> from kernel scheduling so that they are set aside for dedicated tasks like dpdk.
`nohz_full` will put cpus 1 to <n> in adaptive-ticks mode so as not to interrupt them with scheduling-clock interrupts.
`rcu_nocbs` will fence off cpus 1 to <n> from random interruptions of softirq RCU callbacks.

Now, use from these CPUs 1 to <n> in DPDK testpmd, pktgen command-lines to make sure that the DPDK apps use these dedicated CPUs.

Disable CPU power savings from the kernel command-line

Sometimes, disabling CPU power savings from the BIOS alone might not make the CPU(s) run on full horse-power. As an additional confirmation, disable power savings from the kernel command-line too:

```
processor.max_cstate=0 intel_idle.max_cstate=0 intel_pstate=disable
```

The above will disable power savings on CPUs and disable intel_idle as well as the intel_pstate CPU frequency scaling driver. In addition, always double-confirm from '`cat /proc/cpuinfo | grep -i Mhz`' output that all of the intended CPUs are operating at max speed.

NOTE:
1. The above tuning parameters might not always be needed. The user should be thoroughly familiar with the system in use, and the kind of apps/workload running on the system before applying any of these parameters.
2. Some parameters are specific to Intel x86-64 and might not work for ARM, PPC or AMD based systems. Always consult the respective CPU's manual and related Linux parameters for the same.
3. These parameters were NOT used for the performance numbers published in this guide.

lspci output

Figure 2 depicts sample lspci output of the PCIe function under test.

```
65:00.0 Memory controller: Xilinx Corporation Device 903f
        Subsystem: Xilinx Corporation Device 0007
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr+ Stepping- SERR+ FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0, Cache Line Size: 32 bytes
        Interrupt: pin A routed to IRQ 308
        Region 0: Memory at 38bffff60000 (64-bit, prefetchable) [size=128K]
        Region 2: Memory at 38bffffd3000 (64-bit, prefetchable) [size=4K]
        Capabilities: [40] Power Management version 3
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0-,D1-,D2-,D3hot-,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [60] MSI-X: Enable+ Count=8 Masked-
                Vector table: BAR=0 offset=00010000
                PBA: BAR=0 offset=00014000
        Capabilities: [70] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 1024 bytes, PhantFunc 0, Latency L0s <64ns, L1 <1us
                        ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset+ SlotPowerLimit 0.000W
                DevCtl: Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported-
                        RlxdOrd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+ FLReset-
                        MaxPayload 256 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend+
                LnkCap: Port #0, Speed 8GT/s, Width x16, ASPM not supported, Exit Latency L0s unlimited, L1 unlimited
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 8GT/s, Width x16, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range BC, TimeoutDis+, LTR-, OBFF Not Supported
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
                LnkCtl2: Target Link Speed: 8GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
                         EqualizationPhase2+, EqualizationPhase3+, LinkEqualizationRequest-
        Capabilities: [100 v1] Advanced Error Reporting
                UESta:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
                UEMsk:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq+ ACSViol-
                UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- UnxCmplt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-
                CESta:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
                CEMsk:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
                AERCap: First Error Pointer: 00, GenCap- CGenEn- ChkCap- ChkEn-
        Capabilities: [140 v1] Single Root I/O Virtualization (SR-IOV)
                IOVCap: Migration-, Interrupt Message Number: 000
                IOVCtl: Enable- Migration- Interrupt- MSE- ARIHierarchy+
                IOVSta: Migration-
                Initial VFs: 4, Total VFs: 4, Number of VFs: 0, Function Dependency Link: 00
                VF offset: 4, stride: 1, Device ID: a03f
                Supported Page Size: 00000553, System Page Size: 00000001
                Region 0: Memory at 000038bffffb0000 (64-bit, prefetchable)
                Region 2: Memory at 000038bffffcc000 (64-bit, prefetchable)
                VF Migration: offset: 00000000, BIR: 0
        Capabilities: [180 v1] Alternative Routing-ID Interpretation (ARI)
                ARICap: MFVC- ACS-, Next Function: 1
                ARICtl: MFVC- ACS-, Function Group: 0
        Capabilities: [1c0 v1] #19
        Kernel driver in use: igb_uio
```

Figure 2: lspci output of the PCIe function being tested

QDMA Settings

The QDMA IP is highly configurable. This section lists only a subset of the available settings limited to the tests carried out in this report.

Configuration Option	Description
Number of Queues	The QDMA IP supports up to 2048 queues. The number of queues for a given test are specified when starting the software application. ***Benchmarking is done for 1, 2, 4 and 8 queues.***

Packet Size	Tests accept a range of packet sizes along with a packet size increment. ***Benchmarking is done with packet size in the range of 64B to 4KB for streaming queue performance and 64B to 32KB for memory mapped queue performance***
Completion Descriptor size	Completion queue descriptors can be configured to 8-byte, 16-byte, 32-byte or 64-byte. ***Benchmarking is done with 16B descriptor format.***
Descriptor Prefetch	Prefetch causes descriptors to be opportunistically prefetched so that descriptors are available before the packet is received. ***Benchmarking is done with prefetch enabled***.
Prefetch Cache depth	Prefetch cache depth is selectable from Vivado when building the design. Supported values are 8, 16, 32, 64. The Prefetch cache can support that many active queues at any given time. ***Benchmarking is done with prefetch cache depth set to 64.***
Completion Coalesce Buffer depth	Completion coalesce buffer depth is selectable from Vivado when building the design. Supported values are 8, 16, 32. ***Benchmarking is done with completion coalesce buffer depth set to 32.***

Table 2: QDMA Settings

Performance Benchmark Results

DPDK Driver

This section provides the performance results captured using DPDK driver in streaming mode using the customized bitstream provided in release package.

QDMA currently has a limitation that the packet buffers be aligned to the 256 bytes boundary for optimal DMA performance. Because the mbuf data pointers in DPDK need not be aligned to the 256 bytes boundary, the higher packet size performance for DPDK reported below is slightly lower than that reported for the Linux driver. This alignment limitation in the QDMA IP will be fixed in future IP releases.

Streaming Mode C2H and H2C performance test

The below dpdk-pktgen command-lines were used for performance measurement.

- The '-9' option is the extension added by Xilinx to enable dpdk-pktgen to support packet sizes beyond 1518 bytes.
- The dpdk-pktgen application was also modified to disable the packet classification.

The '-w' EAL option is specified to enable or disable prefetch and to change the completion descriptor length.
In the table below, 3b:00.0 represents PCIe function in "bus:device.function" format.

# of queues	dpdk-pktgen command line
1	./app/build/pktgen -l 0-2 -n 4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -P -m "[1:2].0" -9
2	./app/build/pktgen -l 0-4 -n 4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -P -m "[1-2:3-4].0" -9

4	./app/build/pktgen -l 0-8 -n 4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -P -m "[1-4:5-8].0" -9
8	./app/build/pktgen -l 0-16 -n 4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 --P -m "[1-8:9-16].0" -9

Table 3: Command-line for dpdk-pktgen application

PF Performance:

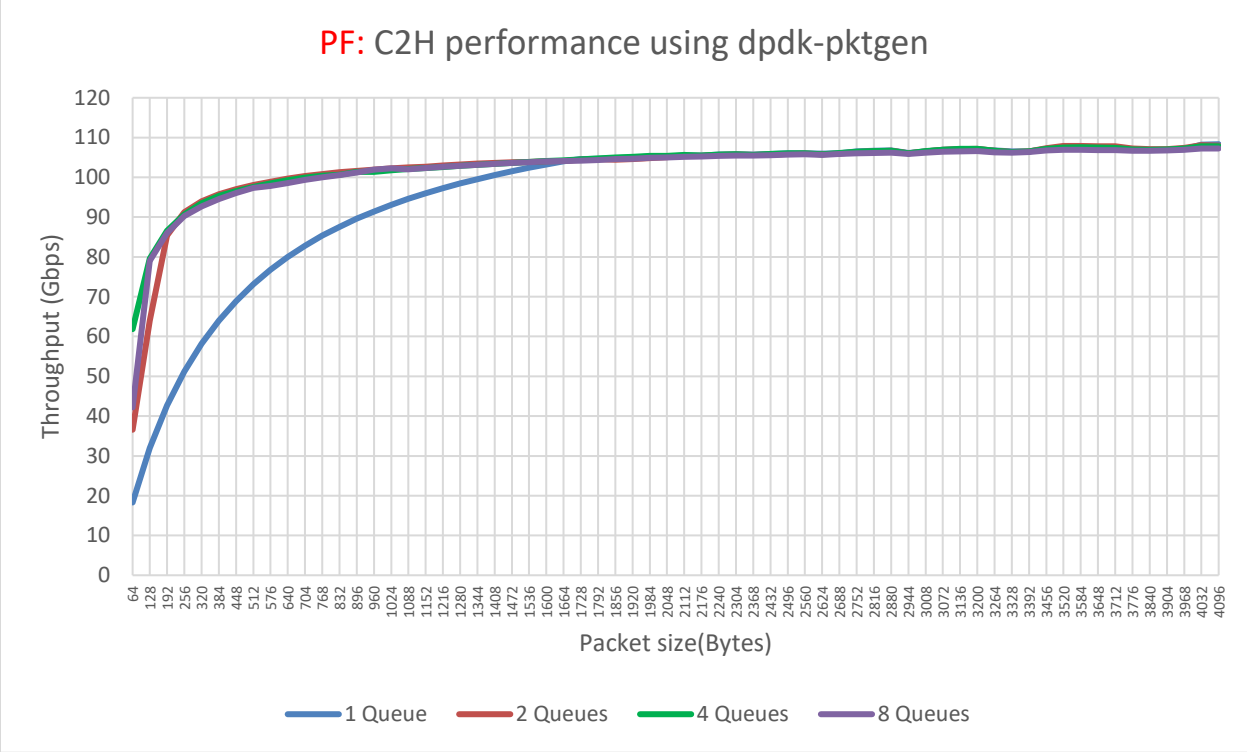


Figure 3: DPDK Driver – QDMA4.0 PF ST C2H performance in Gbps

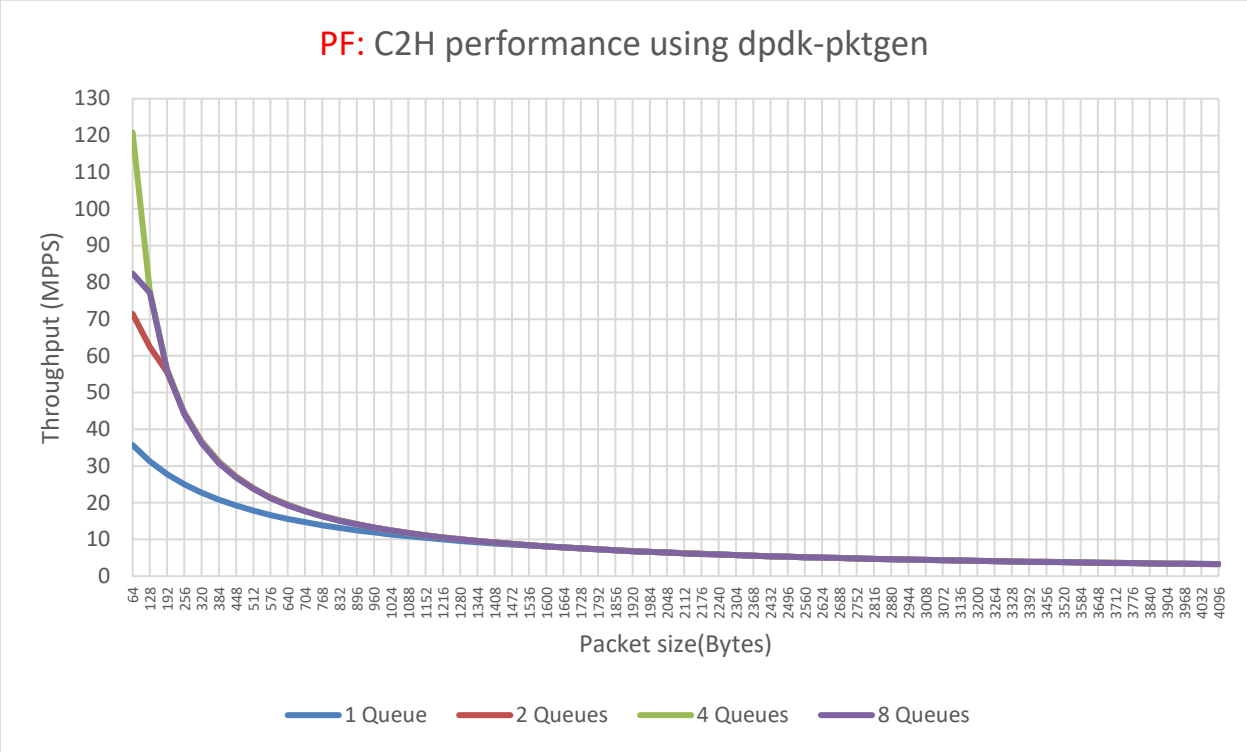


Figure 4: DPDK Driver – QDMA4.0 PF ST C2H performance in Mpps

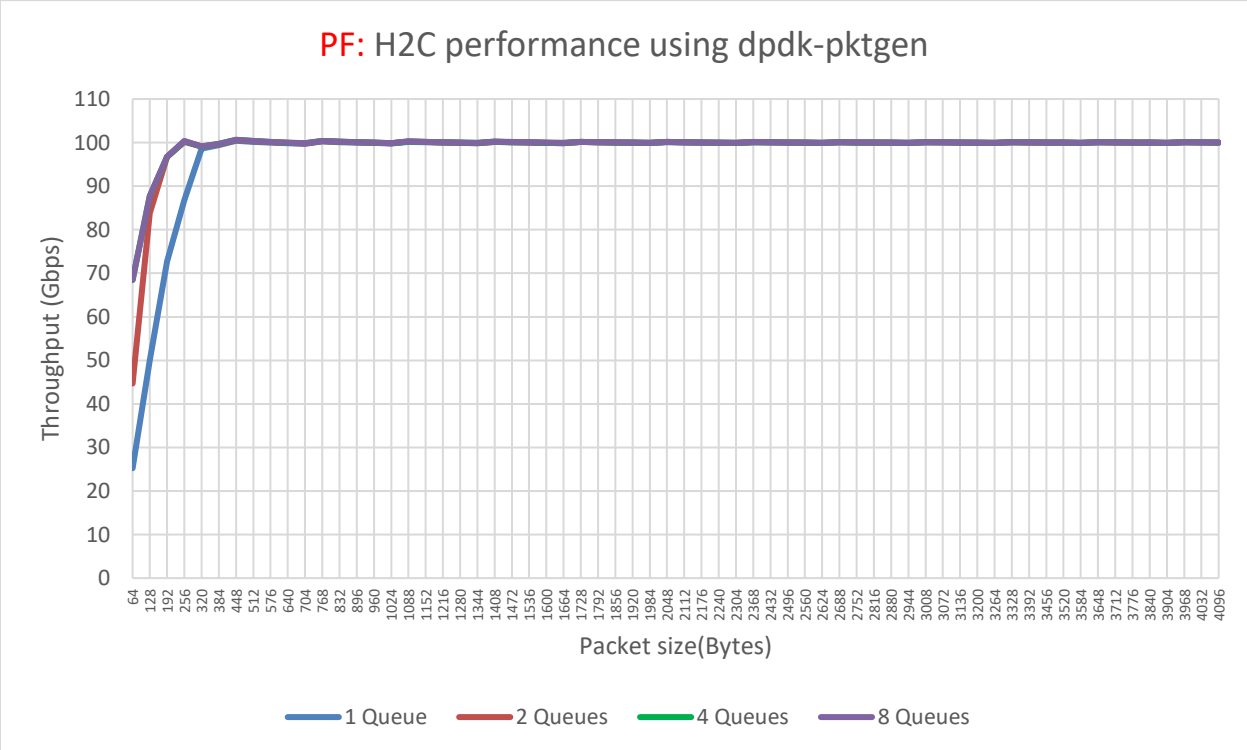


Figure 5: DPDK Driver – QDMA4.0 PF ST H2C Performance in Gbps

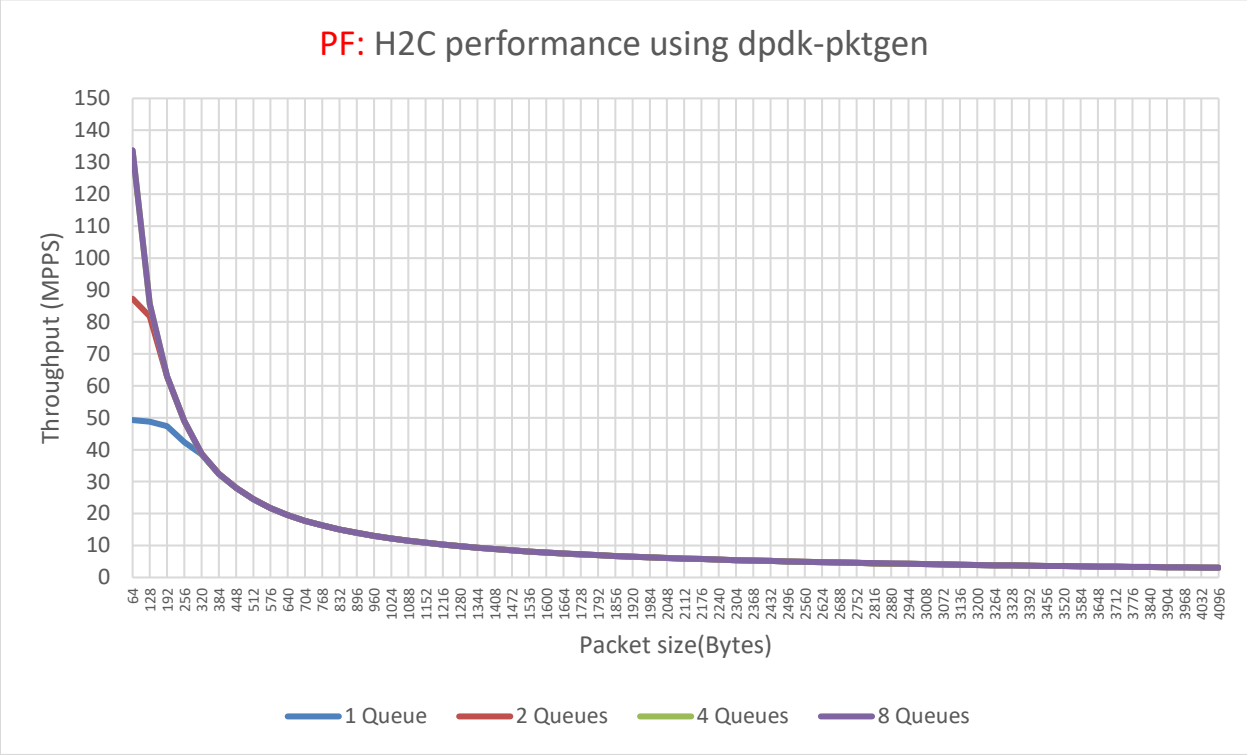


Figure 6: DPDK Driver – QDMA4.0 PF ST H2C Performance in Mpps

VF Performance:

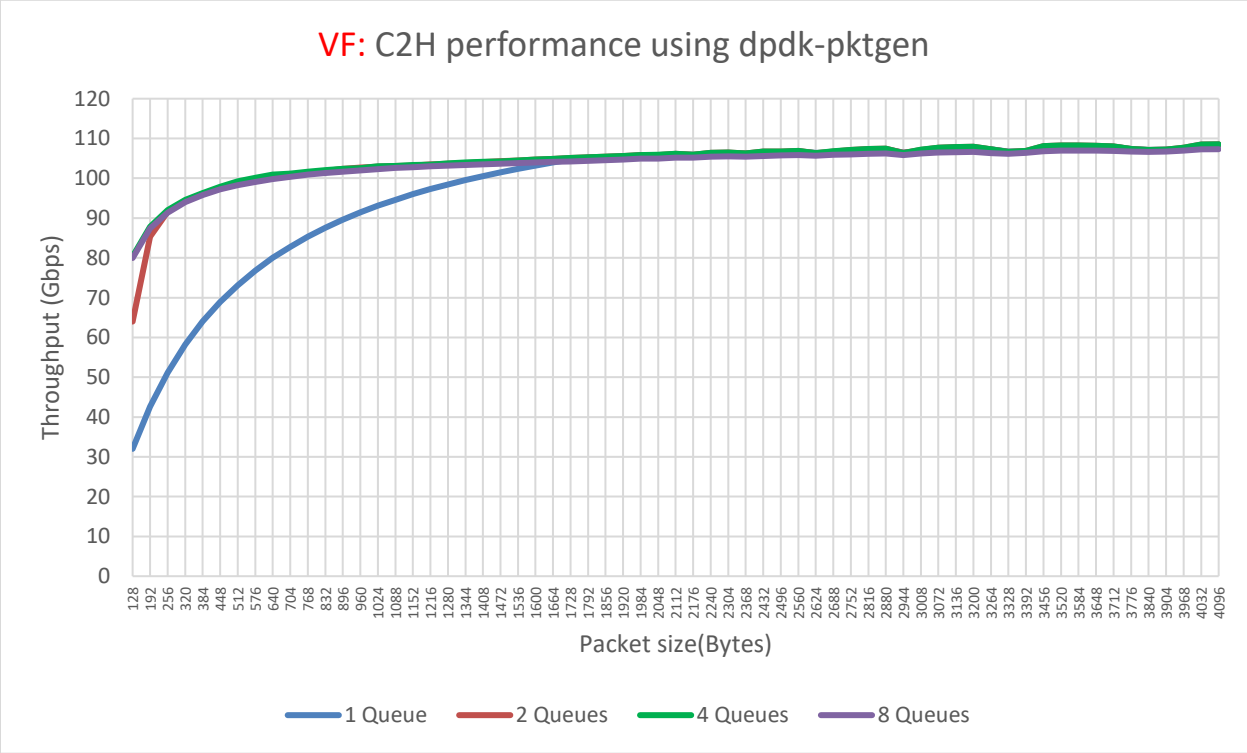


Figure 7: DPDK Driver – QDMA4.0 VF ST C2H performance in Gbps

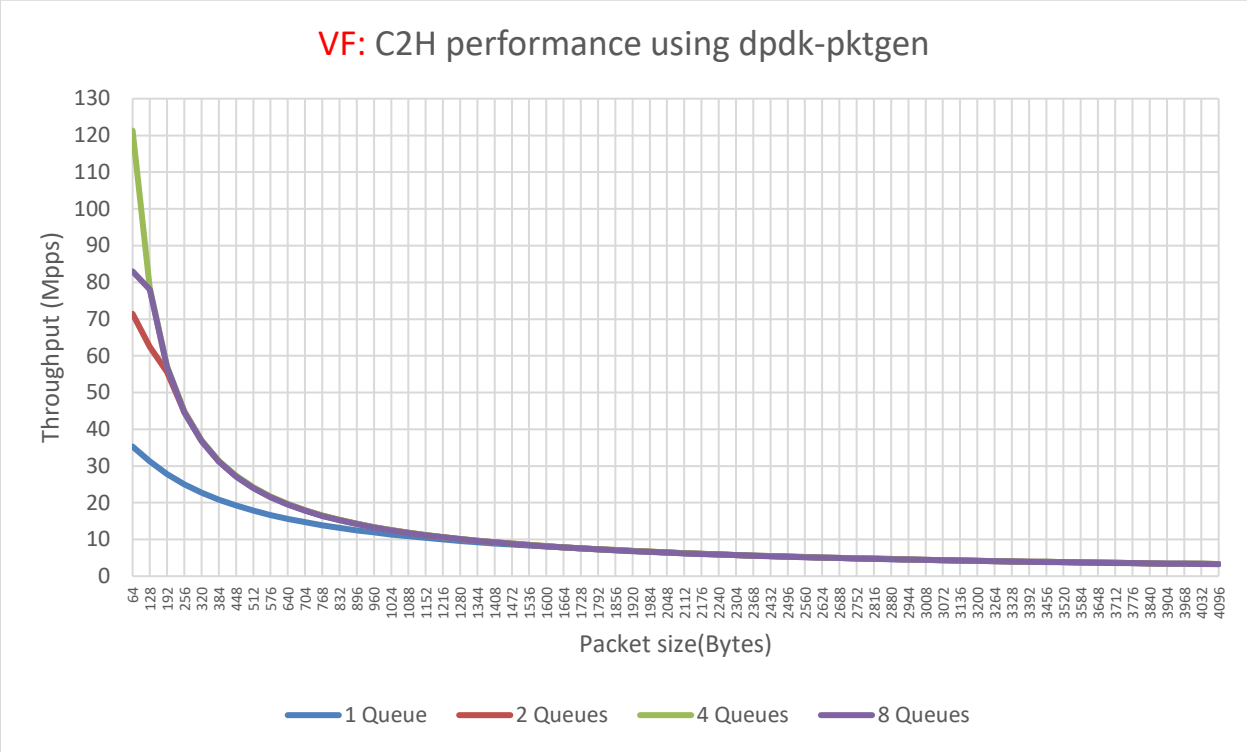


Figure 8: DPDK Driver – QDMA4.0 VF ST C2H performance in Mpps

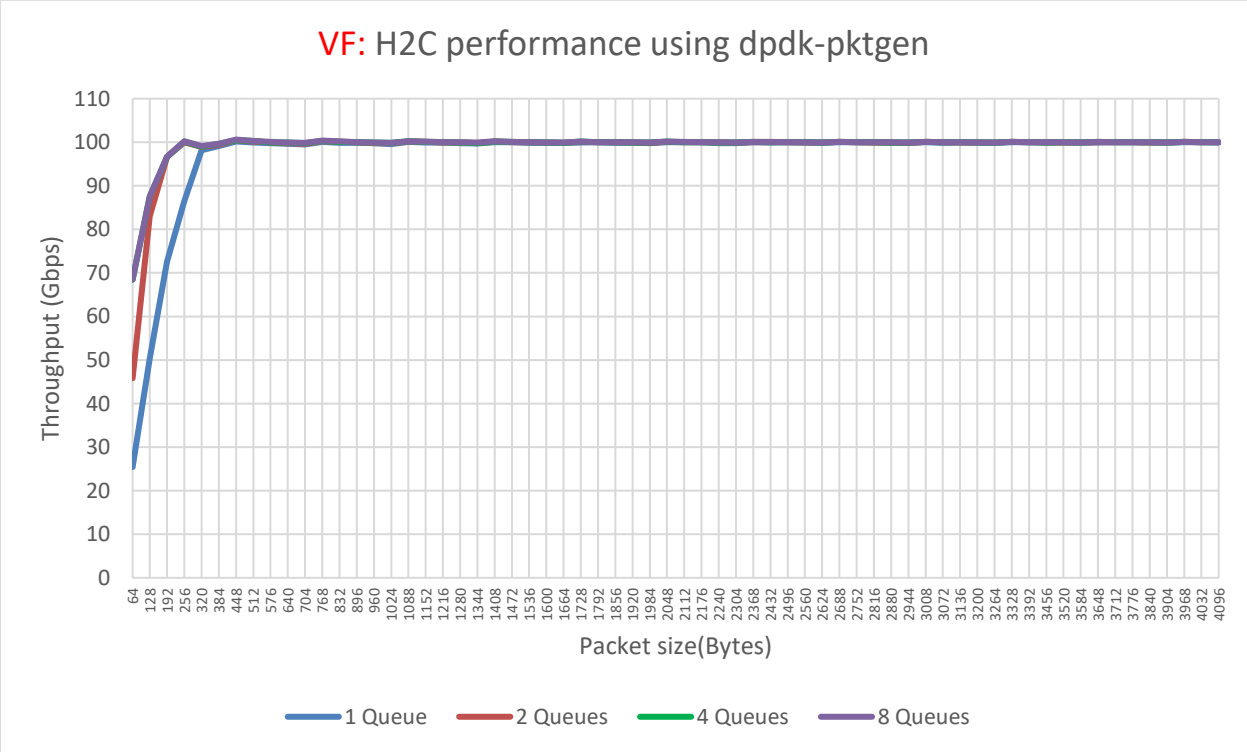


Figure 9: DPDK Driver – QDMA4.0 VF ST H2C Performance in Gbps

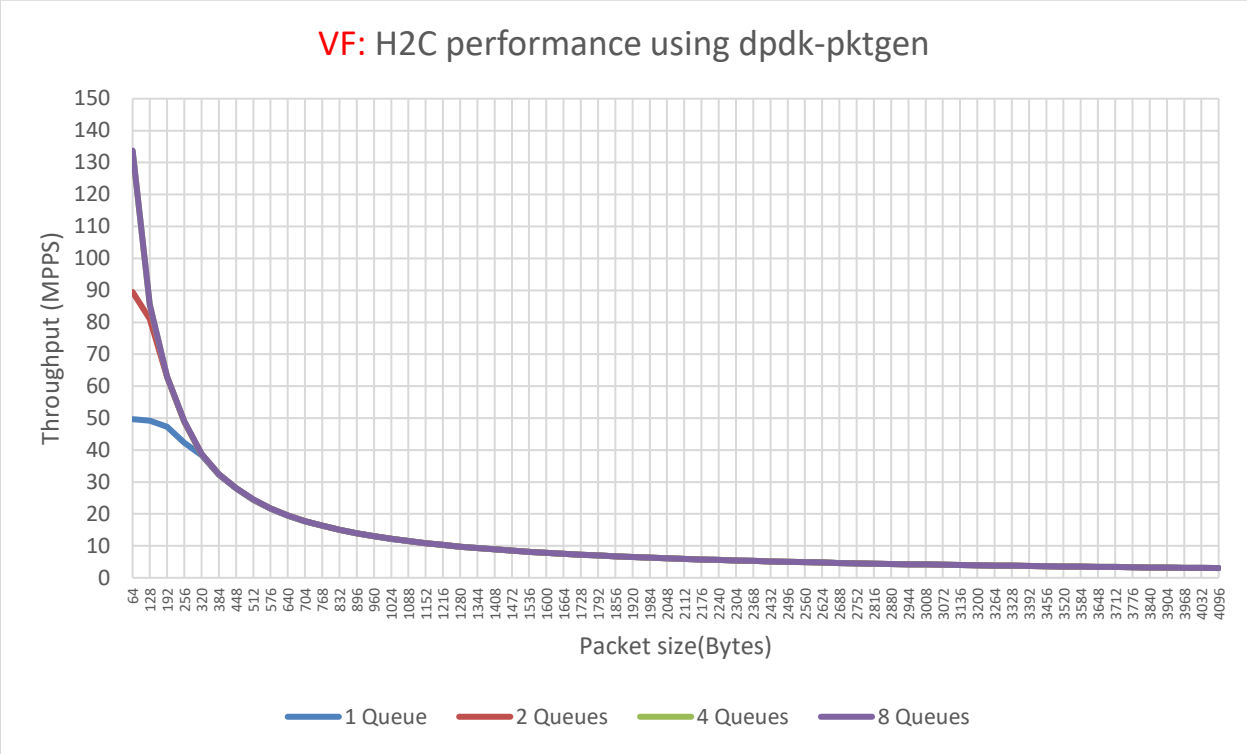


Figure 10: DPDK Driver – QDMA4.0 VF ST H2C Performance in Mpps

For H2C performance tests, the C2H traffic is disabled and H2C packets are generated using the dpdk-pktgen application. The EAL option (-w) is ***not*** required to be specified in the command lines.

Streaming Mode Forwarding performance test

The testpmd application is executed with the below command-line options for different queue configurations.

# of queues	testpmd command line
1	./build/app/testpmd -cf -n4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -i --nb-cores=2 --rxq=1 --txq=1 --rxd=2048 --txd=2048 --burst=64 --mbuf-size=4224
2	./build/app/testpmd -cff -n4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -i --nb-cores=3 --rxq=2 --txq=2 --rxd=2048 --txd=2048 --burst=64 --mbuf-size=4224
4	./build/app/testpmd -cfff -n4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -i --nb-cores=5 --rxq=4 --txq=4 --rxd=2048 --txd=2048 --burst=64 --mbuf-size=4224
8	./build/app/testpmd -cffff -n4 -w 3b:00.0,desc_prefetch=1,cmpt_desc_len=16 -- -i --nb-cores=9 --rxq=8 --txq=8 --rxd=2048 --txd=2048 --burst=64 --mbuf-size=4224

Table 4: Command-line for testpmd application

PF Performance

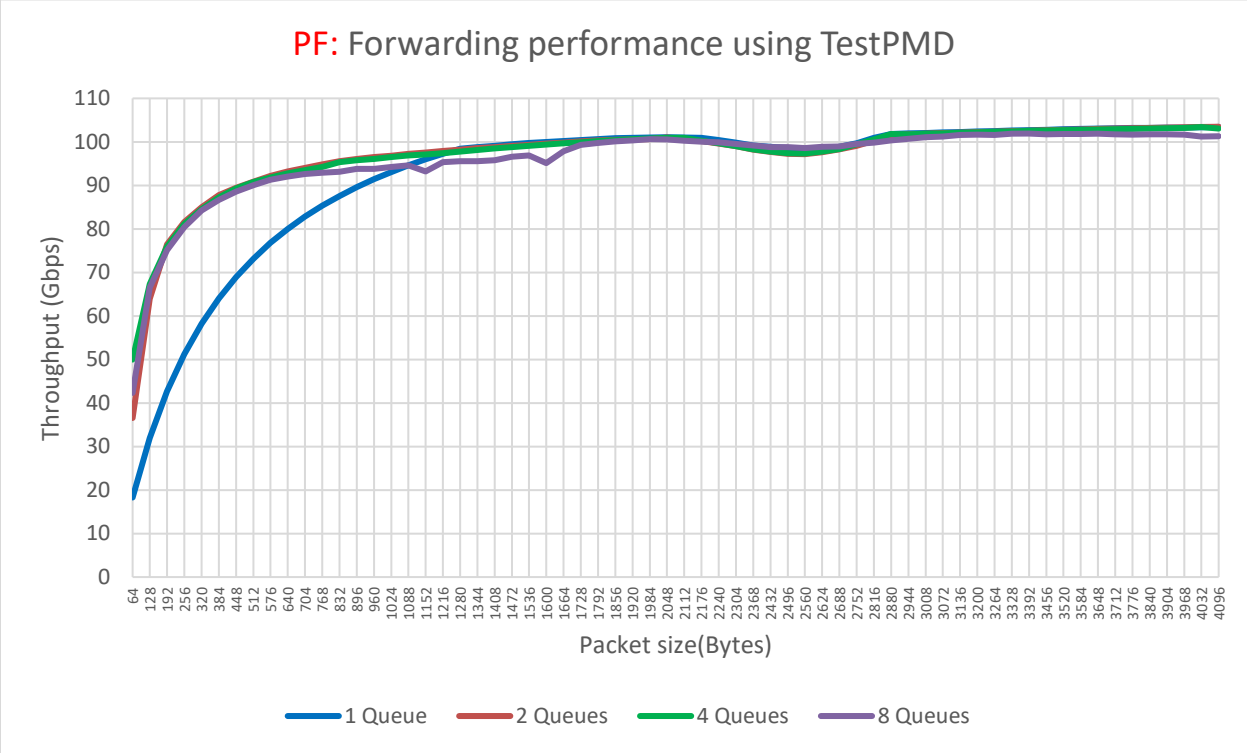


Figure 11: DPDK Driver – QDMA4.0 PF Forwarding performance in Gbps

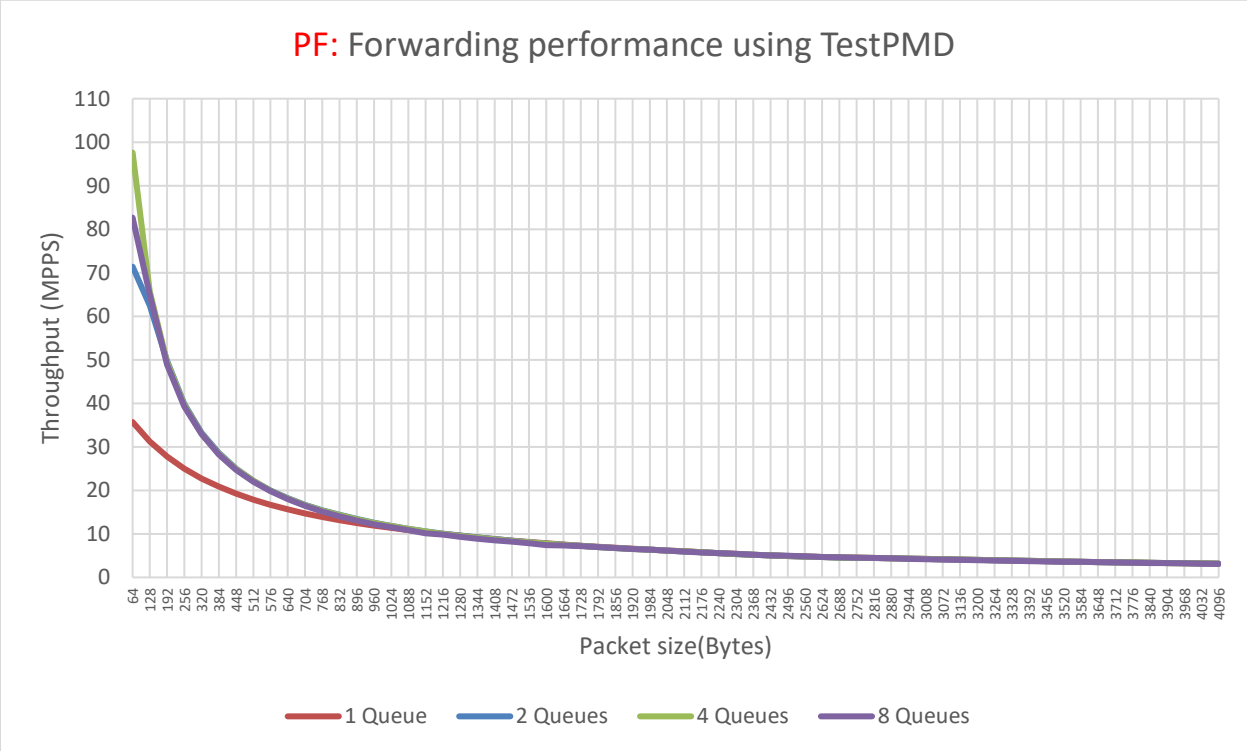


Figure 12: DPDK Driver – QDMA4.0 PF Forwarding performance in Mpps

Packet Size (Bytes)	8 queues		4 queues		2 queues		1 queue	
	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)
64	82.66	42.32	97.61	49.98	71.39	36.55	35.70	18.28
128	64.95	66.51	65.73	67.31	62.48	63.98	31.24	31.99
192	48.91	75.13	49.45	75.96	49.78	76.46	27.77	42.66
256	39.25	80.39	39.61	81.12	39.84	81.59	25.00	51.20
320	32.90	84.22	33.03	84.55	33.20	85.00	22.73	58.18
384	28.21	86.68	28.40	87.23	28.56	87.74	20.83	64.00
448	24.72	88.59	24.93	89.34	24.96	89.47	19.23	68.92
512	21.98	90.03	22.17	90.80	22.19	90.87	17.86	73.14
576	19.81	91.26	19.93	91.83	20.01	92.19	16.67	76.80
640	17.98	92.04	18.12	92.76	18.21	93.21	15.63	80.00
704	16.45	92.64	16.60	93.51	16.70	94.04	14.71	82.82
768	15.13	92.94	15.34	94.23	15.43	94.81	13.89	85.33
832	13.99	93.14	14.32	95.31	14.36	95.55	13.16	87.58
896	13.09	93.81	13.36	95.76	13.41	96.09	12.50	89.60
960	12.22	93.82	12.51	96.07	12.56	96.49	11.90	91.43
1024	11.51	94.26	11.79	96.55	11.82	96.81	11.36	93.09
1088	10.87	94.58	11.14	96.92	11.17	97.22	10.87	94.61
1152	10.12	93.22	10.54	97.14	10.59	97.57	10.42	96.00
1216	9.80	95.35	10.01	97.40	10.06	97.90	10.00	97.28
1280	9.33	95.54	9.55	97.77	9.59	98.24	9.62	98.46
1344	8.89	95.59	9.13	98.13	9.17	98.55	9.19	98.80
1408	8.51	95.81	8.74	98.47	8.77	98.82	8.80	99.12
1472	8.20	96.61	8.39	98.80	8.41	99.09	8.45	99.46
1536	7.88	96.86	8.07	99.11	8.08	99.34	8.12	99.75
1600	7.43	95.15	7.76	99.39	7.78	99.60	7.81	100.00
1664	7.35	97.84	7.49	99.65	7.50	99.83	7.53	100.23
1728	7.18	99.31	7.23	99.93	7.24	100.09	7.27	100.43
1792	6.96	99.73	6.99	100.20	7.00	100.32	7.02	100.66
1856	6.74	100.09	6.76	100.44	6.77	100.52	6.79	100.83
1920	6.53	100.35	6.55	100.64	6.55	100.59	6.57	100.93
1984	6.34	100.67	6.35	100.77	6.34	100.67	6.36	101.00
2048	6.14	100.57	6.16	100.90	6.16	101.01	6.17	101.04
2112	5.94	100.29	5.97	100.94	5.96	100.63	5.98	101.03
2176	5.75	100.05	5.76	100.29	5.76	100.19	5.80	100.94
2240	5.57	99.90	5.56	99.66	5.56	99.62	5.60	100.39
2304	5.40	99.53	5.37	99.02	5.37	98.98	5.42	99.82
2368	5.23	99.14	5.19	98.32	5.19	98.24	5.24	99.20
2432	5.08	98.78	5.02	97.74	5.02	97.71	5.08	98.83

2496	4.95	98.83	4.88	97.37	4.87	97.25	4.94	98.56
2560	4.81	98.56	4.75	97.31	4.75	97.19	4.80	98.29
2624	4.71	98.86	4.66	97.74	4.65	97.64	4.69	98.37
2688	4.60	98.91	4.57	98.37	4.57	98.27	4.60	98.86
2752	4.52	99.50	4.52	99.42	4.50	99.10	4.53	99.68
2816	4.43	99.82	4.46	100.54	4.45	100.25	4.48	100.93
2880	4.36	100.36	4.42	101.79	4.41	101.50	4.42	101.82
2944	4.28	100.70	4.32	101.77	4.32	101.77	4.33	101.95
3008	4.20	101.08	4.23	101.87	4.23	101.89	4.24	102.06
3072	4.12	101.24	4.15	101.97	4.15	102.02	4.16	102.18
3136	4.05	101.60	4.07	102.10	4.07	102.14	4.08	102.27
3200	3.97	101.68	3.99	102.19	4.00	102.29	4.00	102.40
3264	3.89	101.62	3.92	102.31	3.92	102.36	3.93	102.51
3328	3.83	101.86	3.85	102.43	3.85	102.47	3.85	102.60
3392	3.76	101.91	3.78	102.52	3.78	102.58	3.78	102.68
3456	3.68	101.76	3.71	102.61	3.71	102.67	3.72	102.78
3520	3.62	101.83	3.65	102.69	3.65	102.76	3.65	102.88
3584	3.55	101.85	3.58	102.77	3.59	102.86	3.59	102.97
3648	3.49	101.88	3.52	102.84	3.53	102.96	3.53	103.03
3712	3.43	101.73	3.47	102.92	3.47	103.03	3.47	103.11
3776	3.37	101.67	3.41	102.98	3.41	103.11	3.42	103.17
3840	3.31	101.73	3.35	103.05	3.36	103.19	3.36	103.22
3904	3.26	101.71	3.30	103.12	3.31	103.26	3.31	103.27
3968	3.20	101.68	3.25	103.19	3.26	103.34	3.25	103.32
4032	3.14	101.26	3.21	103.41	3.21	103.41	3.20	103.38
4096	3.09	101.27	3.14	103.03	3.16	103.48	3.16	103.44

Table 5: DPDK Driver – QDMA4.0 PF Forwarding performance test results

VF Performance

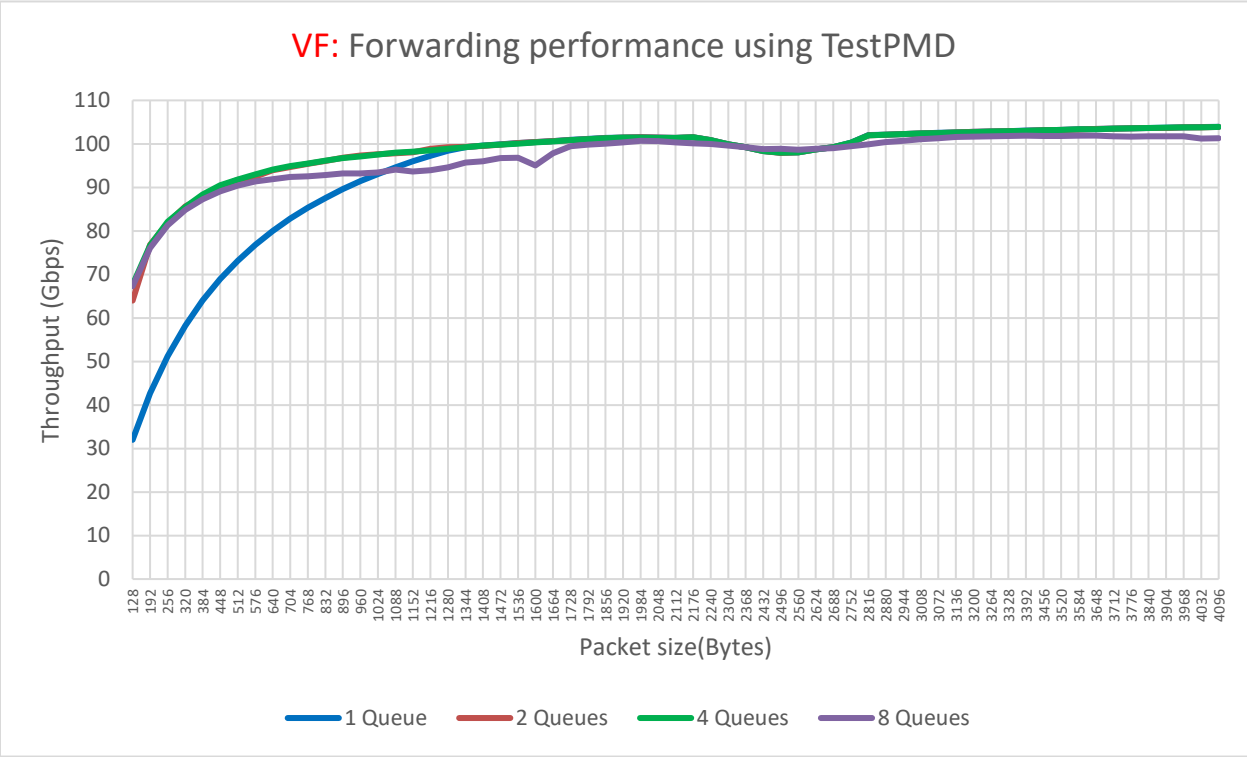


Figure 13: DPDK Driver – QDMA4.0 VF Forwarding performance in Gbps

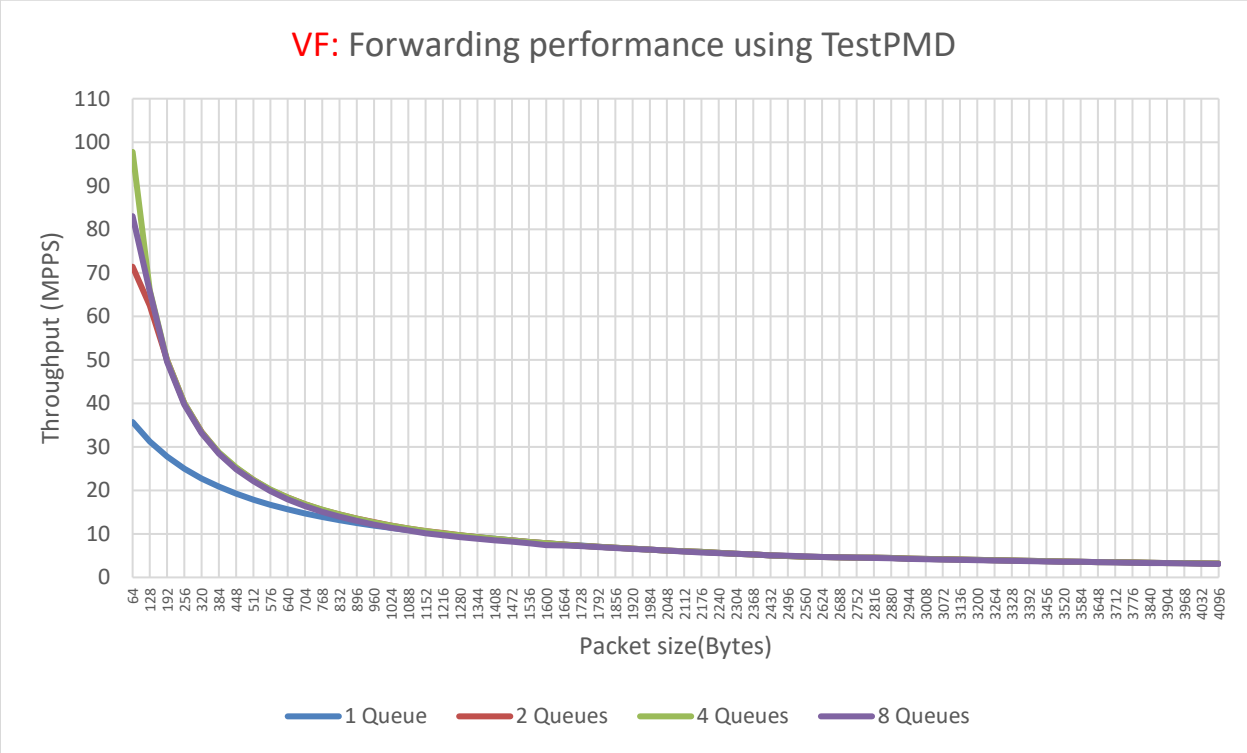


Figure 14: DPDK Driver – QDMA4.0 VF Forwarding performance in Mpps

Packet Size (Bytes)	8 queues		4 queues		2 queues		1 queue	
	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)	Packet rate (Mpps)	Throughput (Gbps)
64	83.00	42.50	97.77	50.06	71.39	36.55	35.70	18.28
128	65.72	67.29	66.03	67.62	62.49	63.99	31.24	31.99
192	49.55	76.11	49.92	76.67	50.00	76.81	27.78	42.66
256	39.68	81.26	40.08	82.08	39.94	81.81	25.00	51.20
320	33.15	84.86	33.39	85.48	33.46	85.67	22.73	58.18
384	28.41	87.26	28.76	88.34	28.56	87.74	20.83	64.00
448	24.86	89.10	25.26	90.52	25.16	90.17	19.23	68.92
512	22.08	90.45	22.41	91.80	22.39	91.73	17.86	73.14
576	19.83	91.39	20.18	92.99	20.06	92.44	16.67	76.80
640	17.95	91.92	18.38	94.12	18.35	93.94	15.62	80.00
704	16.41	92.40	16.85	94.92	16.81	94.66	14.71	82.82
768	15.07	92.57	15.54	95.46	15.53	95.40	13.89	85.33
832	13.95	92.87	14.44	96.14	14.44	96.11	13.16	87.58
896	13.00	93.21	13.49	96.71	13.50	96.78	12.50	89.60
960	12.13	93.19	12.64	97.09	12.67	97.30	11.90	91.43
1024	11.40	93.40	11.90	97.51	11.92	97.61	11.36	93.09
1088	10.81	94.08	11.26	97.98	11.24	97.87	10.87	94.61
1152	10.16	93.64	10.66	98.22	10.64	98.07	10.42	96.00
1216	9.66	93.94	10.12	98.49	10.16	98.88	10.00	97.28
1280	9.24	94.59	9.66	98.90	9.69	99.26	9.62	98.46
1344	8.90	95.74	9.23	99.24	9.24	99.31	9.23	99.25
1408	8.52	96.01	8.84	99.58	8.84	99.52	8.84	99.58
1472	8.21	96.73	8.48	99.83	8.48	99.83	8.48	99.89
1536	7.88	96.81	8.14	100.08	8.15	100.17	8.15	100.17
1600	7.43	95.05	7.84	100.34	7.84	100.38	7.85	100.44
1664	7.35	97.86	7.56	100.58	7.56	100.64	7.56	100.65
1728	7.19	99.45	7.29	100.83	7.29	100.82	7.30	100.93
1792	6.96	99.79	7.05	101.10	7.05	101.09	7.05	101.13
1856	6.74	100.05	6.82	101.33	6.82	101.32	6.83	101.37
1920	6.53	100.34	6.60	101.45	6.60	101.42	6.61	101.46
1984	6.34	100.69	6.39	101.45	6.40	101.52	6.39	101.50
2048	6.14	100.67	6.19	101.41	6.19	101.42	6.19	101.42
2112	5.94	100.38	6.00	101.35	6.00	101.35	6.00	101.37
2176	5.75	100.11	5.83	101.50	5.83	101.49	5.83	101.52
2240	5.58	99.96	5.63	100.84	5.63	100.85	5.63	100.83
2304	5.40	99.58	5.42	99.92	5.42	99.92	5.42	99.93
2368	5.24	99.23	5.24	99.26	5.24	99.24	5.24	99.25
2432	5.08	98.81	5.05	98.34	5.05	98.33	5.05	98.34
2496	4.95	98.85	4.91	97.95	4.90	97.91	4.91	97.99

2560	4.82	98.68	4.79	98.06	4.79	98.04	4.79	98.06
2624	4.71	98.90	4.70	98.71	4.70	98.70	4.70	98.70
2688	4.61	99.03	4.62	99.26	4.62	99.26	4.62	99.25
2752	4.52	99.47	4.55	100.23	4.55	100.21	4.55	100.25
2816	4.43	99.87	4.53	101.97	4.53	101.97	4.53	102.01
2880	4.36	100.40	4.43	102.11	4.43	102.12	4.43	102.14
2944	4.28	100.73	4.34	102.25	4.34	102.24	4.34	102.26
3008	4.20	101.11	4.25	102.37	4.25	102.37	4.25	102.37
3072	4.12	101.33	4.17	102.49	4.17	102.49	4.17	102.52
3136	4.05	101.58	4.09	102.61	4.09	102.62	4.09	102.63
3200	3.97	101.67	4.01	102.73	4.01	102.73	4.01	102.75
3264	3.90	101.73	3.94	102.82	3.94	102.82	3.94	102.84
3328	3.82	101.82	3.87	102.92	3.87	102.91	3.87	102.94
3392	3.75	101.88	3.80	103.02	3.80	103.01	3.80	103.03
3456	3.68	101.84	3.73	103.12	3.73	103.10	3.73	103.13
3520	3.62	101.81	3.67	103.22	3.67	103.22	3.67	103.24
3584	3.55	101.86	3.60	103.32	3.60	103.30	3.60	103.34
3648	3.49	101.90	3.54	103.39	3.54	103.40	3.54	103.41
3712	3.43	101.75	3.48	103.48	3.48	103.49	3.49	103.49
3776	3.37	101.67	3.43	103.55	3.43	103.55	3.43	103.57
3840	3.31	101.76	3.37	103.61	3.37	103.62	3.37	103.65
3904	3.26	101.73	3.32	103.69	3.32	103.68	3.32	103.70
3968	3.20	101.73	3.27	103.76	3.27	103.76	3.27	103.77
4032	3.14	101.20	3.22	103.83	3.22	103.82	3.22	103.84
4096	3.09	101.32	3.17	103.90	3.17	103.90	3.17	103.92

Table 6: DPDK Driver – QDMA4.0 VF Forwarding performance test results

Latency Measurements

The provided Reference Design and Bitfile can be used to measure latency in any system when traffic is ongoing. When it is enabled, C2H data payload will be replaced with a known counter value (as a timestamp) and will be measured on the H2C side once the testpmd application has looped the data back. The difference in value between the data payload received at the H2C side and the current counter value will be the sum of C2H and H2C latency.

Latency measurement can be done by following these steps:

- Set the number of clock cycles within each **Measurement window** (see register offset below). The counters will gather data within this time window and take a snapshot of the result for users to read. Default value is 1s (0xEE6B280).
    - **Note: The u**ser must make sure to wait long enough for the measurement window to fill up completely after reset or in between readings before reading the next counter values, otherwise zero or the previous value will be returned.
    - All eight (8) counters must be read at least once, or reset through the Control register, before a new reading will be presented

- Set the mode bit [1] in **Control** (see register offset below) to 1 to allow continuous packet measurement. A value of 0 is currently not supported (reserved).

- Set the reset bit [0] in **Control** (see register offset below) to 1 and then 0 to reset the counters and start measurement.

The module will have four different measurement counters:

- **Max_latency**: Max latency number measured within the measurement window.
- **Min_latency:** Min latency number measured within the measurement window.
- **Sum_latency:** Sum of all latency numbers measured within the measurement window.
- **Pkt_rcvd:** Number of packets received within the measurement window.

Note: Average latency can be measured by taking the sum_latency divided by pkt_rcvd.

Latency Counters Register Offset:

- **0x104:** Measurement window [63:32]
- **0x100:** Measurement window [31:0]
- **0x108:** Control
- **0x110:** Max_latency [63:32]
- **0x10C:** Max_latency [31:0]
- **0x118:** Min_latency [63:32]
- **0x114:** Min_latency [31:0]
- **0x120:** Sum_latency [63:32]
- **0x11C:** Sum_latency [31:0]
- **0x128:** Pkt_rcvd [63:32]
- **0x124:** Pkt_rcvd [31:0]

Linux Kernel Reference Driver

The data below is collected with indirect interrupt (i.e., interrupt aggregation) mode.

Streaming Mode Performance

The dma-perf config files used for the below streaming mode tests are part of the linux reference kernel driver source, hosted at GitHub https://github.com/Xilinx/dma_ip_drivers, under the directory QDMA/linux-kernel/apps/dma-perf/dmaperf_config:

- C2H unidirectional: st-c2h-pfetch1.zip
- H2C unidirectional: st-h2c.zip
- C2H & H2C bi-directional: st-bi.zip

PF Performance

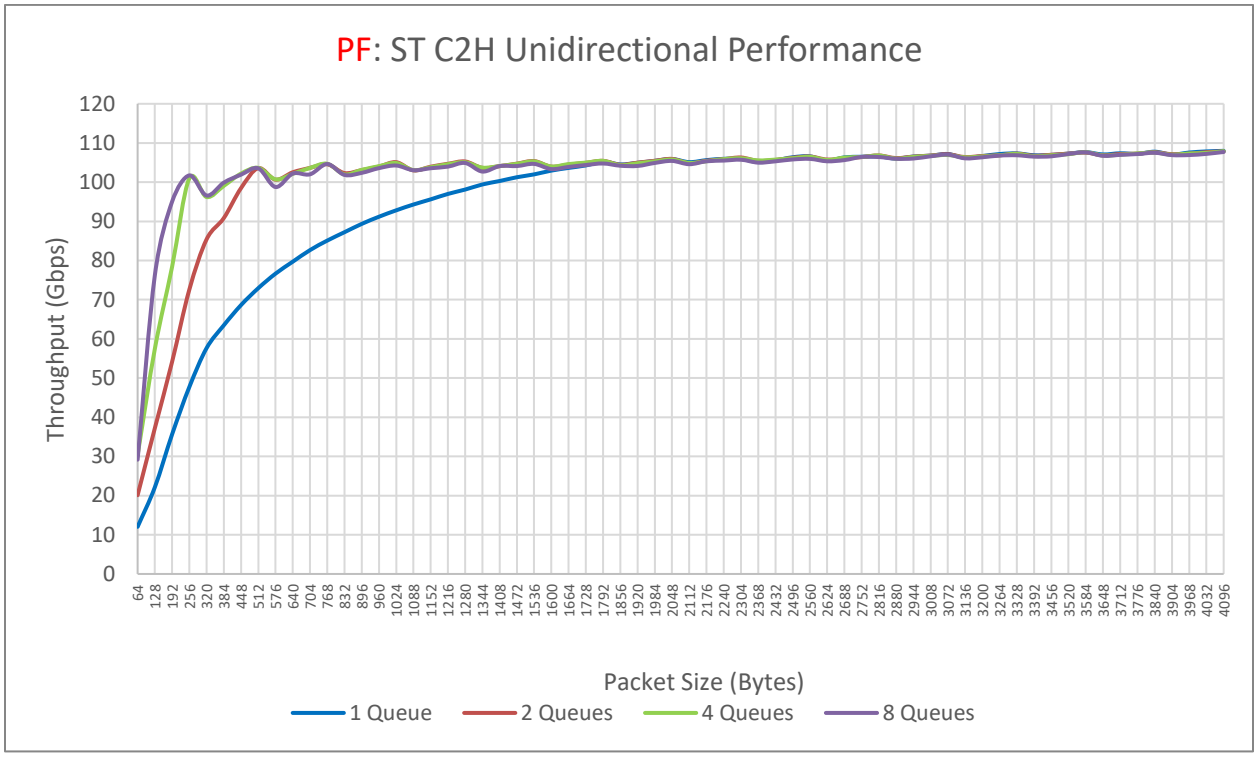


Figure 15: Linux Kernel Reference Driver – QDMA4.0 ST C2H Unidirectional Performance

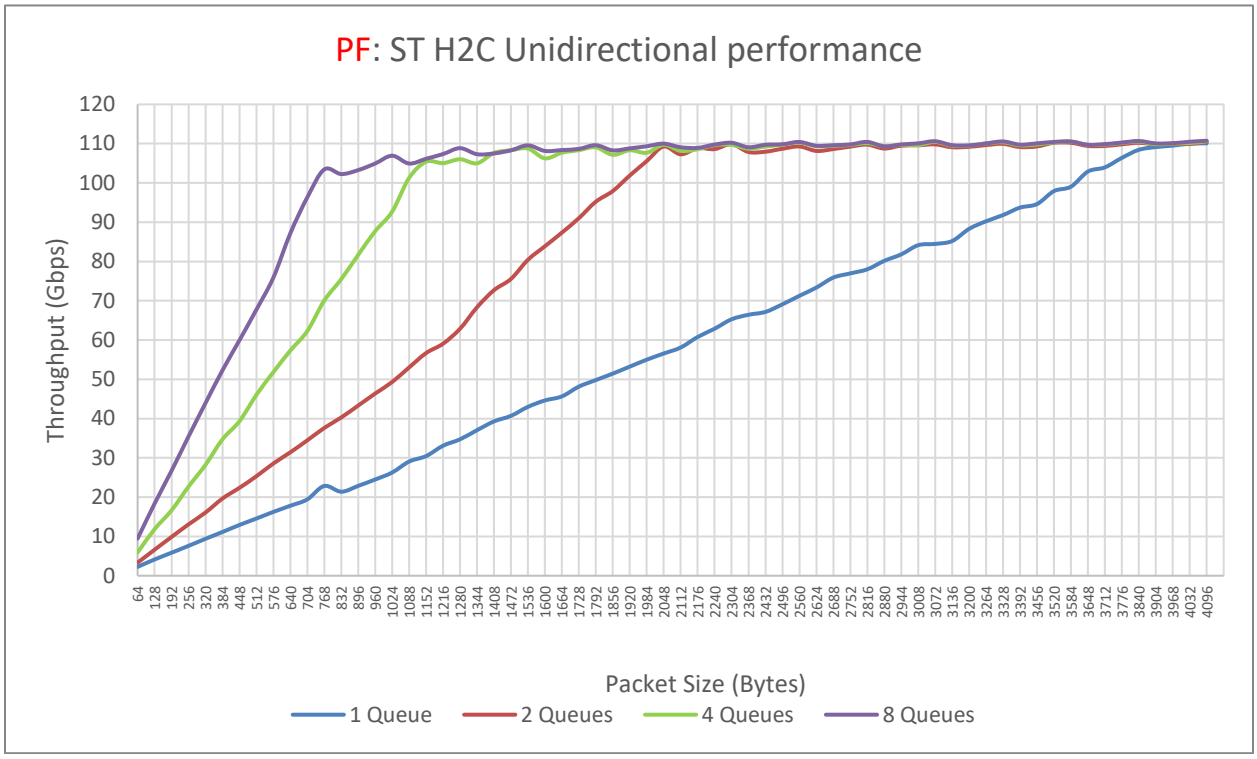


Figure 16 Linux Kernel Reference Driver – QDMA4.0 ST H2C Unidirectional performance

The above H2C graph shows that the QDMA IP can achieve line rate at small packet size (with 8 queues). When a smaller number of queues are involved, the results are not optimal because there are not enough I/O requests in flight to fill the pipeline, especially in the single queue scenario. The “dma-perf” tool and the driver are still being optimized for these cases.

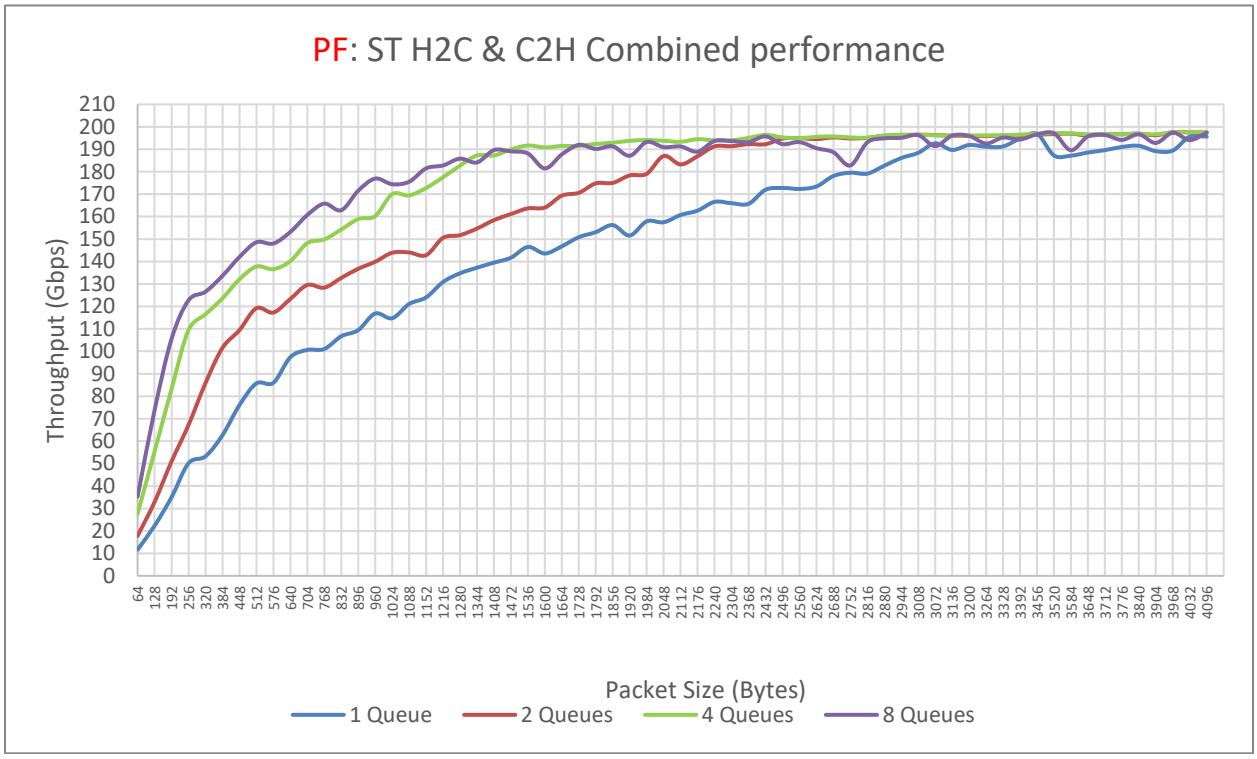


Figure 17: Linux Kernel Reference Driver - QDMA4.0 ST combined performance with Bidirectional traffic

Bi-directional ST performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

VF Performance

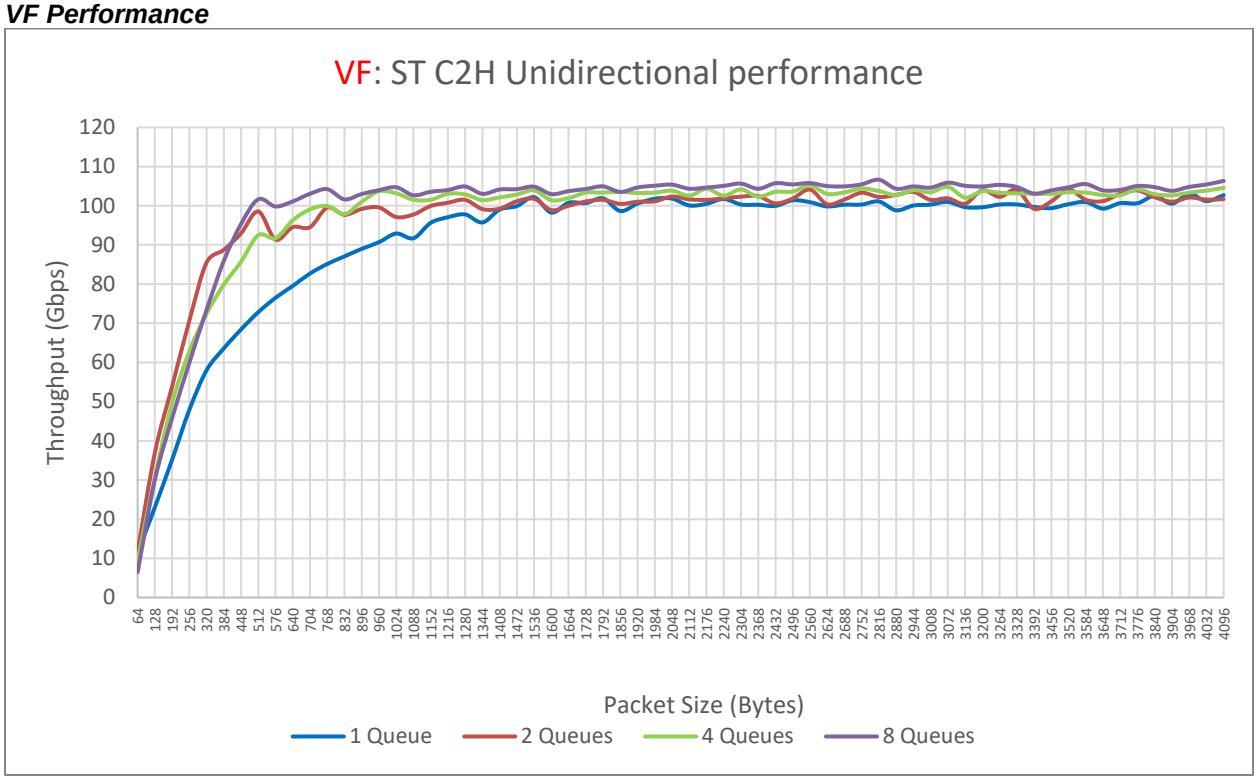


Figure 18: Linux Kernel Reference Driver – QDMA4.0 ST VF C2H Unidirectional Performance

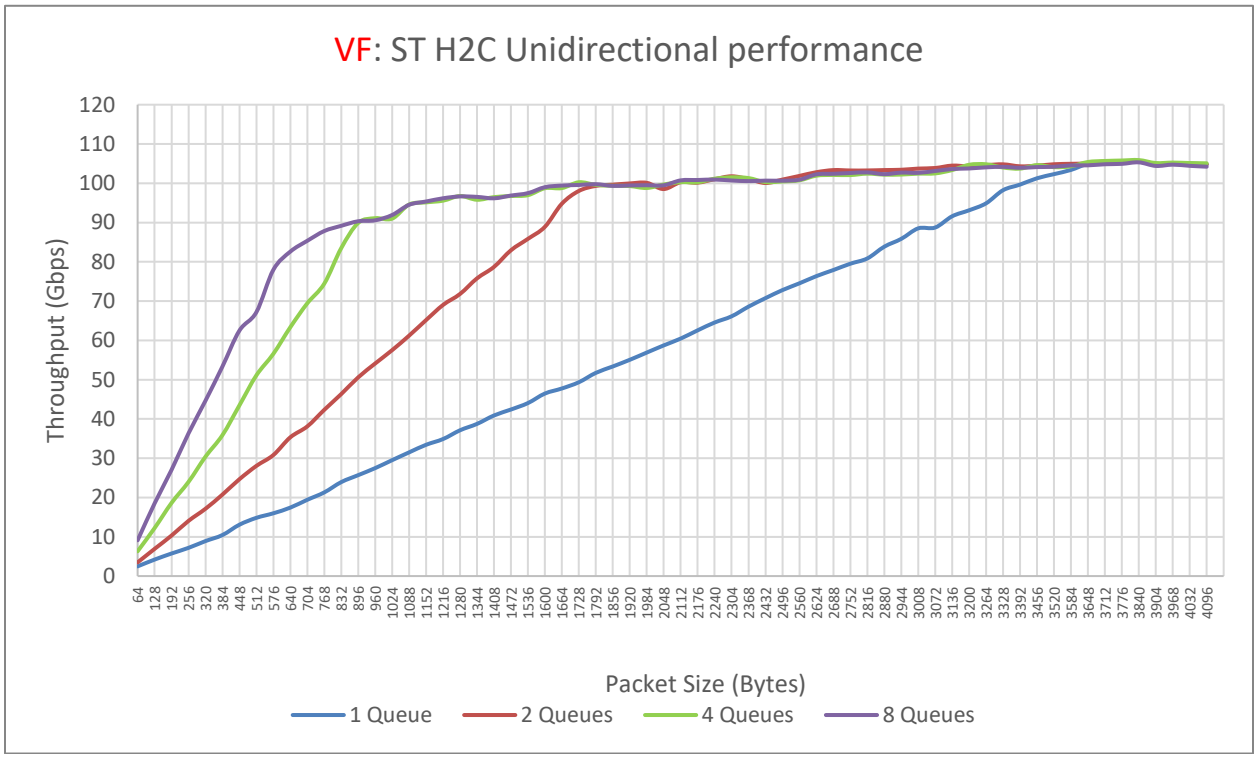


Figure 19: Linux Kernel Reference Driver – QDMA4.0 ST VF H2C Unidirectional Performance

The above H2C graph shows that the QDMA IP can achieve line rate at small packet size (with 8 queues). When a smaller number of queues are involved the results are not optimal because there are not enough I/O requests in flight to fill the pipeline, especially in the single queue scenario. The “dma-perf” tool and the driver are still being optimized for these cases.

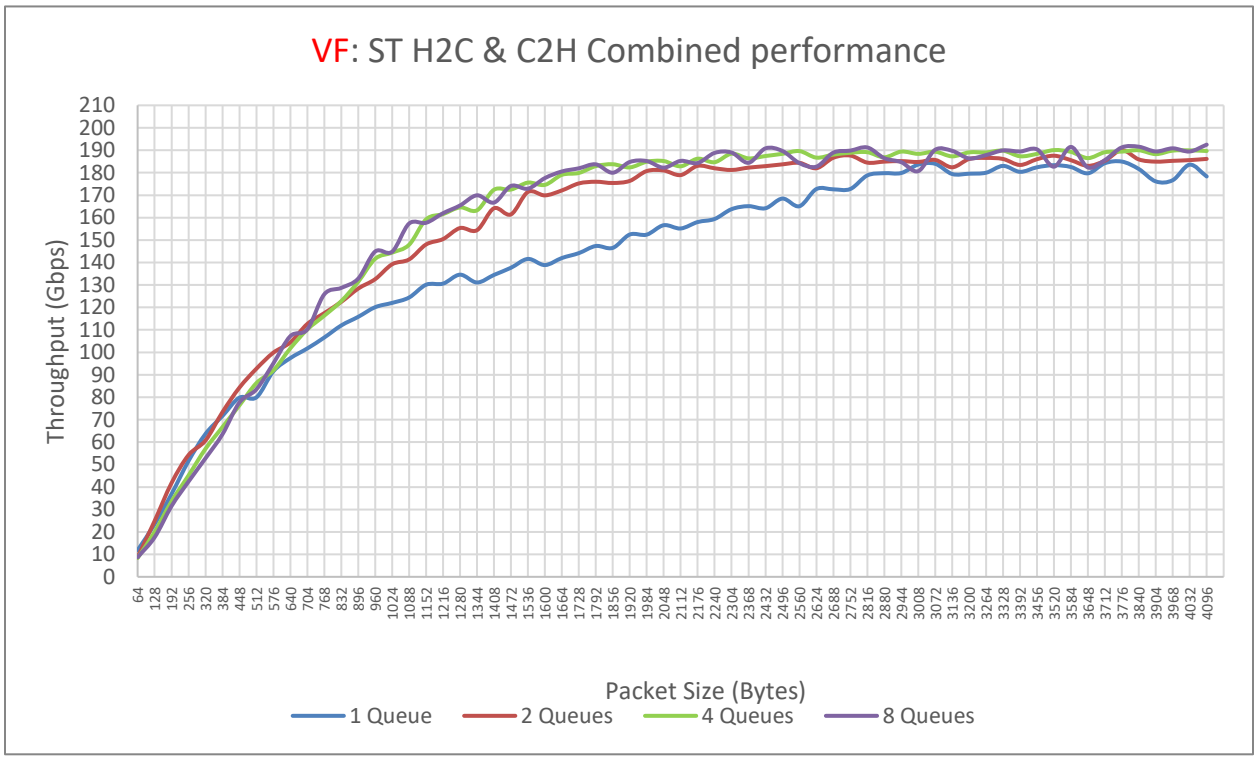


Figure 20: Linux Kernel Reference Driver – QDMA4.0 ST VF combined performance with Bidirectional traffic

Bi-directional ST performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

Memory Mapped Mode Performance BRAM Design

The data below is collected with BRAM. If using DDR memory, the memory controller overhead needs to be taken into consideration.

The dma-perf config files used for the above memory-map mode tests are part of the Linux reference kernel driver source, hosted at GitHub https://github.com/Xilinx/dma_ip_drivers, under directory QDMA/linux-kernel/apps/dma-perf/dmaperf_config:

- C2H unidirectional: mm-c2h.zip
- H2C unidirectional: mm-h2c.zip
- C2H & H2C bi-directional: mm-bi.zip

PF Performance

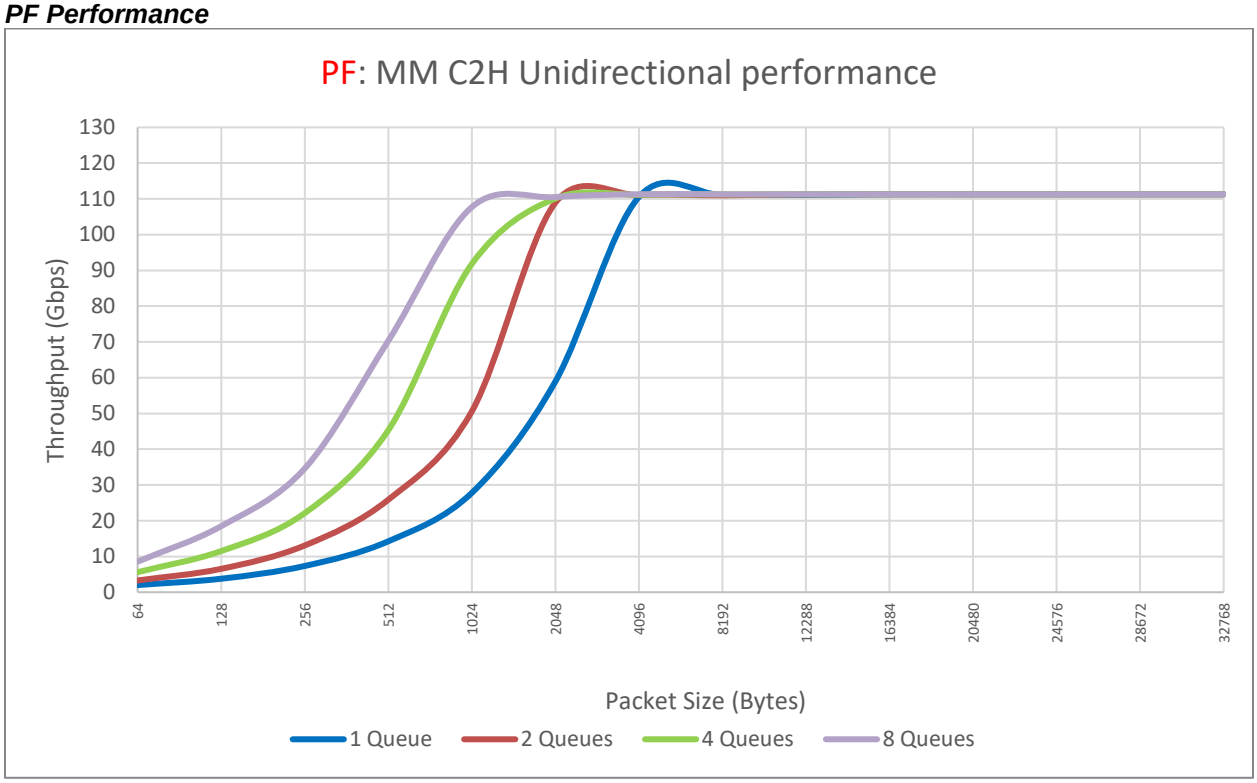


Figure 21: Linux Kernel Reference Driver –QDMA 4.0 BRAM design MM C2H unidirectional performance

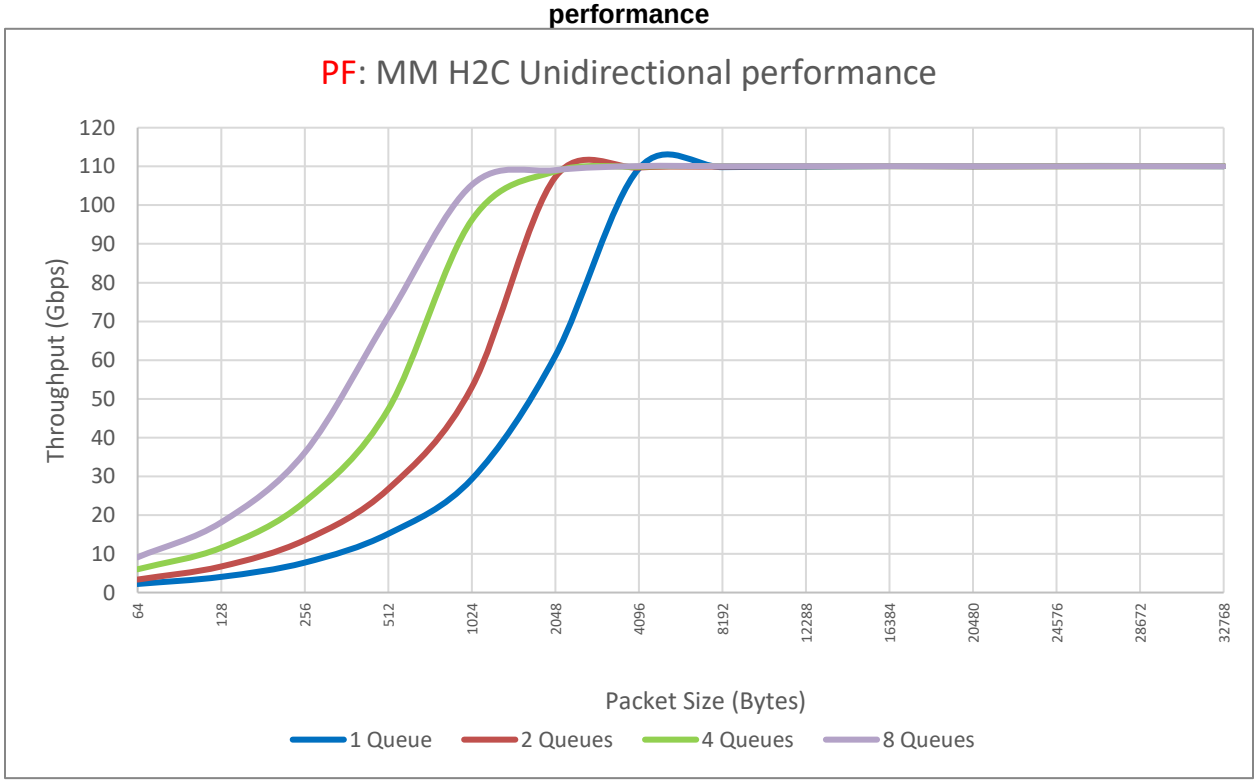


Figure 22: Linux Kernel Reference Driver – QDMA4.0 BRAM design MM H2C unidirectional performance

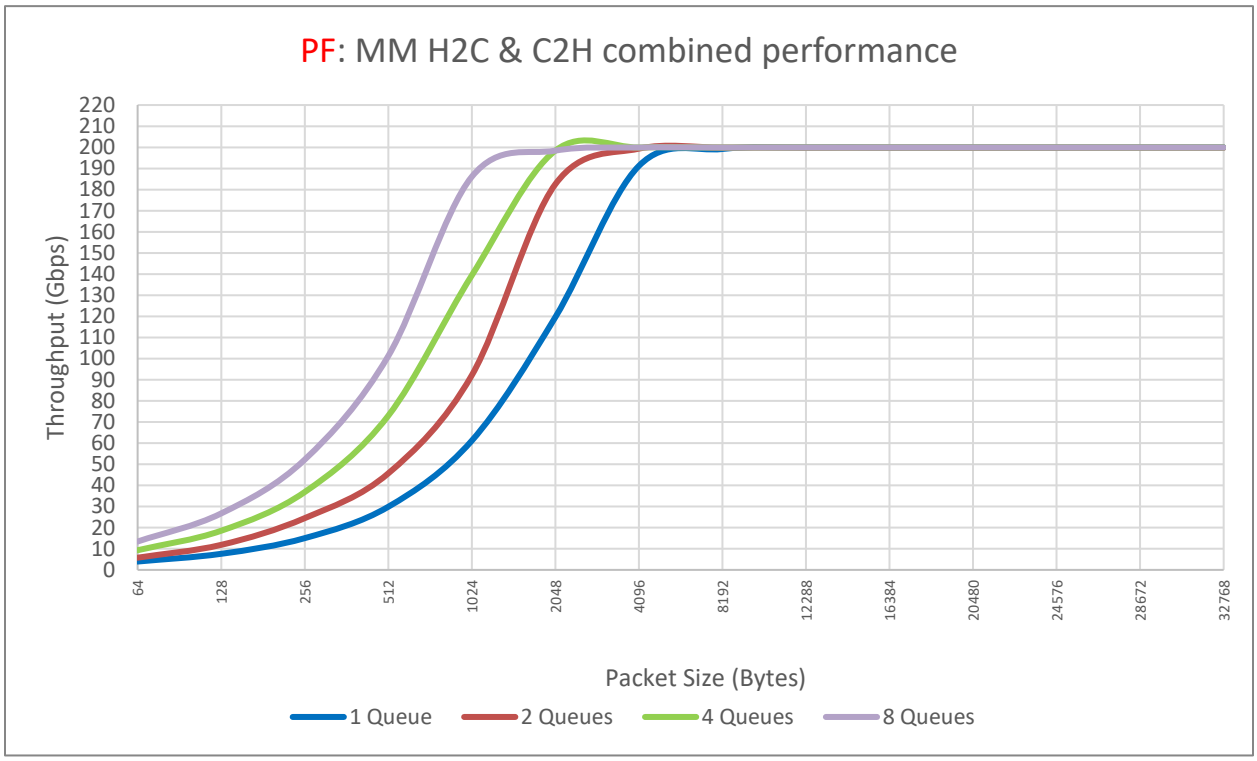


Figure 23: Linux Kernel Reference Driver – QDMA 4.0 BRAM design MM combined bidirectional performance

Bi-directional MM performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

VF Performance

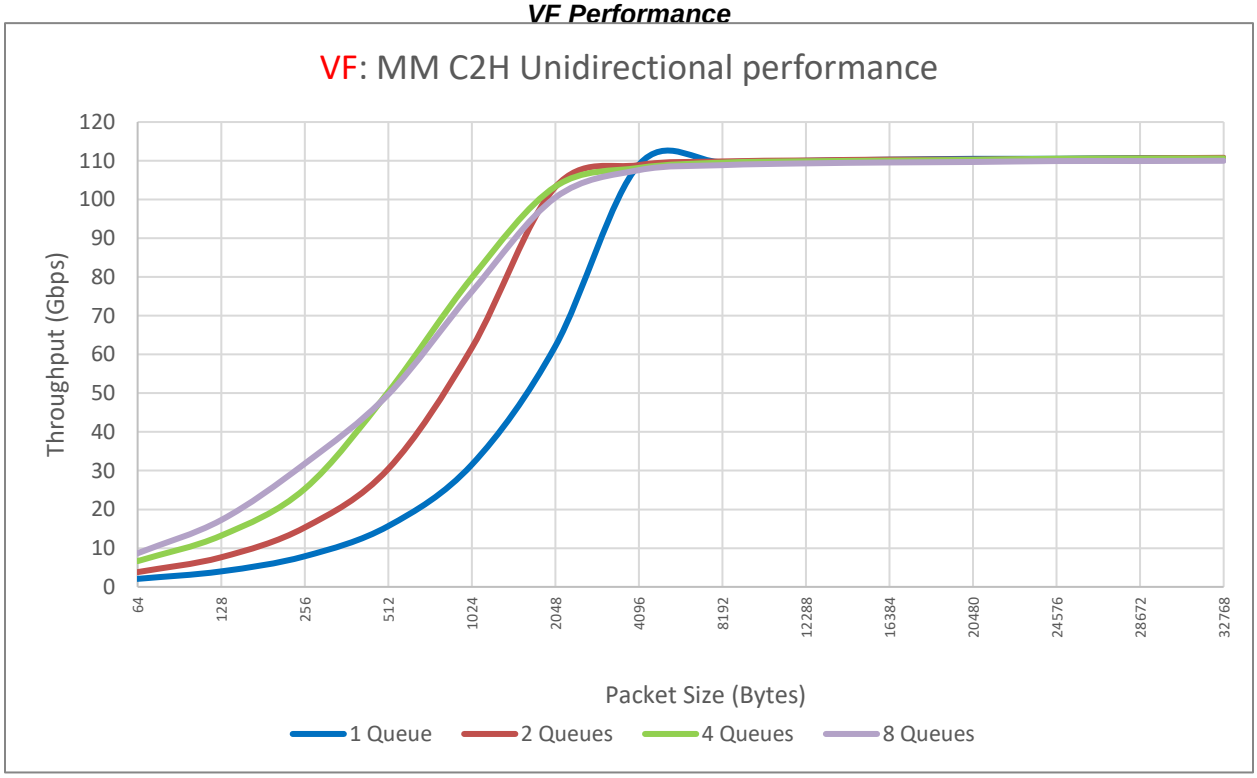


Figure 24: Linux Kernel Reference Driver –QDMA4.0 VF BRAM design MM C2H unidirectional performance

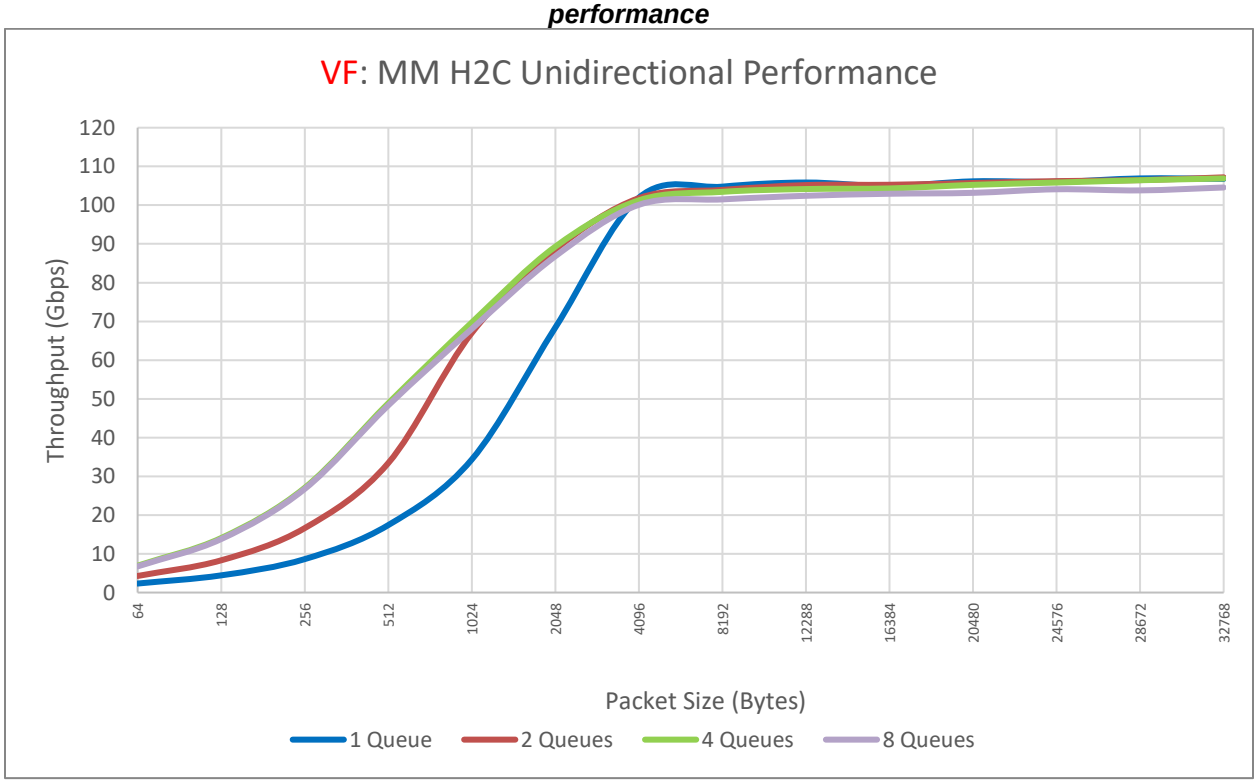


Figure 25: Linux Kernel Reference Driver – QDMA4.0 VF BRAM design VF MM H2C unidirectional performance

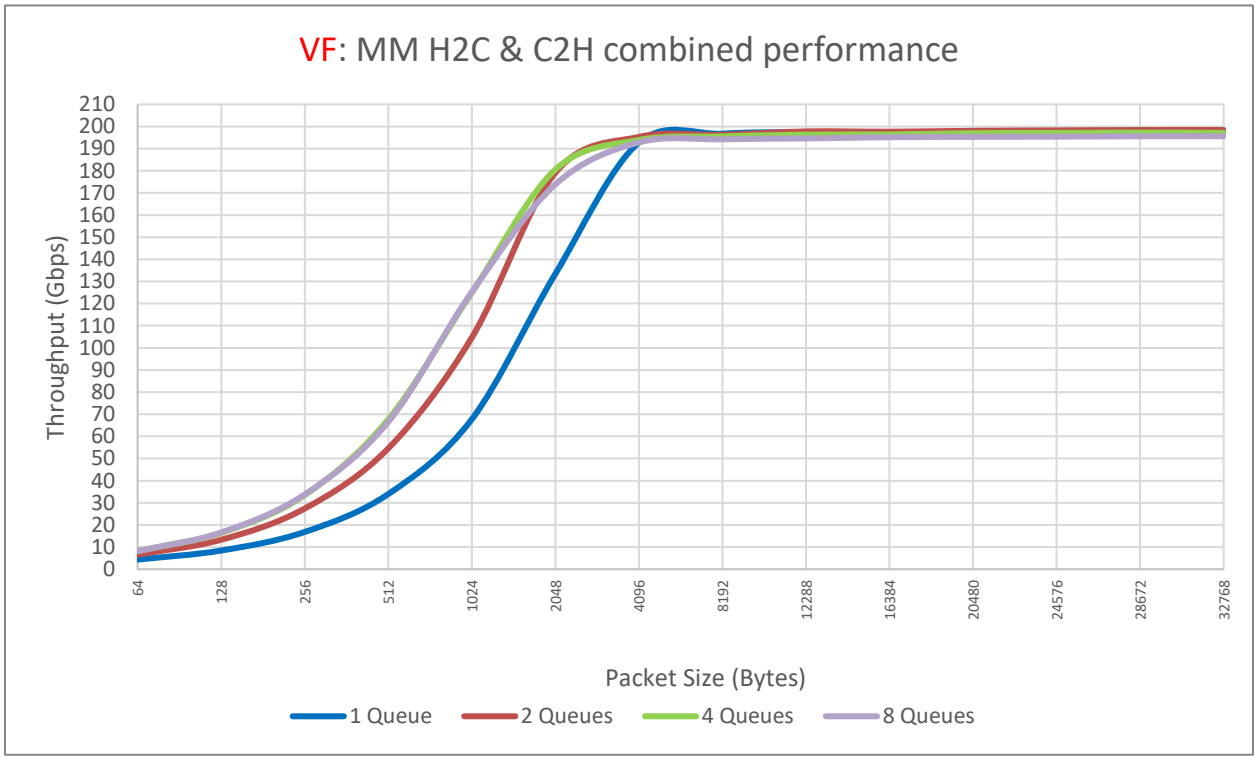


Figure 26: Linux Kernel Reference Driver – QDMA 4.0 VF BRAM design MM combined bidirectional performance

Bi-directional MM performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

Memory Mapped Mode Performance With DDR Design

The data below is collected with DDR.

The dma-perf config files used for the above memory-map mode tests are part of the Linux reference kernel driver source, hosted at GitHub https://github.com/Xilinx/dma_ip_drivers, under the directory QDMA/linux-kernel/apps/dma-perf/dmaperf_config:

- C2H unidirectional: mm-c2h.zip
- H2C unidirectional: mm-h2c.zip
- C2H & H2C bi-directional: mm-bi.zip

PF Performance

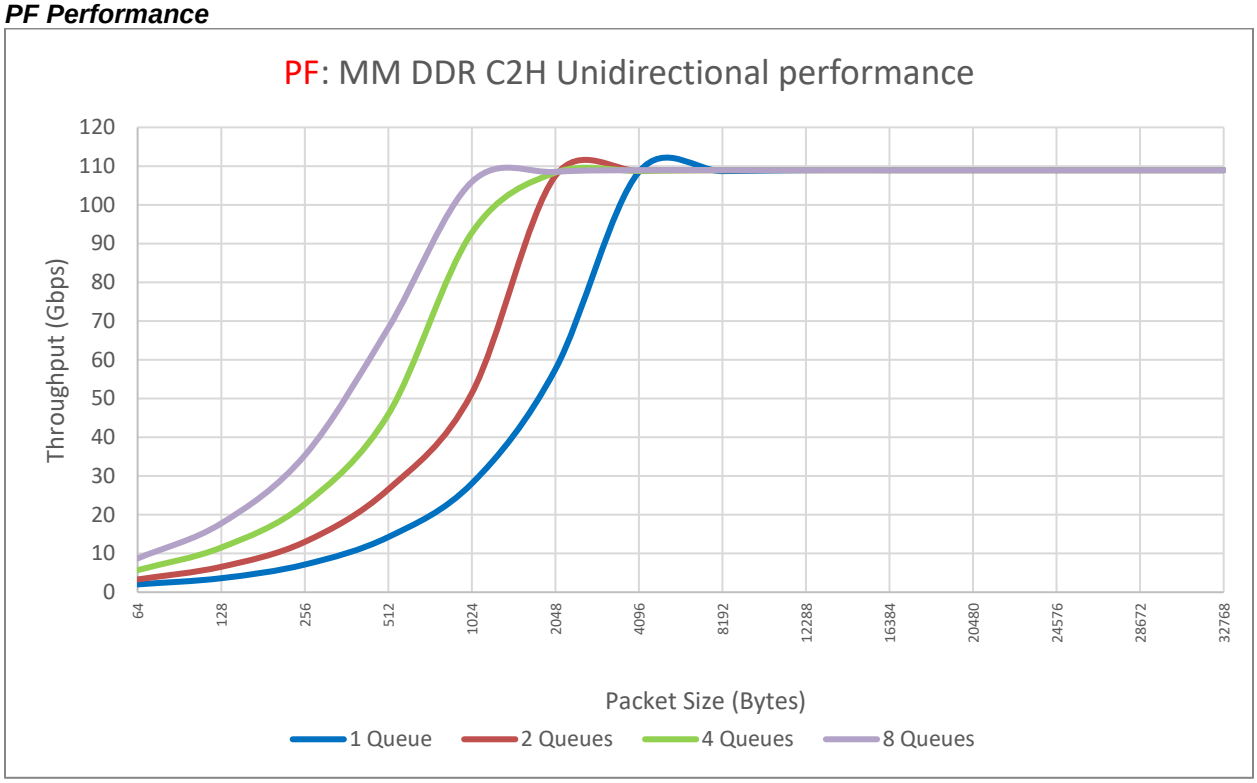


Figure 27: Linux Kernel Reference Driver – QDMA4.0 DDR Design MM C2H unidirectional performance

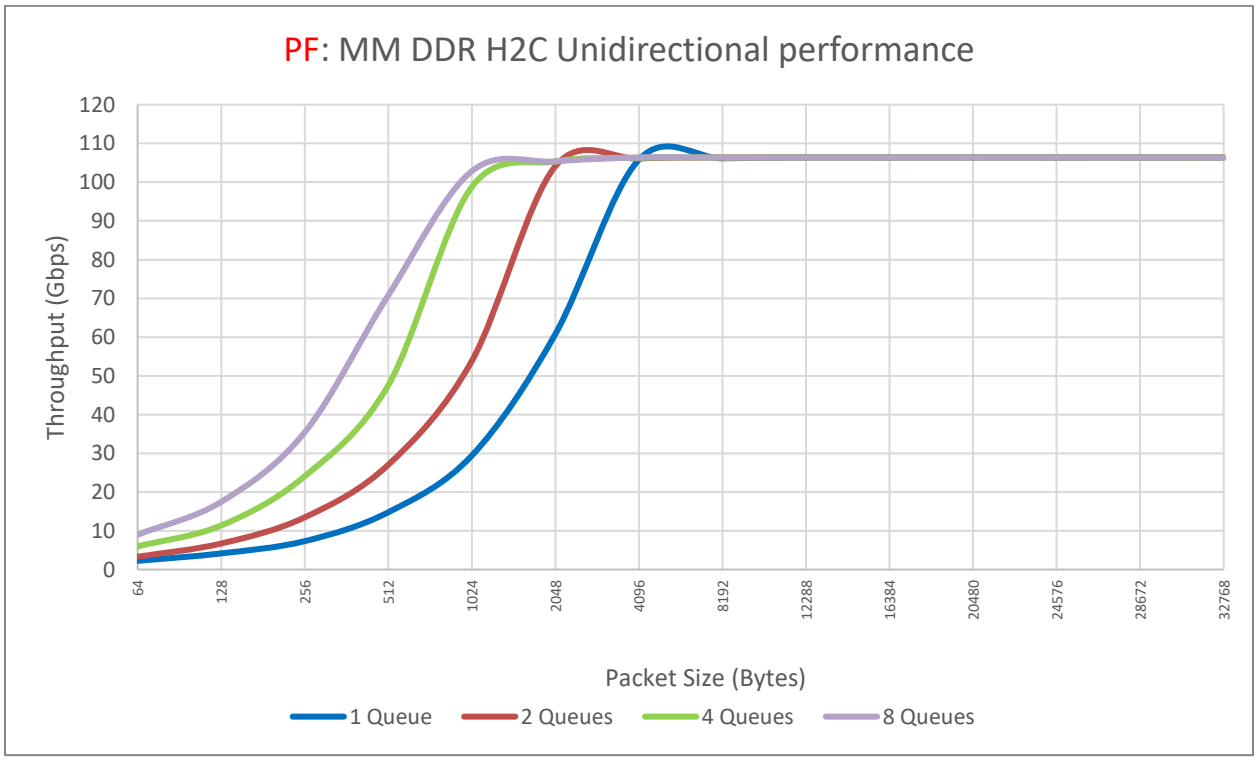


Figure 28: Linux Kernel Reference Driver – QDMA4.0 DDR design MM H2C unidirectional performance

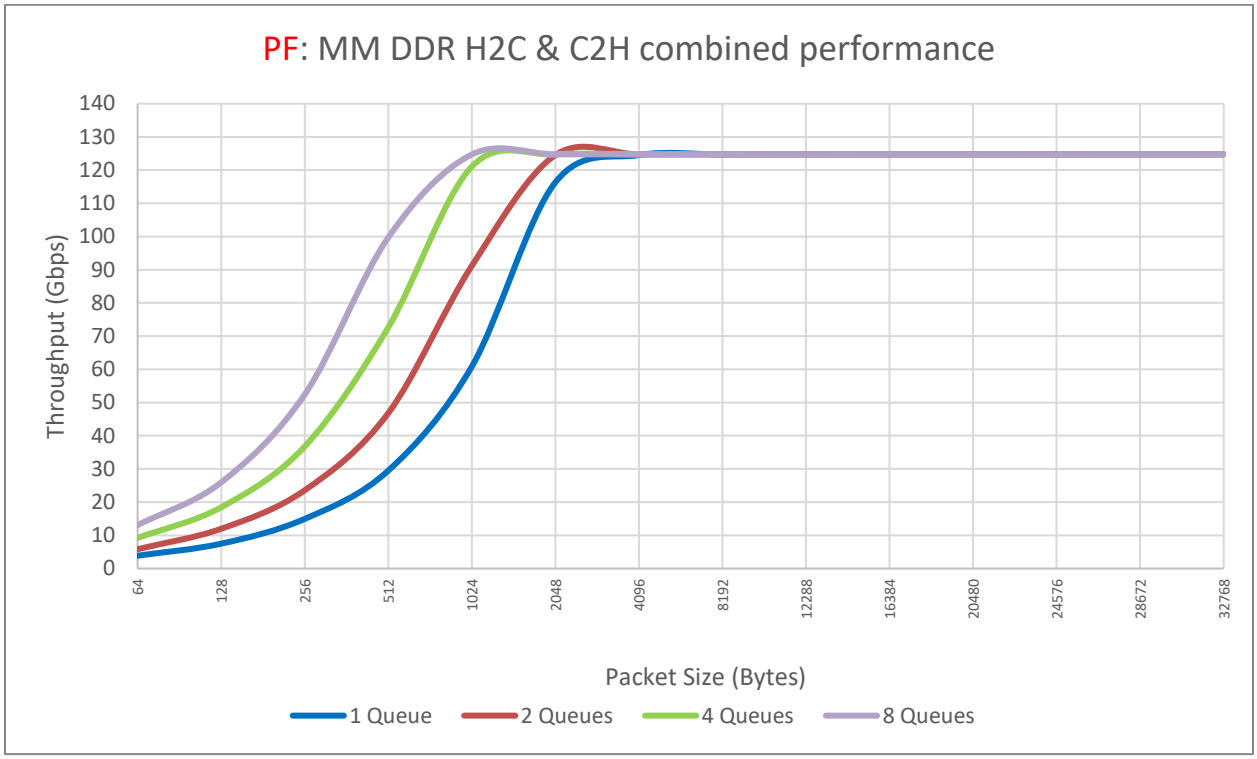


Figure 29: Linux Kernel Reference Driver – QDMA4.0 DDR design MM combined bidirectional performance

Bi-directional MM performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

VF Performance

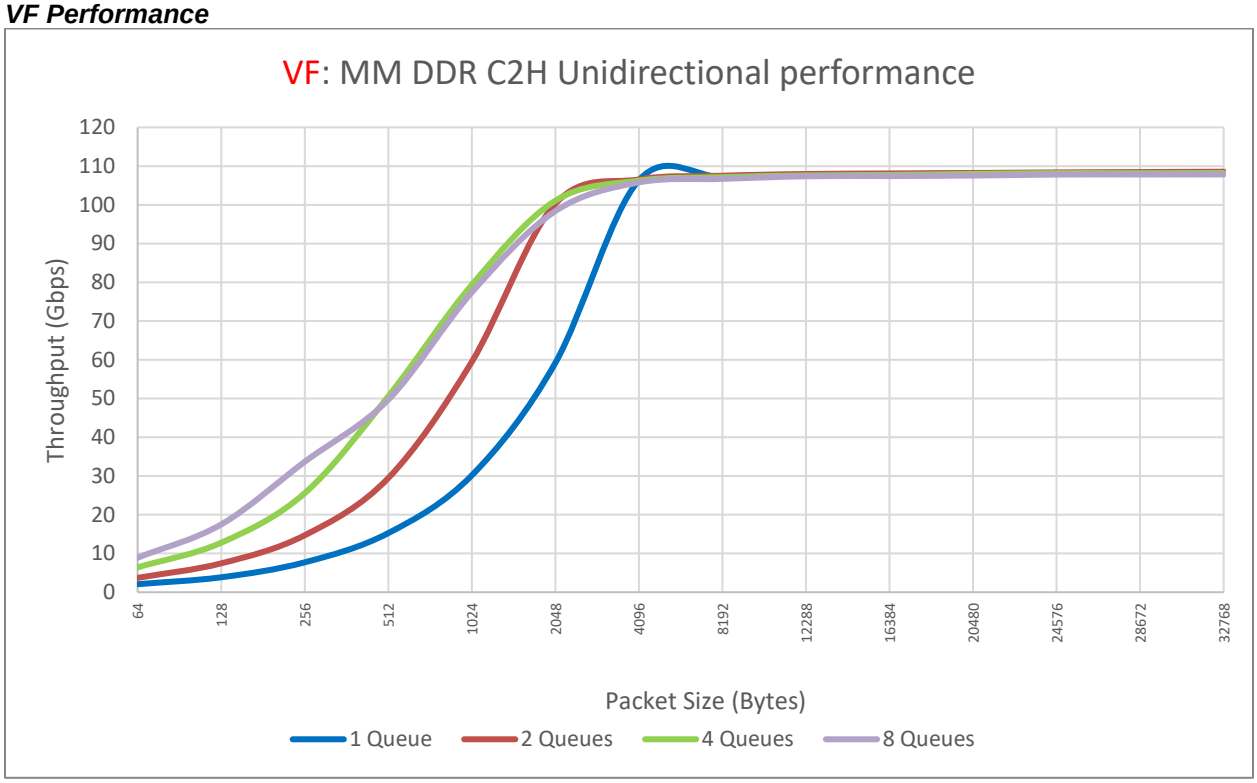


Figure 30: Linux Kernel Reference Driver – QDMA4.0 DDR Design MM C2H unidirectional performance

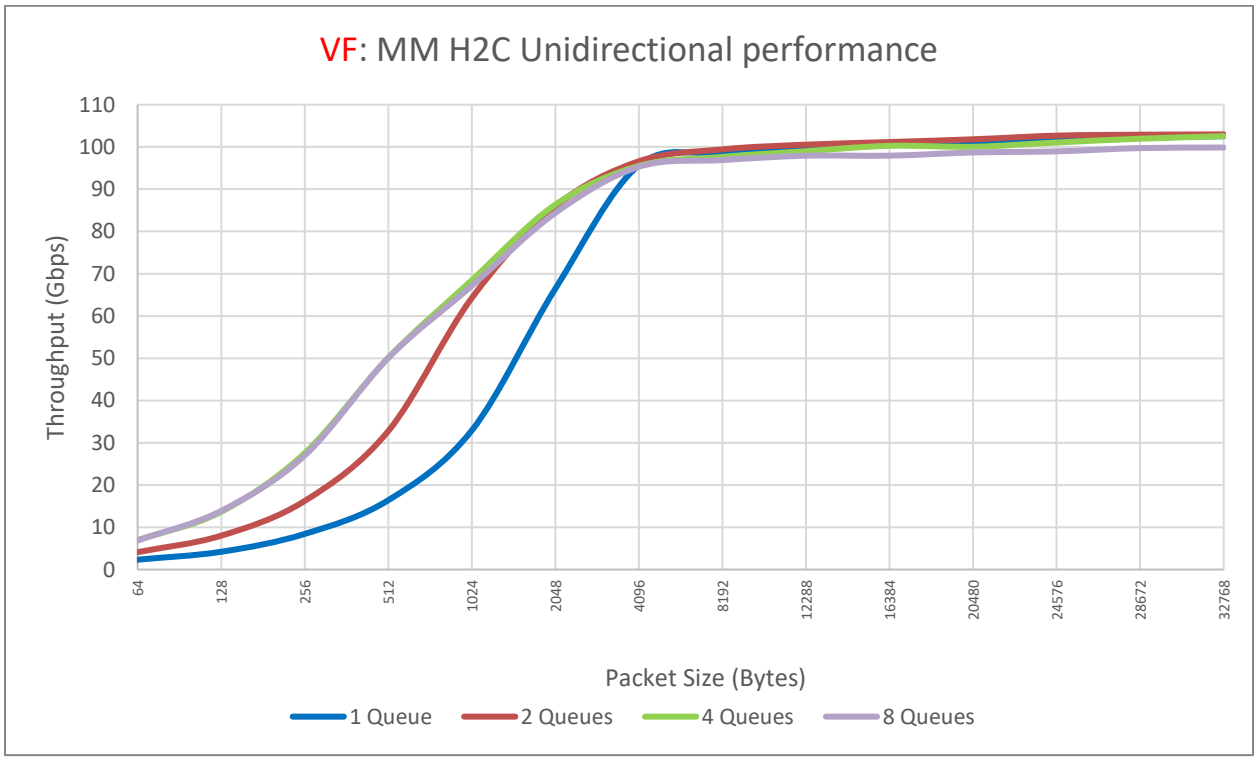


Figure 31: Linux Kernel Reference Driver – QDMA4.0 DDR design MM H2C unidirectional performance

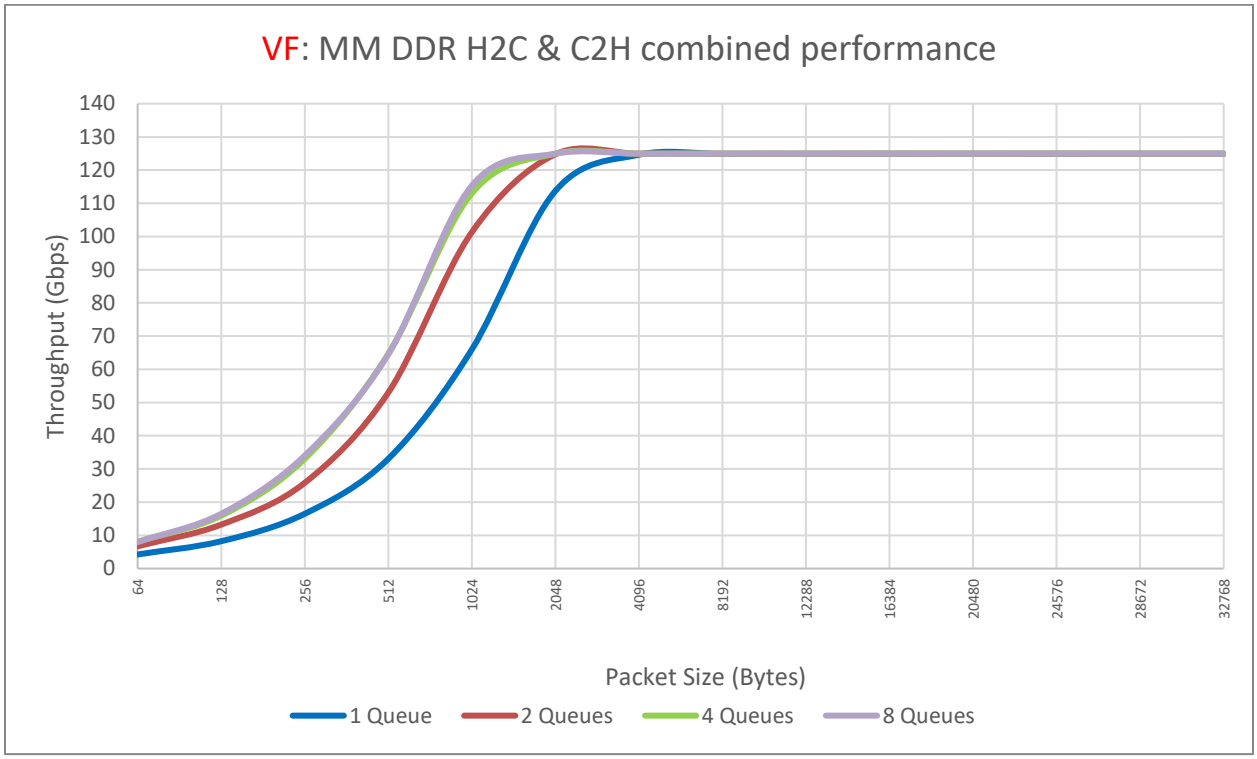


Figure 32: Linux Kernel Reference Driver – QDMA4.0 DDR design MM combined bidirectional performance

Bi-directional MM performance numbers are taken with traffic being enabled in both the H2C and C2H directions simultaneously.

Summary

The QDMA IP provides many capabilities that allow for very high throughput and efficiency. At the same time however, there are factors that impact performance, such as packet size, DMA overhead, system latency, and settings such as MPS, MRRS, etc.

This report provides enough data to choose the number of queues needed to achieve optimal performance depending on the application.

Typically, networking applications optimize for small packet performance and so can use more queues to saturate the Ethernet interface, while compute or storage applications might optimize for 4KB performance and saturate with fewer queues. As the report suggests, more queues help to achieve small packet performance, but the max number of queues cannot exceed the number of threads available for the application.

For the streaming mode this report suggests that 4 or more queues with prefetch enabled results in high performance for different packet sizes.

For the memory mapped mode, the QDMA IP easily achieves the line rate with the typical 4K workload even with a single queue when using BRAM. If DDR is desired, more queues might be needed to obtain the best performance. This would highly depend on the memory configuration and the access pattern. For example, concurrent read and write to the same memory bank would greatly reduce the efficiency and should be avoided if possible.

The bi-directional performance should be expected to be lower than uni-directional H2C and C2H, because the PCIe RQ interface is shared.

In a multi-socket machine where NUMA is enabled, the latency for DMA reads can be prohibitively high, causing lower performance. Caution must be taken in the driver to avoid using memory far away from the CPU core.

Based on knowledge of the application, it is possible to further reduce the DMA and TLP overheads to achieve better throughput than in this document.

References

These documents provide supplemental material useful with this performance report.

- QDMA Subsystem for PCI Express v4.0 – (PG302)
- dpdk-pktgen application
- UltraScale+ Devices Integrated Block for PCI Express v1.3

Revision History

The following table shows the revision history for this document.

Date	Version	Description
08-Nov-2020	1.0	QDMA4.0 2020.1 Performance Report