Hardware and software design of APEnetX: a custom FPGA-based NIC for scientific computing

The 27th International Conference on Computing in High-Energy & Nuclear Physics

Carlotta Chiarini

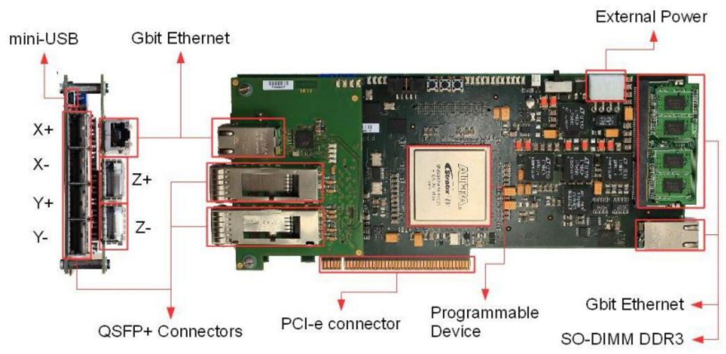
PhD student

On behalf of the APE group

INFN

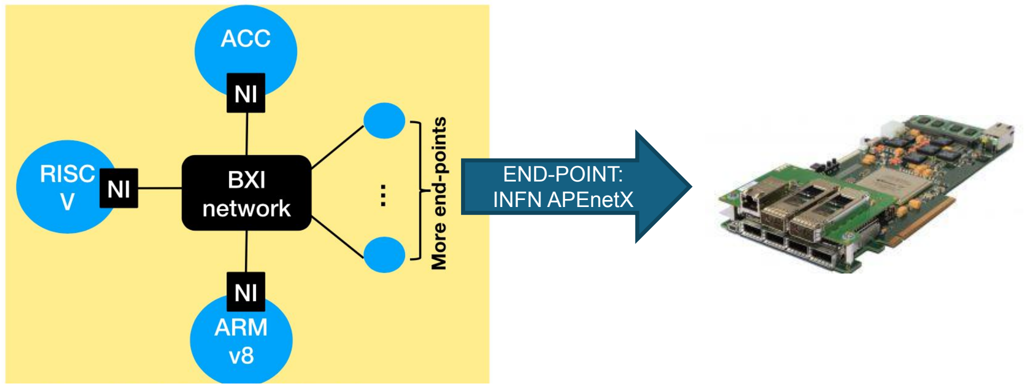
INFN APE project

Five generations of Network Interface Cards (NIC) for 3D-clusters nodes interconnection [2]

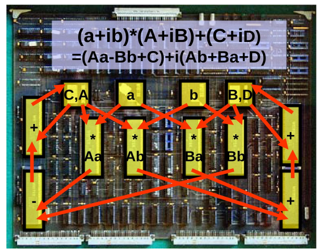


APEnet+ (2012)

…and now APEnetX!

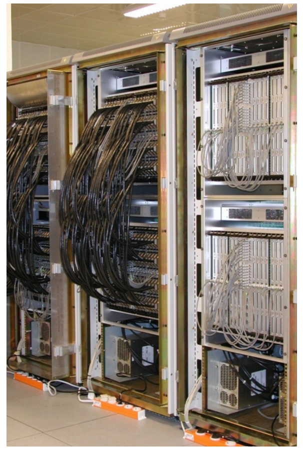


Four generations of massively parallel supercomputers based on custom processor [1]

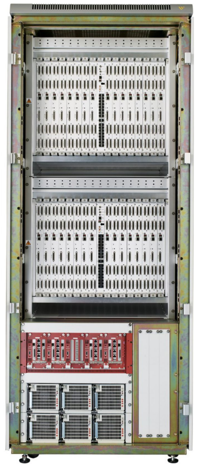


APE1
1 gigaFLOPS
(1986)

APE100
25 gigaFLOPS
(1994)



APE1000
128 gigaFLOPS
(1999)



apeNEXT
800 gigaFLOPS
(2004)

Focus of the presentation…

- Overview of the APEnetX SW/HW architecture
- Description of the integrated QoS mechanism to make the network resilient to congestion states

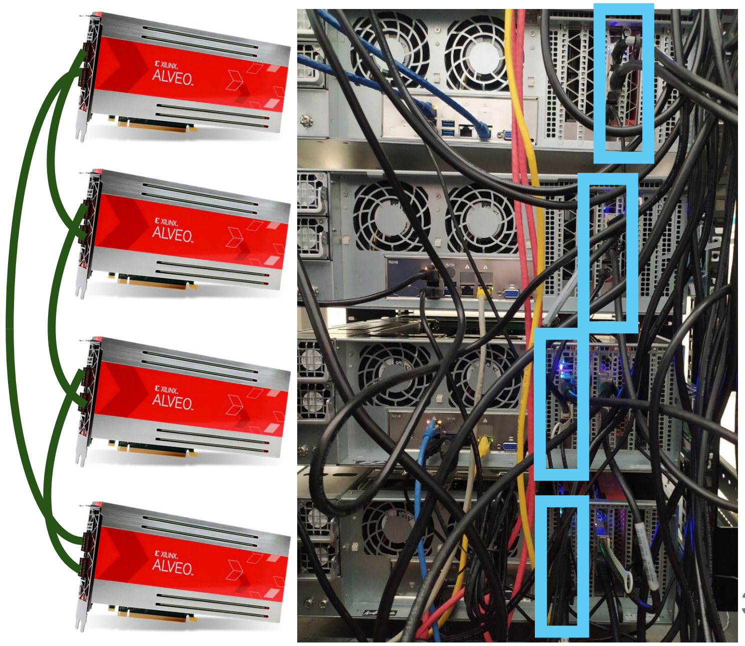
INFN APEnetX & Testbed

MAIN FEATURES

- NIC based on a PCIe x16 Gen3 interface
- Xilinx Alveo U200 built on the Xilinx 16 nm Ultrascale architecture
- 1D/2D/3D toroidal mesh topology for dead-lock free communications
- RDMA semantics supported + GPU direct
- Proprietary software driver and low-level communication library
- Custom APEnetX network simulator

TESTBED

- 4xServers
  - Sapphire Rapids family Intel CPUs @2.7GHz
    - AVX512 instructions
  - PCIe Gen3 support
- Each server hosts an APEnetX prototype
- Connected point-to-point with QSFP28 cables



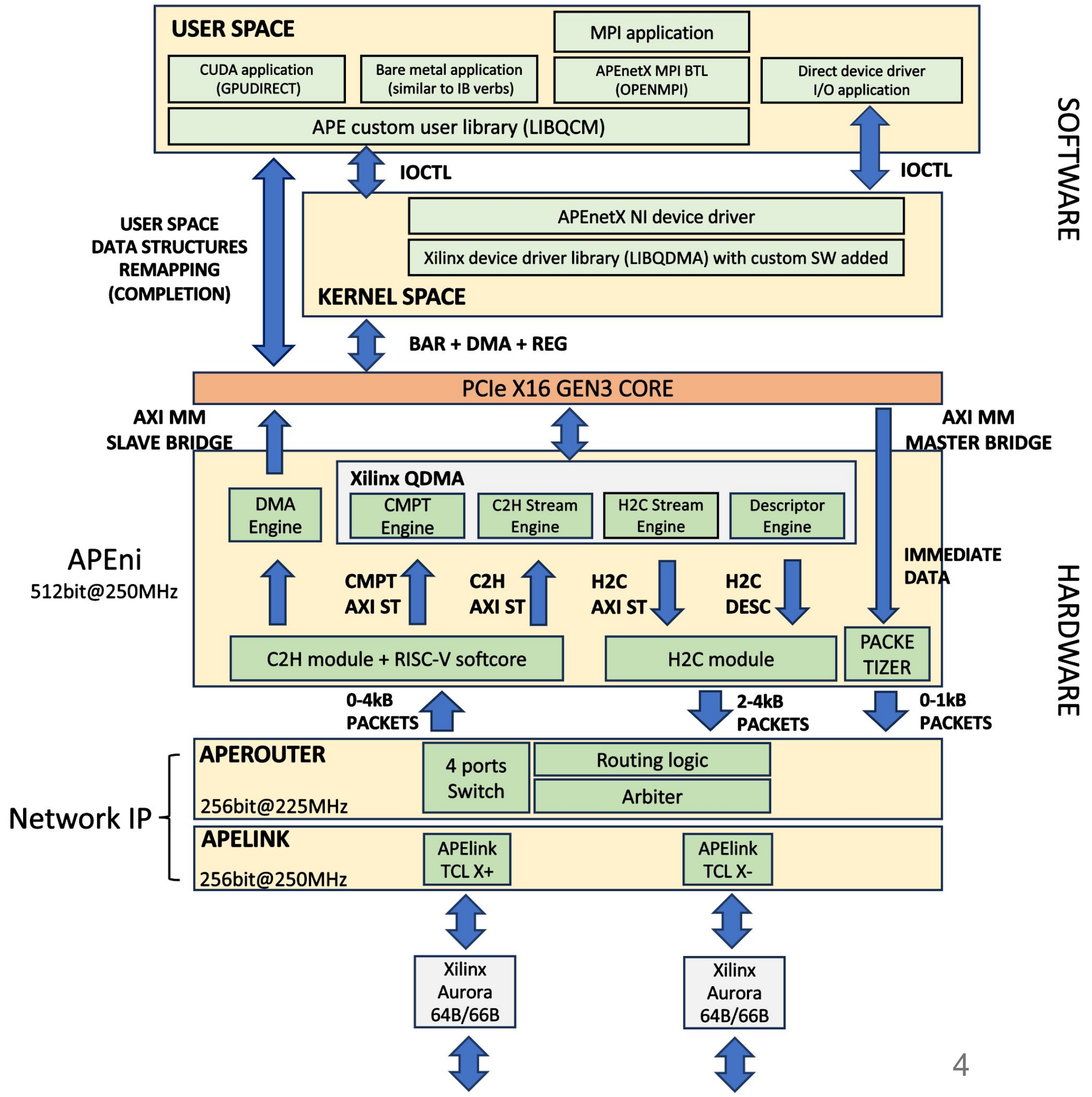
APEnetX architecture

SOFTWARE

- User space:
  - LIBQCM: APE custom user library
  - Synthetic tests and MPI applications (APEnetX MPI BTL)
- Kernel space:
  - Interaction with hardware and low-level operating system functions
  - Support for DMA read and write process

HARDWARE

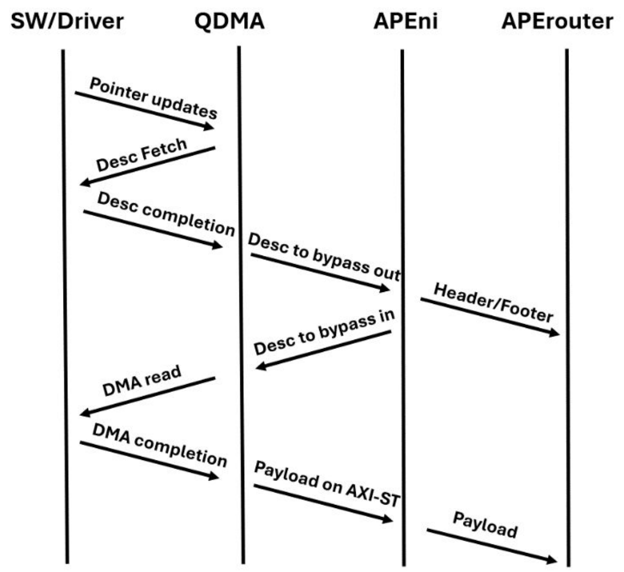
- APEni:
  - PCIe host interface: QDMA Xilinx IP + custom components
- Network IP:
  - APErouter: switching and routing functionalities
  - APElink: custom data link protocol over HSS links
- Transceivers of the target FPGA



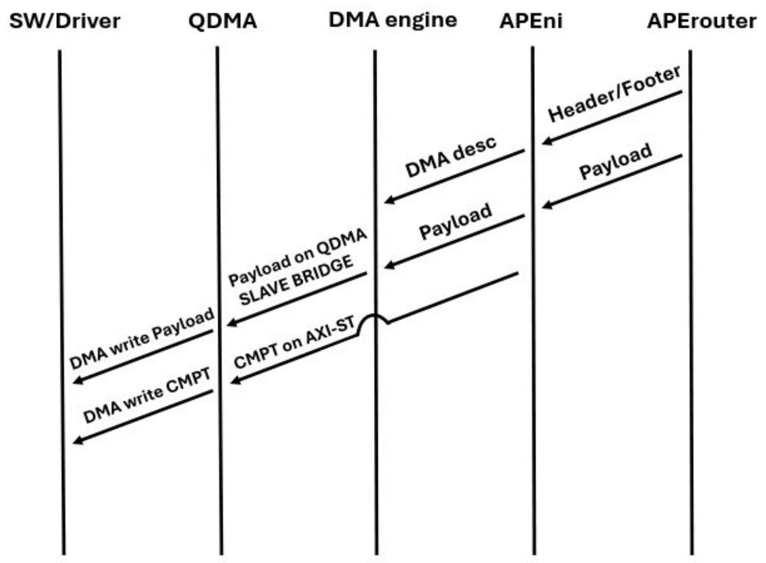
APEnetX HW architecture: APEni

- Leverages Xilinx QDMA IP to interface the PCIe Gen3 (x16) bus
  - Data transfer is initiated by means of “descriptors” and ends with “completions”
  - Up to 2048 queues (each with its own descriptors and completions rings)
- TX transactions are initiated by the host; the NIC intercepts descriptors to prepare network packets
- RX transactions are initiated directly from the NIC upon packet arrival; a completion notifies the host of newly arrived data

TX data flow (H2C)



RX data flow (C2H)

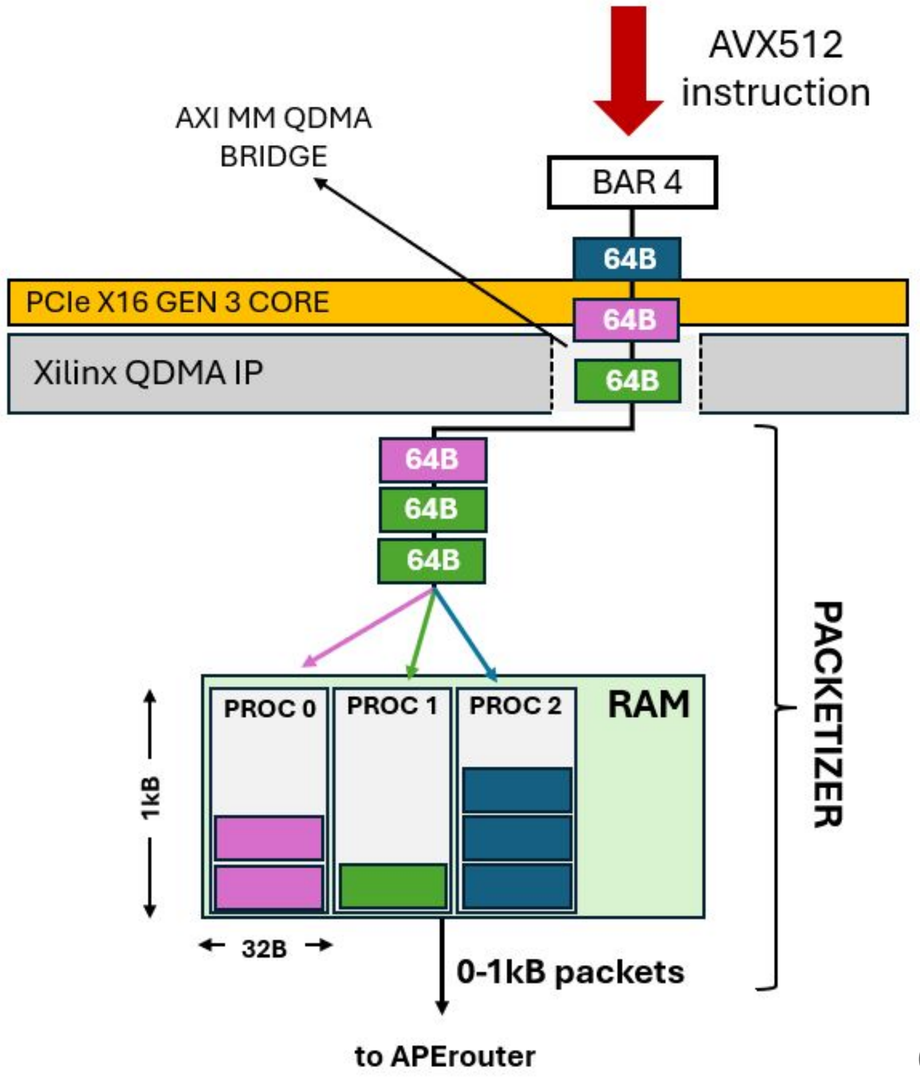


Optimization: Fast-send for small packets

- Bypass QDMA engine to achieve low latency for small packets (0-1kB packets)
  - Intel AVX512 instructions: atomic 64B data on BAR 4
  - QDMA AXI MM BRIDGE
  - Packetizer (HW component)

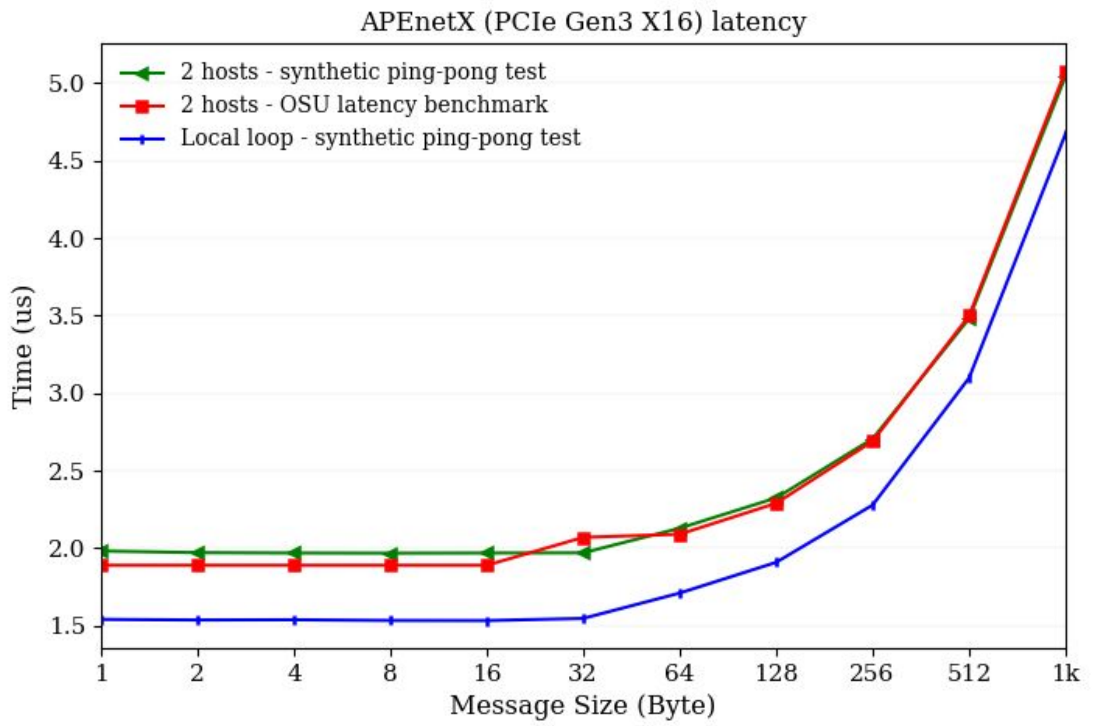
- Critical features:
  - A configurable number of processes is supported
  - Different processes can require access to the NIC concurrently

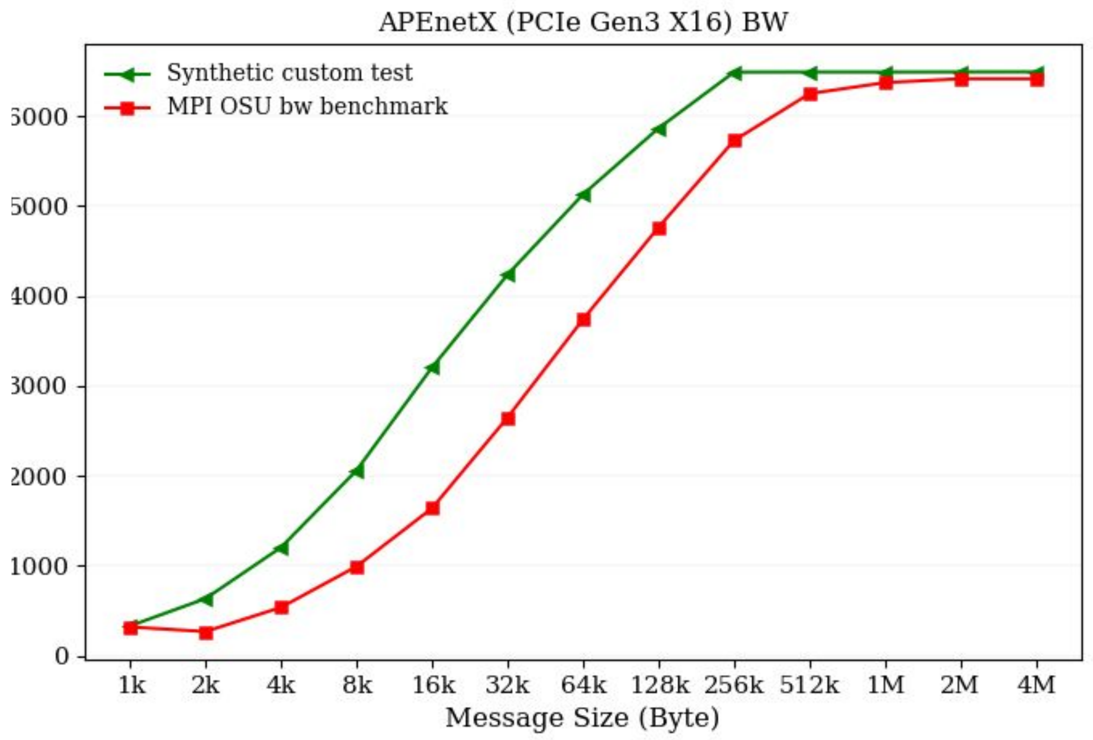
- Packetizer duties:
  - Store chunks of information in a local buffer based o the process ID
  - When the last chunk of a transaction arrives, the packet is forged



APEnetX CPU final performance

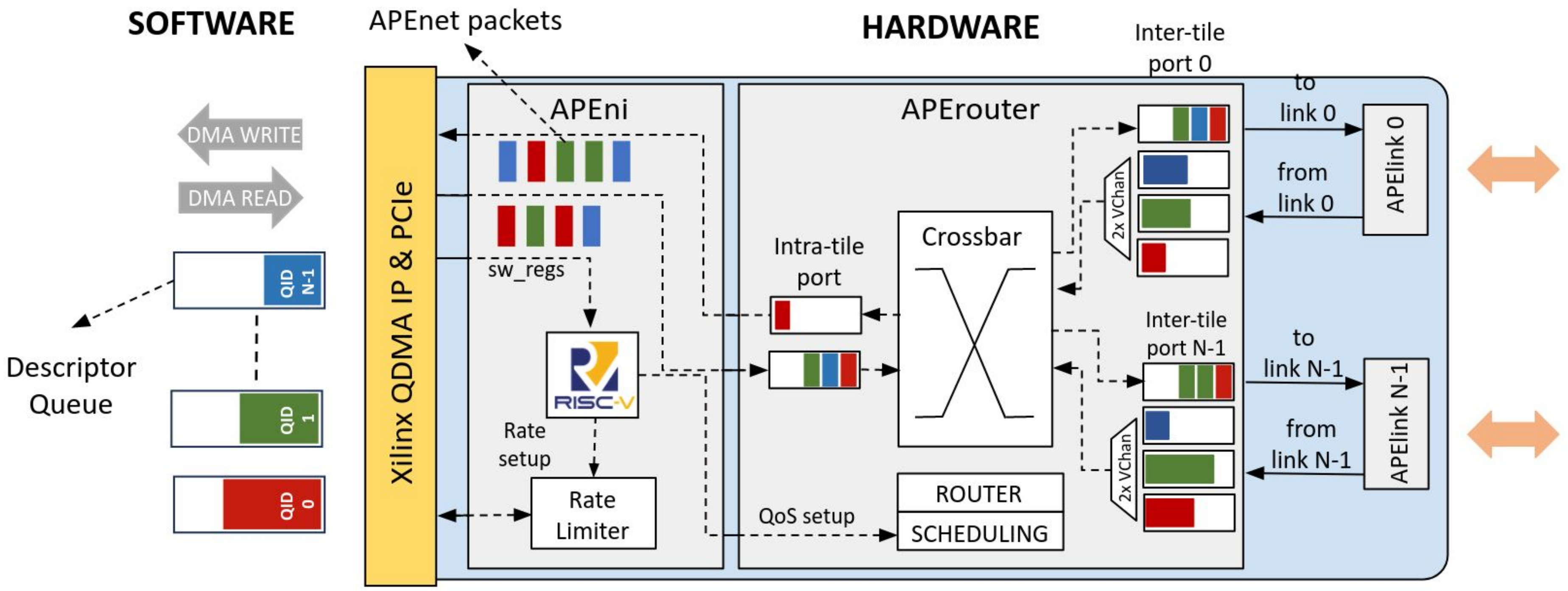
- Latency and bandwidth are evaluated from user space at source to user space at destination
- Both OSU microbenchmarks and synthetic tests are evaluated





Quality of Service on APEnetX

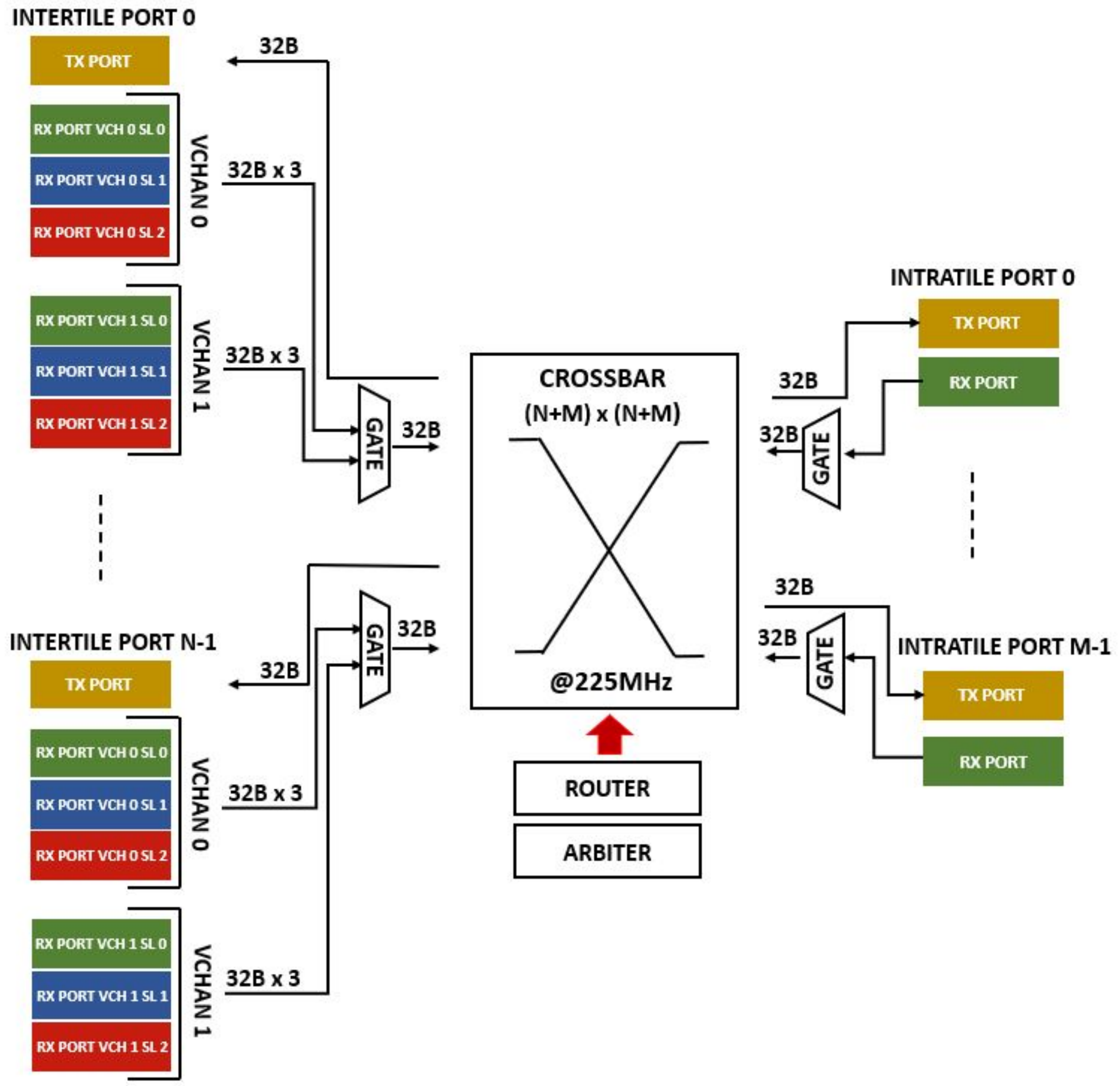
- Quality of Service (QoS) in HPC
  - Fair allocation of resources to optimize execution of critical tasks
  - Service Level (SL) differentiation: low latency or high bandwidth for data flows
- Inter-node QoS (Switch) and Intra-node QoS (PCIe interface)



Inter-node QoS

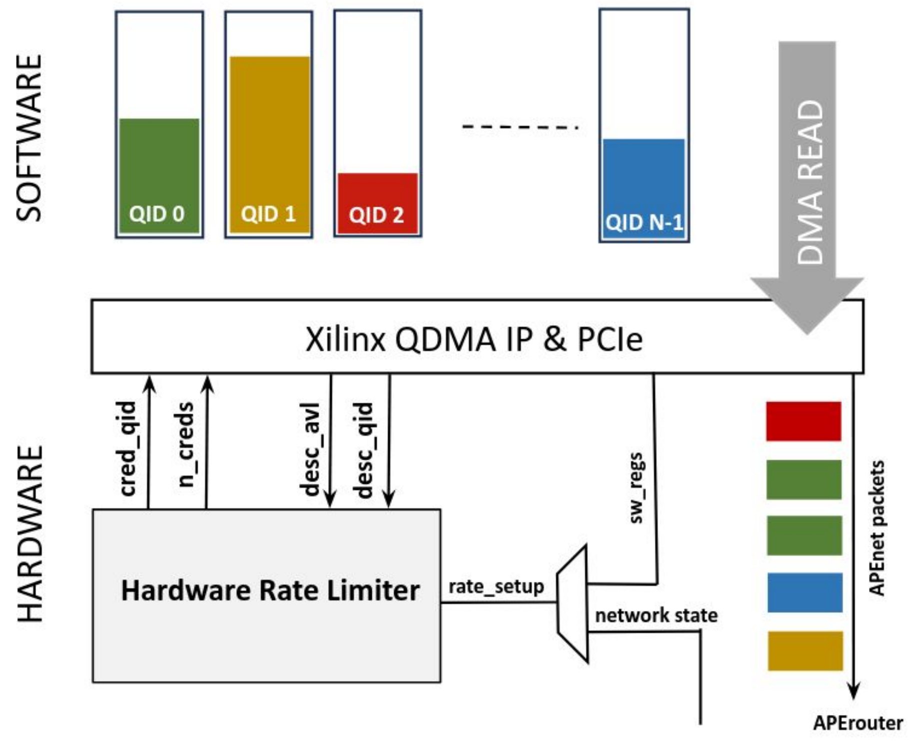
RECIPE

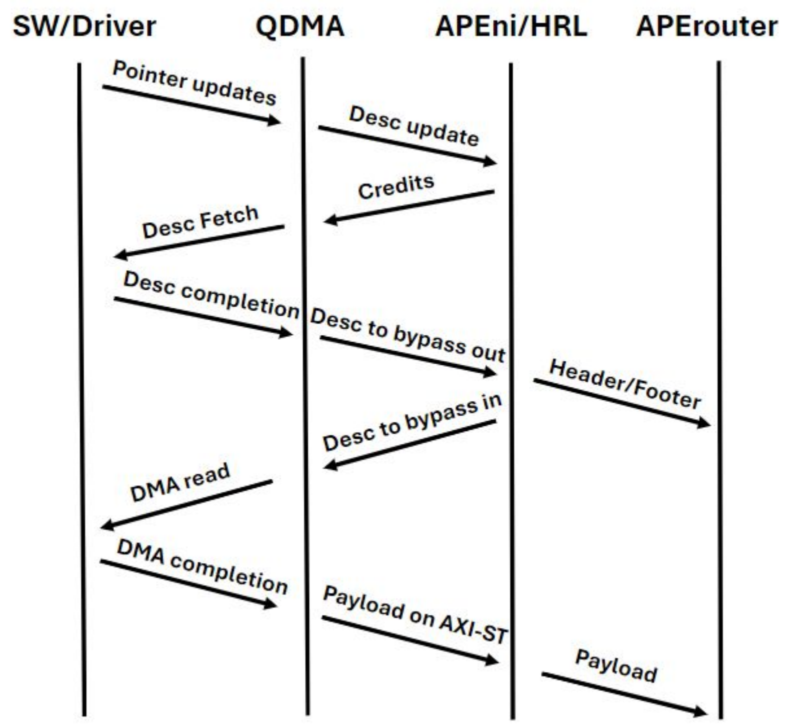
- Each RX Intertile port hosts one FIFO per SL
- Gate prioritizes the transmission of packets based on the SL following a scheduling algorithm
- Softcore setups or change the algorithm
- SL-based flow-control



Intra-node QoS

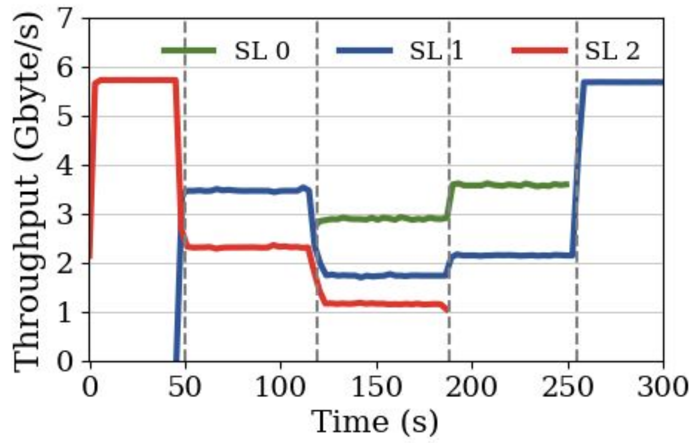
- Prioritization of DMA reads over PCIe based using Xilinx QDMA IP and Hardware Rate Limiter (HRL)
- HRL controls the QDMA by sending credits for each queue ID, prioritizing or limiting descriptor fetches to match bandwidth requirements

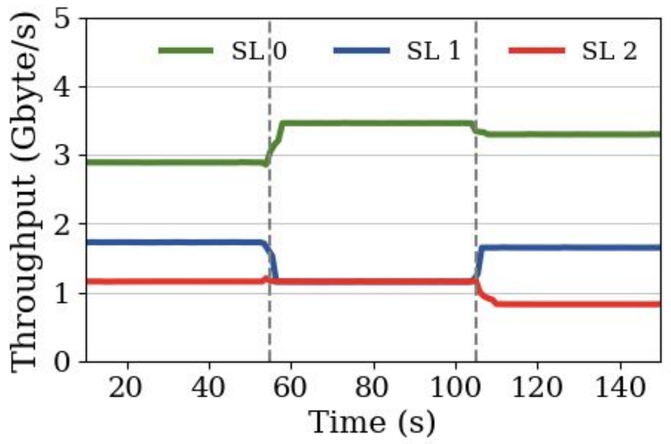


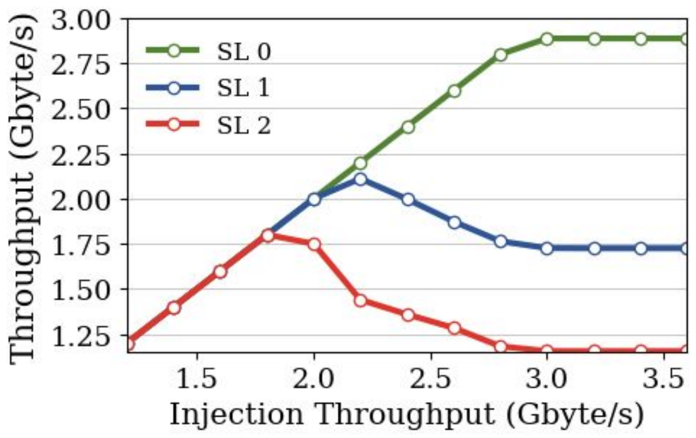


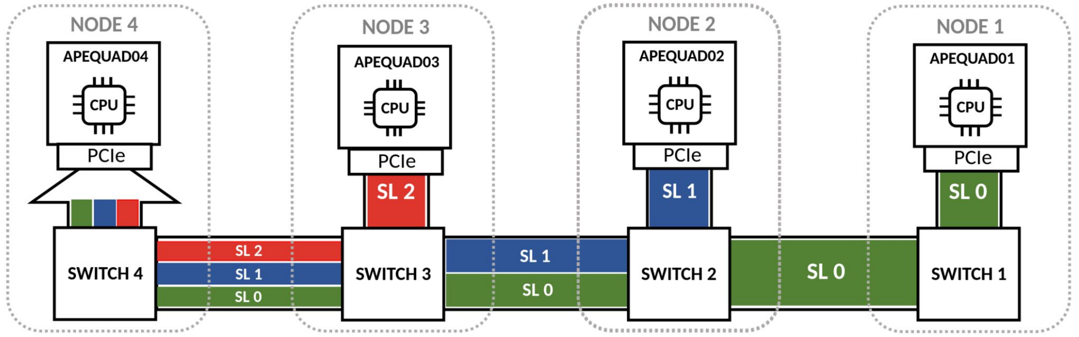
Quality of Service on APEnetX

- Validation on 4-node cluster equipped with APEnetX NICs
  - Scenario: Multiple flows with different Service Levels (**SL0**, **SL1**, **SL2**) share the same link



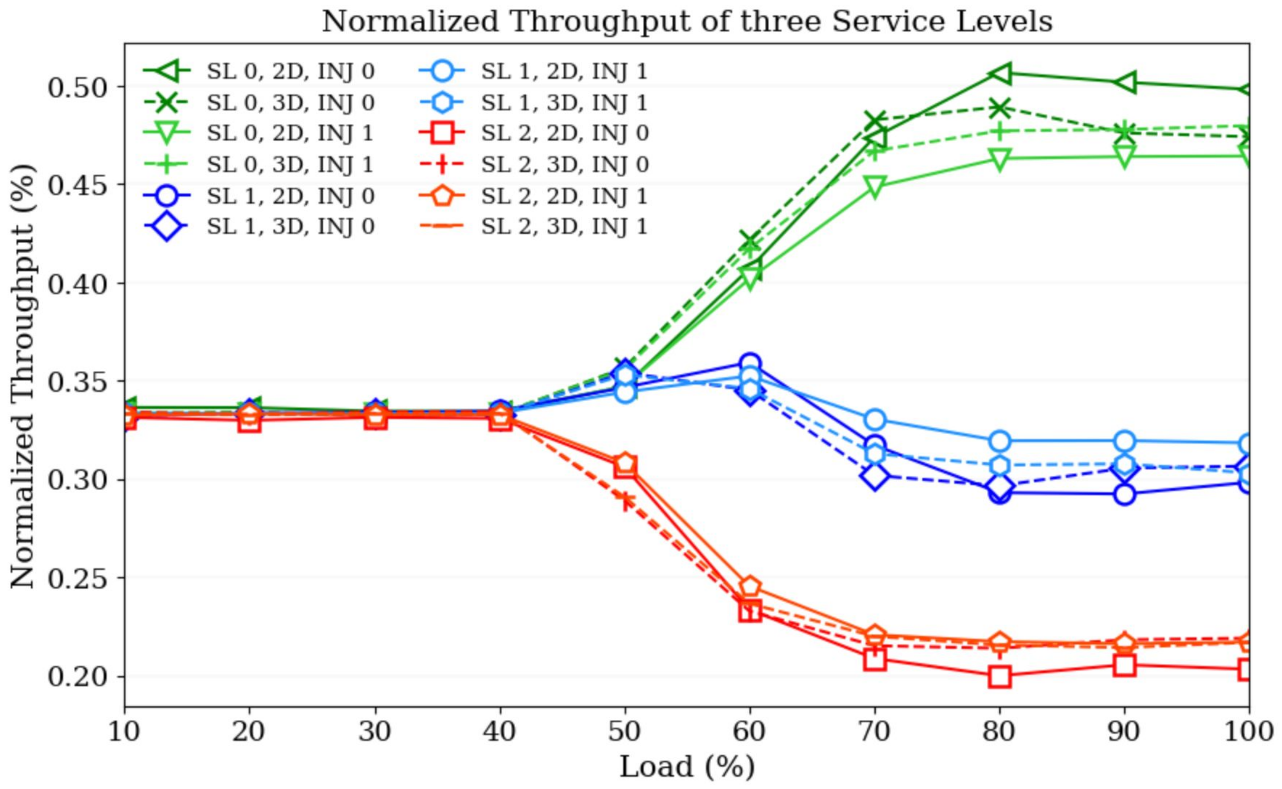






Quality of Service on APEnetX

- Validation on 100/1000-node simulated cluster equipped with APEnetX NICs
  - Scenario: Multiple flows with different Service Levels (**SL0**, **SL1**, **SL2**) share the same link



Conclusion

- APEnetX a FPGA-based NIC for torus interconnect
  - Network Interface: Xilinx QDMA IP + custom components
  - Sub 2 microsecond latency for small packets

- We investigated and implemented the support for QoS mechanism and validated the design
  - Physical APEnetX testbed up to 4 nodes
  - APEnetX simulator for larger scale networks

References

[1] F. Bodin et al. , *APE computers-past, present and future*, Published in: Comput.Phys.Commun. 147 (2002) 1-2, 402-409, DOI: 10.1016/S0010-4655(02)00314-4

[2] R. Ammendola et al., *Latest generation interconnect technologies in APEnet+ networking infrastructure*, Journal of Physics: Conference Series, https://dx.doi.org/10.1088/1742-6596/898/8/082035

[3] A. Biagioni et al., *RED-SEA: Network Solution for Exascale Architectures*, in 2022 25th Euromicro Conference on Digital System Design (DSD), 2022, pp. 712–719.

https://apegate.roma1.infn.it/

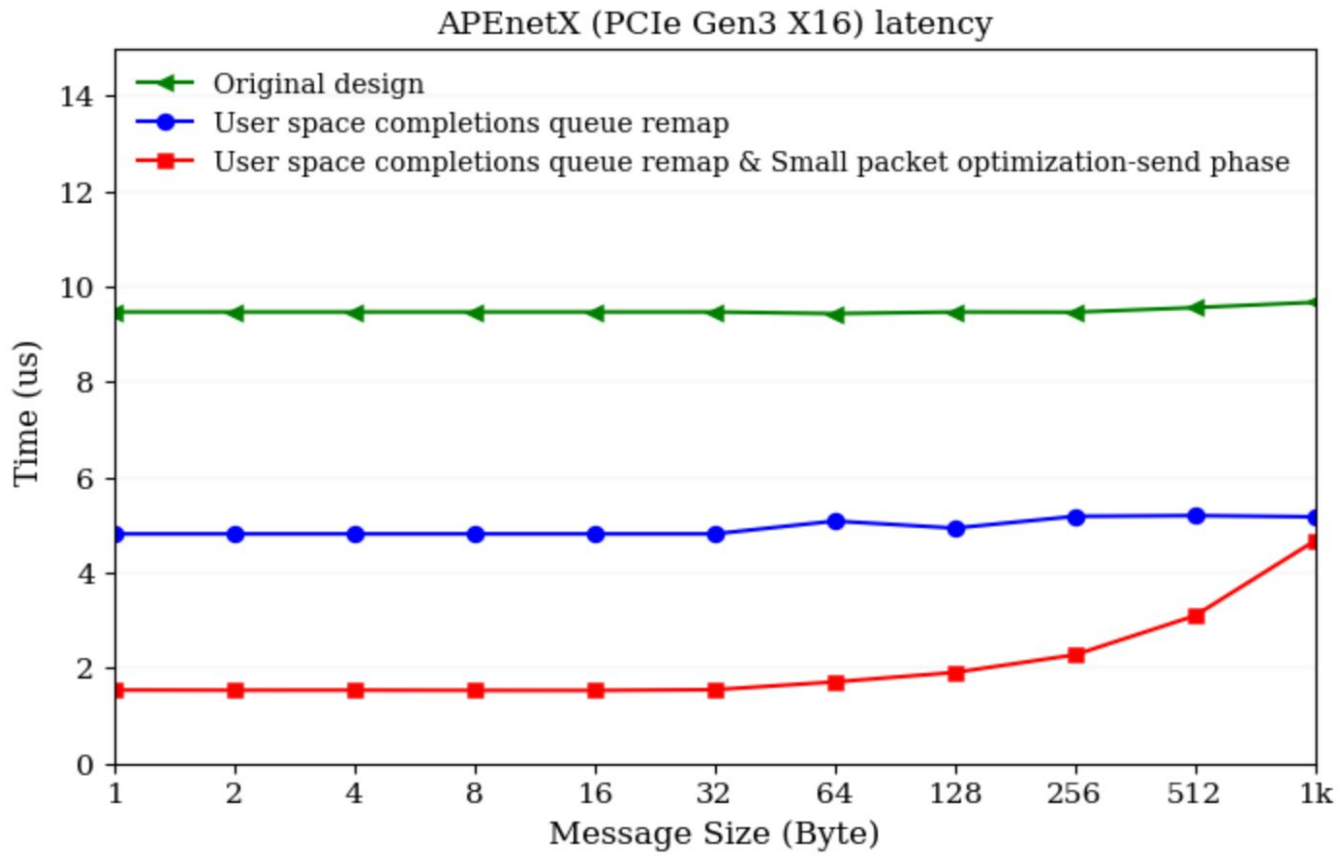
@APELab_INFN

Thank you!

BACKUP SLIDES

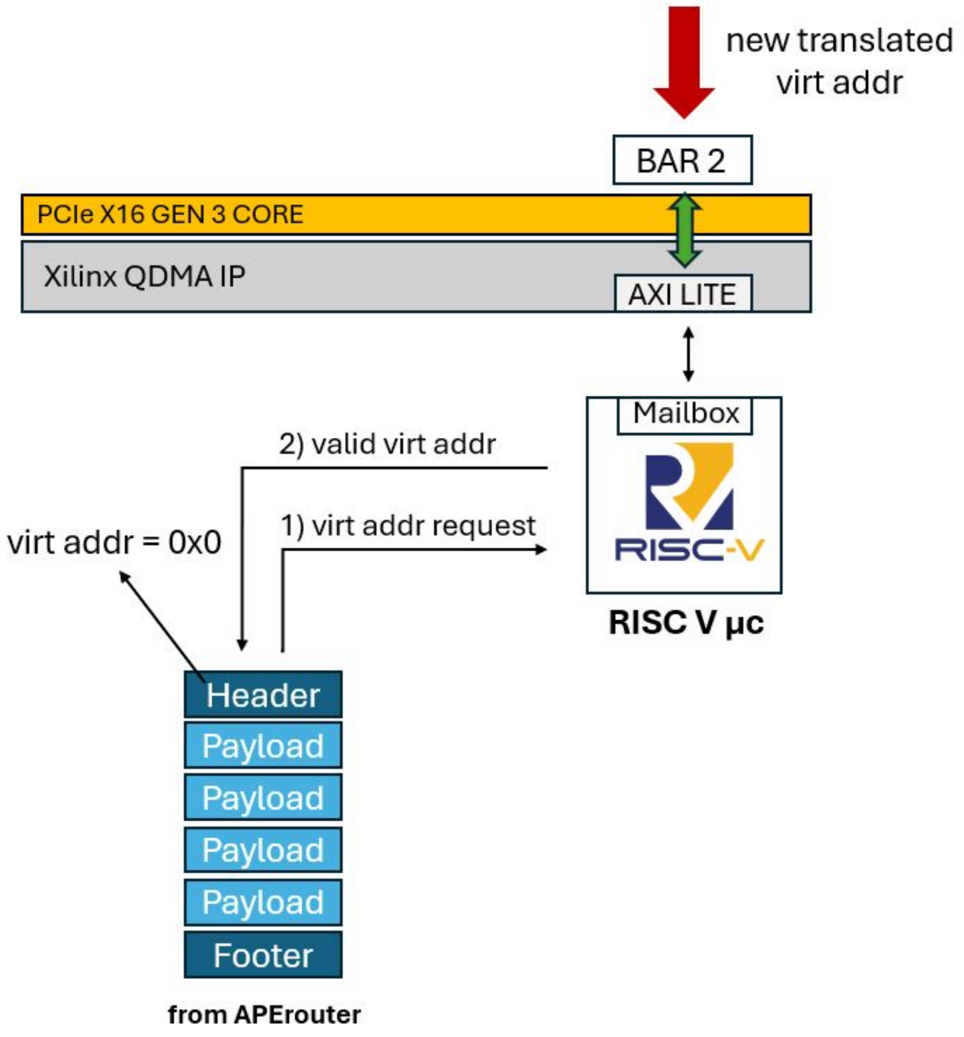
Latency breakdown

- The joint action of Fast-send for small packet and User space completion queue remap improves significantly the latency performance of the prototype

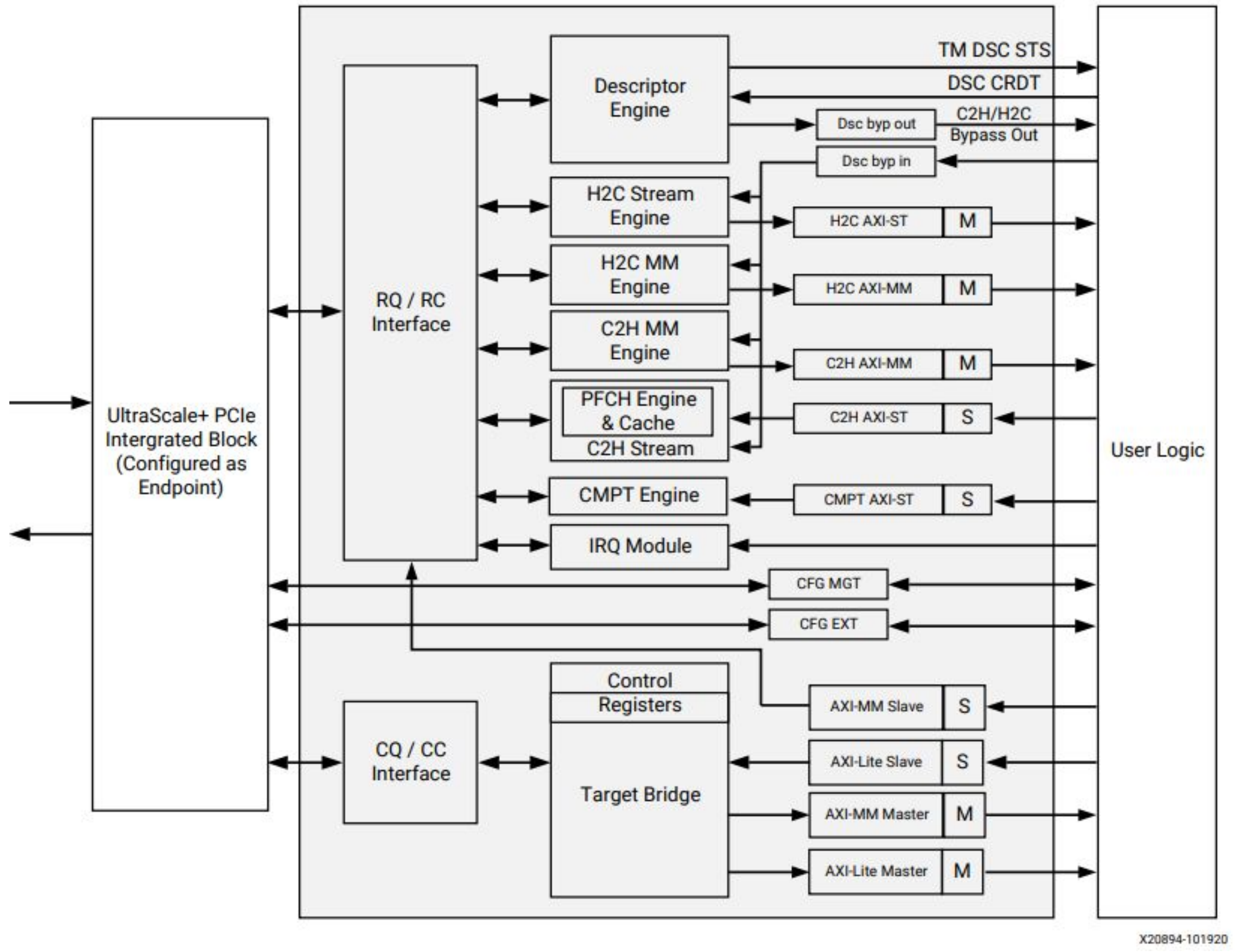


Optimization: RISC-V microcontroller

- The NIC integrates a 64-bit RISC-V core (based on the Rocket RV64GC)
    - A bidirectional Mailbox connects the µC to the host through BAR 2
    - Two queues toward/from the FPGA user logic are available, so that the NIC can issue requests to the µC
- The µC is being used to implement a “Zbuffer”
    - Simple way to send to a "known" virtual address (always 0x0) translated at the destination host
- In the future the µC will be used to help with congestion control functionalities of the NIC



QDMA architecture


UltraScale+ PCIe Intergrated Block (Configured as Endpoint)
RQ / RC Interface
CQ / CC Interface
Descriptor Engine
H2C Stream Engine
H2C MM Engine
C2H MM Engine
PFCH Engine & Cache
C2H Stream
CMPT Engine
IRQ Module
Control Registers
Target Bridge
TM DSC STS
DSC CRDT
C2H/H2C Bypass Out
Dsc byp out
Dsc byp in
H2C AXI-ST
M
H2C AXI-MM
M
C2H AXI-MM
M
C2H AXI-ST
S
CMPT AXI-ST
S
CFG MGT
CFG EXT
AXI-MM Slave
S
AXI-Lite Slave
S
AXI-MM Master
M
AXI-Lite Master
M
User Logic
X20894-101920