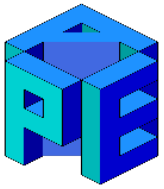


NaNet: a flexible and configurable low-latency NIC for real-time GPU-based systems

alessandro.lonardo,
andrea.biagioni,
francesca.locicero,
piero.vicini
ottorino.frezza
@roma1.infn.it

People Involved

Riccardo Fantechi

Gianluca Lamanna

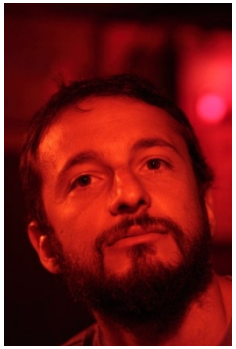
Alessandro Lonardo

Pier Stanislao Paolucci

Davide Rossetti

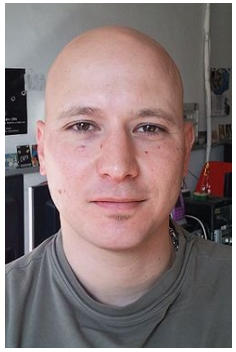
Marco Sozzi

Piero Vicini



Roberto Ammendola

Andrea Biagioni



Ottorino Frezza

Francesca Lo Cicero

Felice Pantaleo

Francesco Simula

Laura Tosoratto

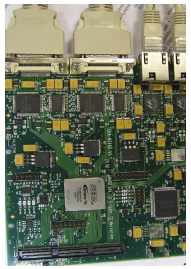
The NA62 Experiment at CERN

- The goal is the precision measurement of the ultra-rare decay process $K^+ \to \pi^+ \nu \bar{\nu}$
  - Extremely sensitive to any unknown new particle.
- Many (uninteresting) events: 10^7 decays/s.
- Ultra-rare: 1 target event on 10^{10} particle decays
  - Expect to collect ~100 events in 2/3 years of operation
  - Need to select the interesting events, filtering efficiently and quickly the data stream coming from the detectors.

Multi-level Trigger Architecture in HEP Experiments

Multi-level triggering reduces the amount of data to a manageable level operating on a set of successive stages between event detection and the storage of their associated data.

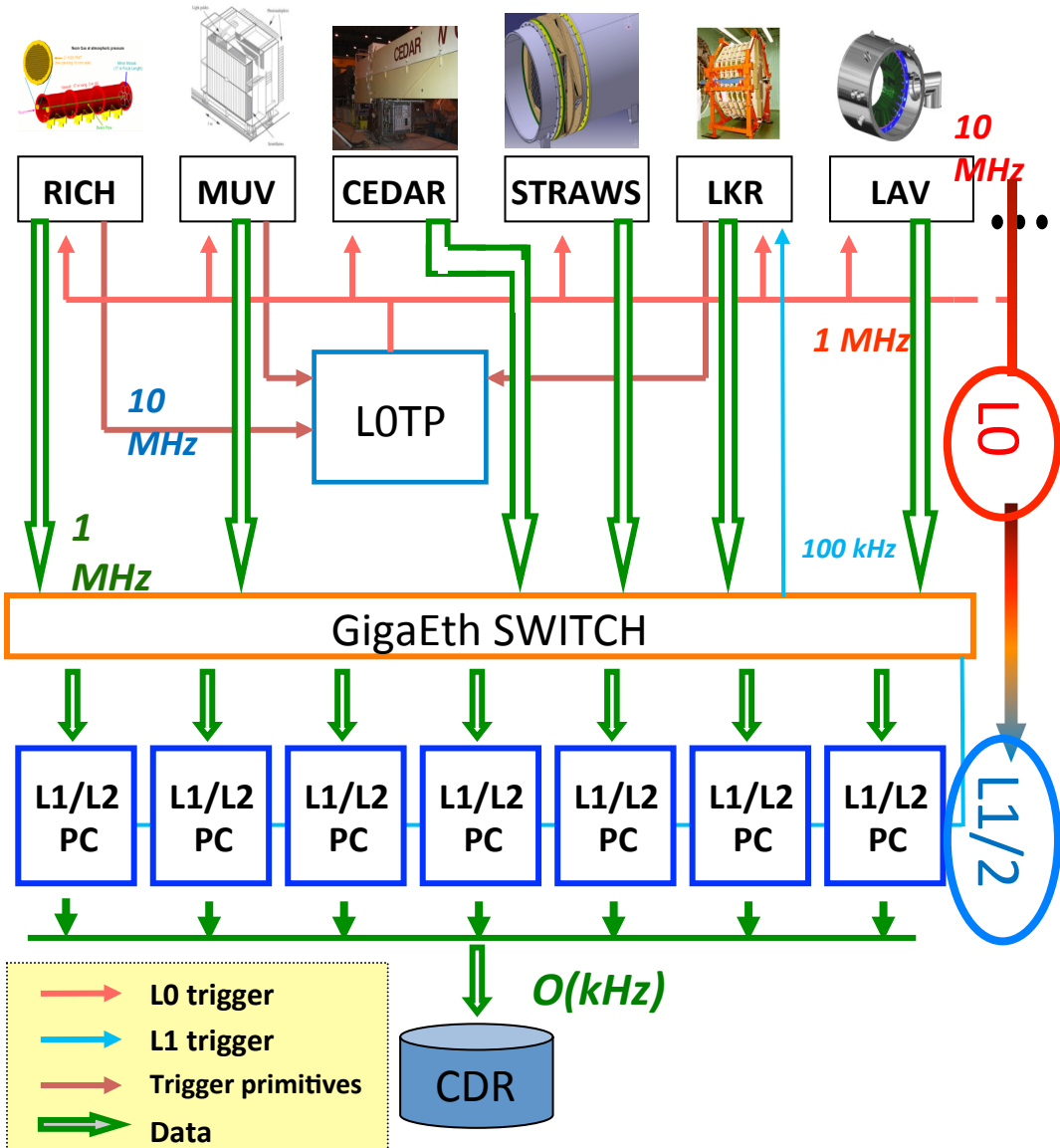
Lower level: custom hardware, simple patterns in reduced information from a few fast detectors;
►move full detector data from buffers to PC memories if OK



Higher levels: hierarchical chain of simplified reconstruction algorithms on some detectors, specific features of interesting events, using switched computing farms;
►store detector data to permanent storage if OK

Offline analysis: full reconstruction algorithms
►reduced data samples for further physics analysis

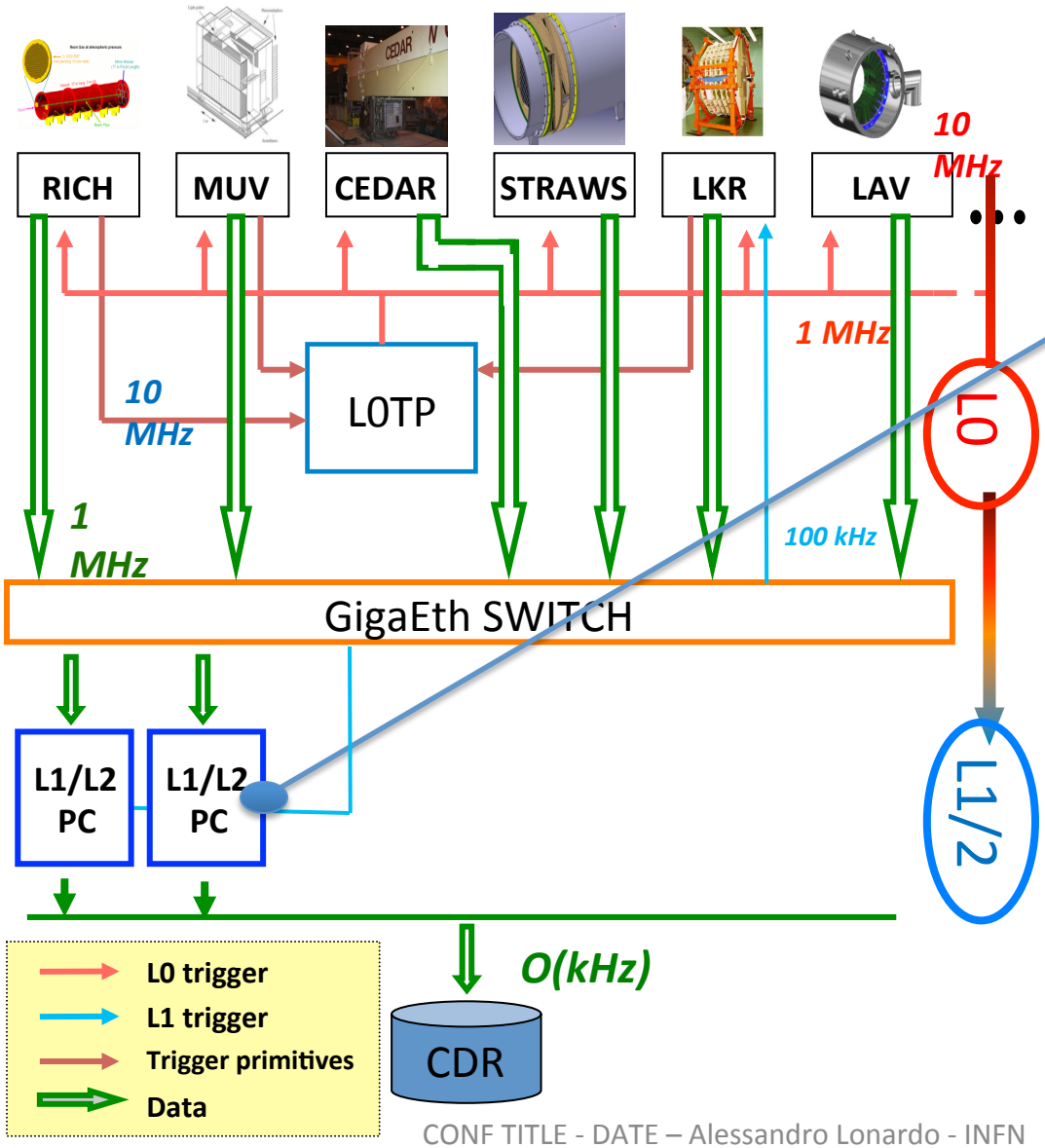
The NA62 Trigger and Data Acquisition System



L0 trigger:
hardware
synchronous
FPGA custom-design
10 MHz to 1 MHz
Max latency 1 ms

L1/L2 trigger:
software asynchronous
switched PC farm
1MHz to some kHz

Using GPUs in the NA62 L1/L2 Trigger

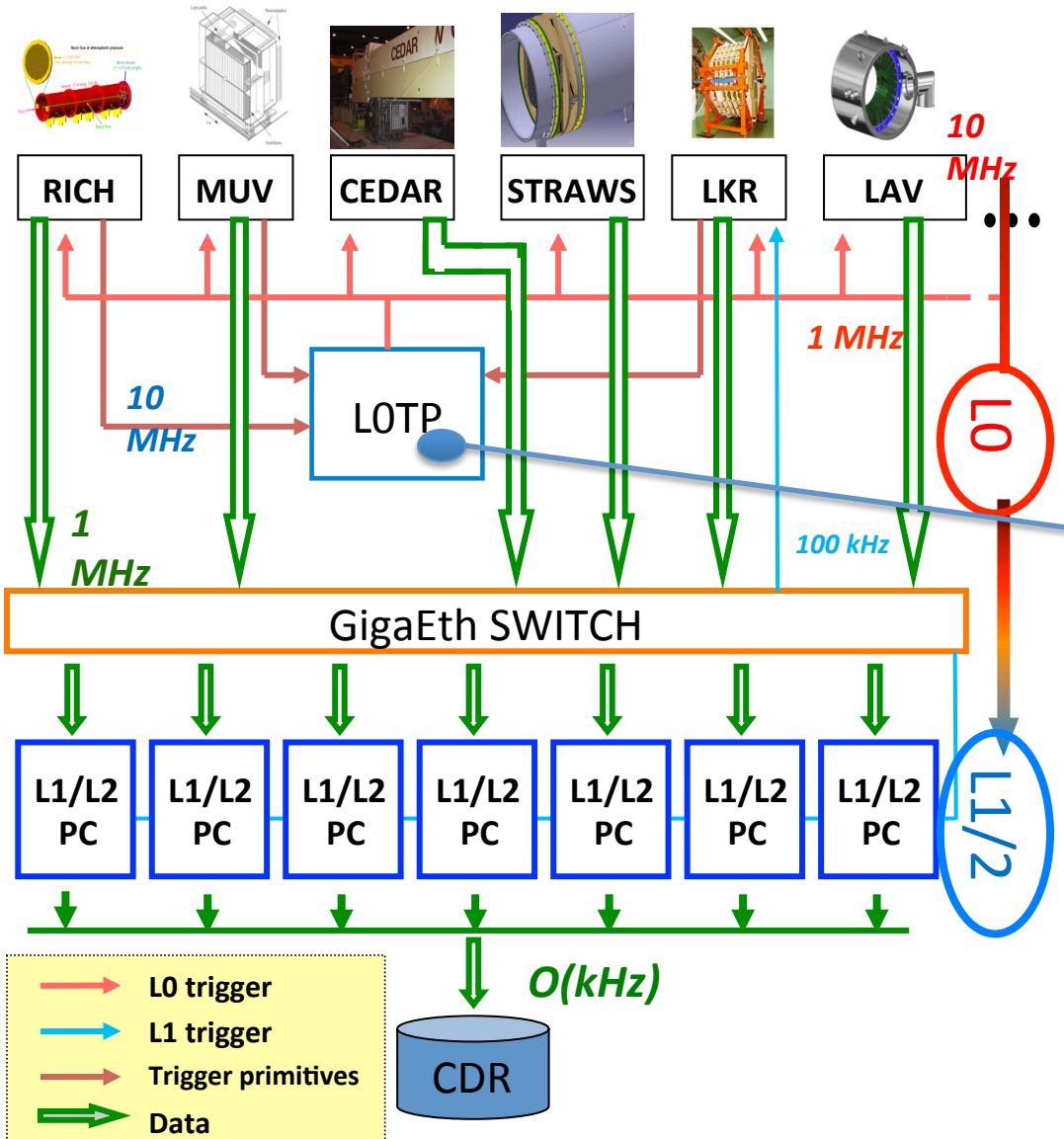


Exploit GPU computing power to scale down the number of nodes in the farm:

- event analysis paralelizzation on many-core architectures.

Not the topic of this talk.

Using GPUs in the NA62 L0 Trigger

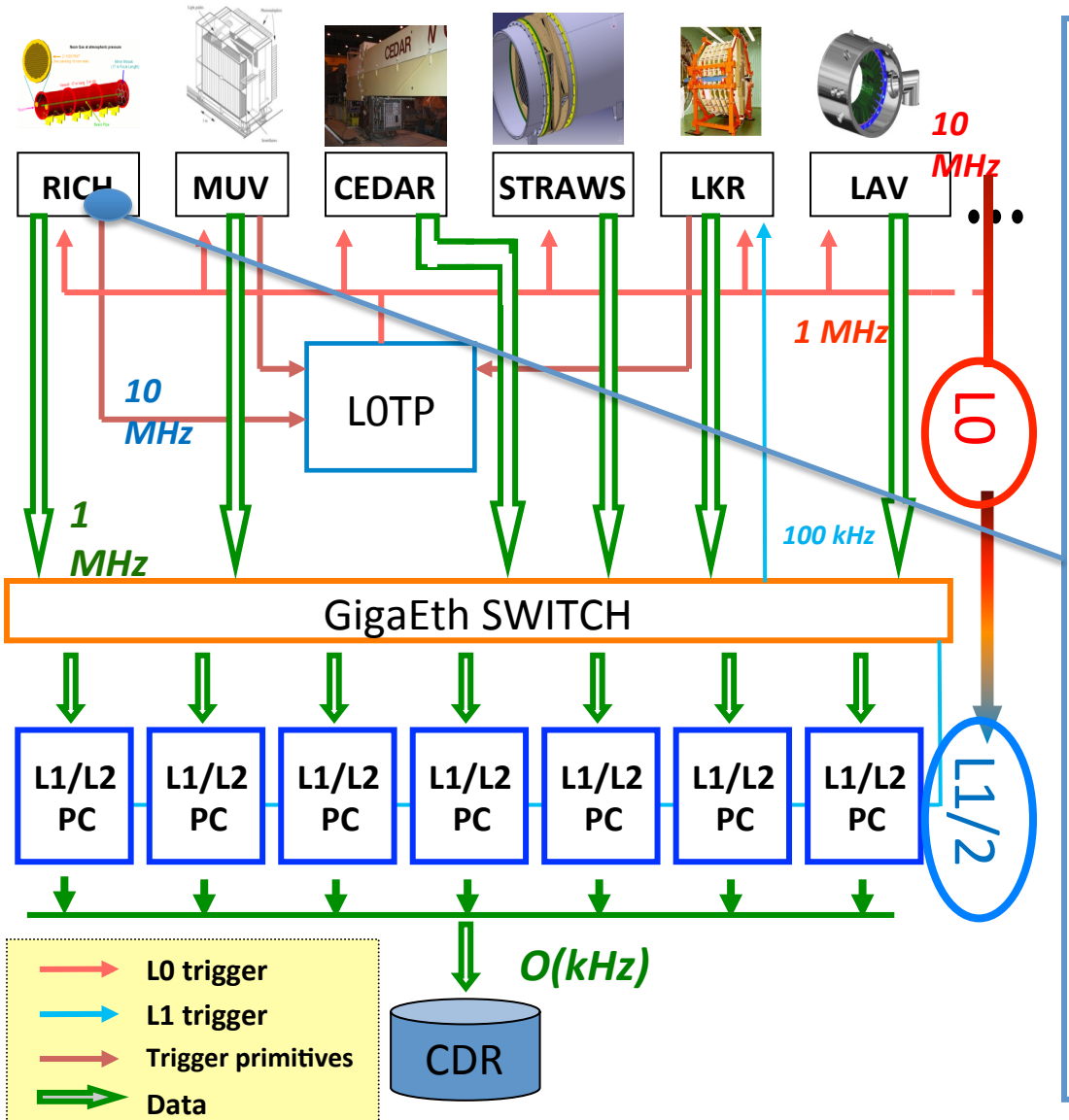


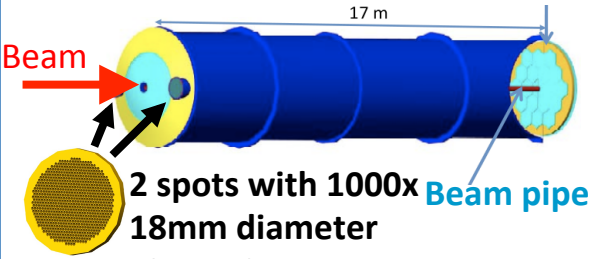
Replace custom hardware with a GPU-based system performing the same task but:

- Programmable
- Upgradable
- Scalable
- Cost effective
- Increasing selection efficiency of interesting events implementing more demanding algorithms.

The topic of this talk.

Using GPUs in the NA62 L0 Trigger: the RICH Detector Case Study

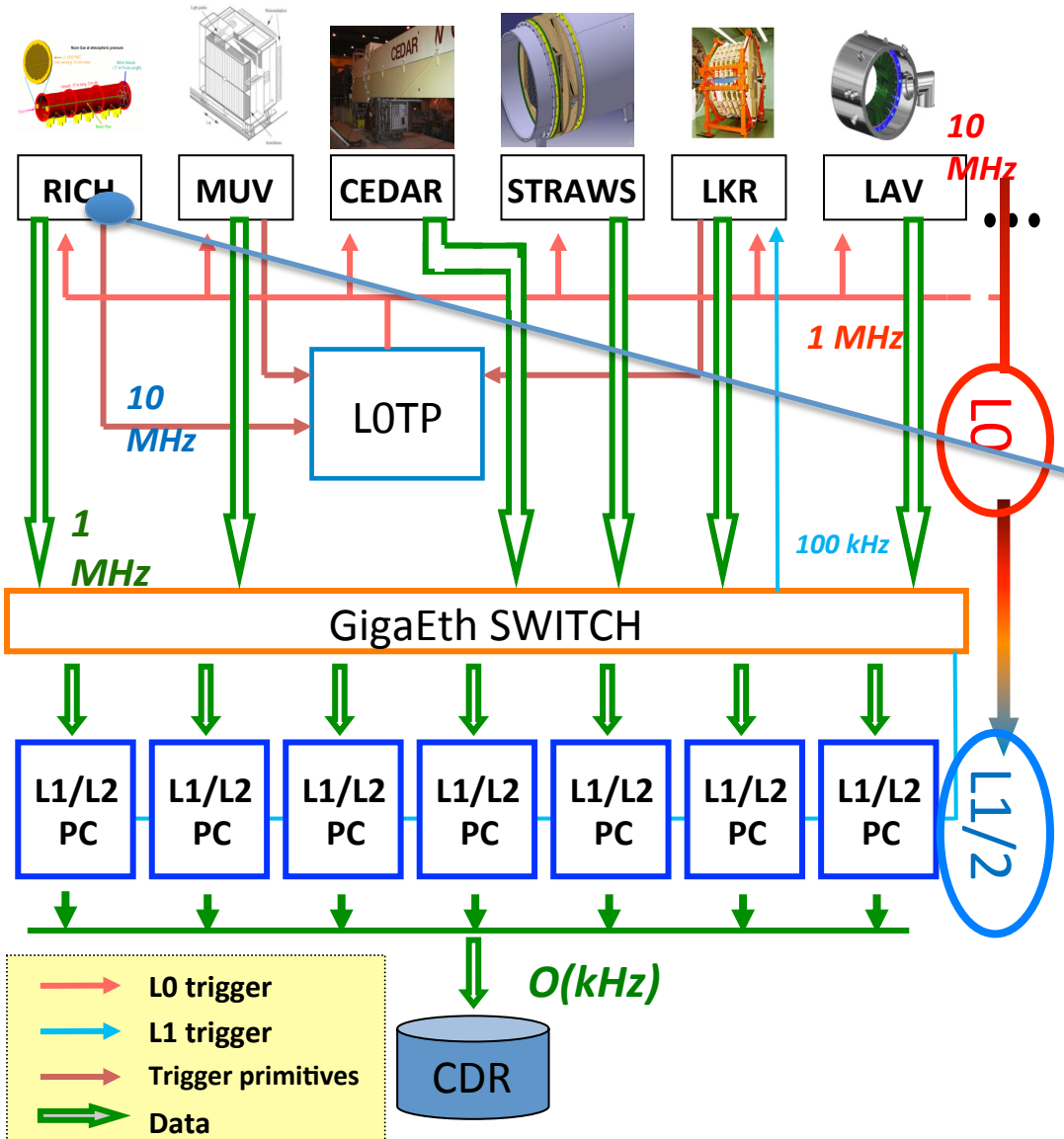


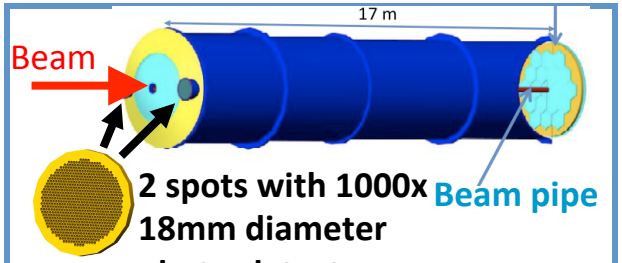


Ring-imaging Čerenkov detector: charged relativistic particles produces cone-shaped light flash in Neon-filled vessel.

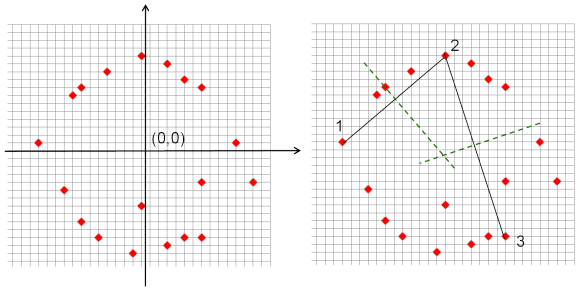
- 100 ps time resolution
- **10 MHz** event rate
- 20 photons detected on average per event (**hits** on photo-detectors)
- **48 byte** per event (max 32 hits per event)

Using GPUs in the NA62 L0 Trigger: the RICH Detector Case Study



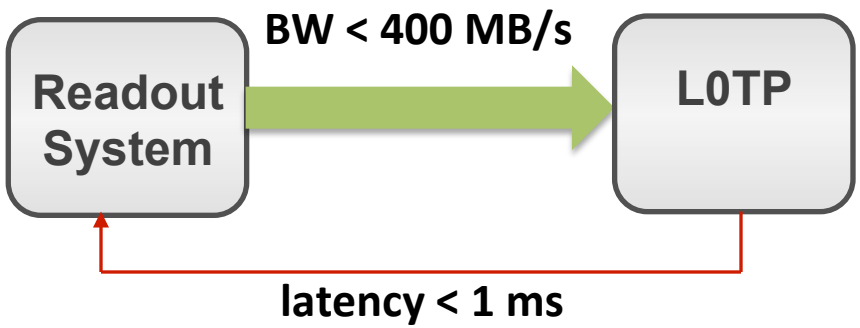


- Matching of partial circular hit pattern on photo-detectors done on GPU

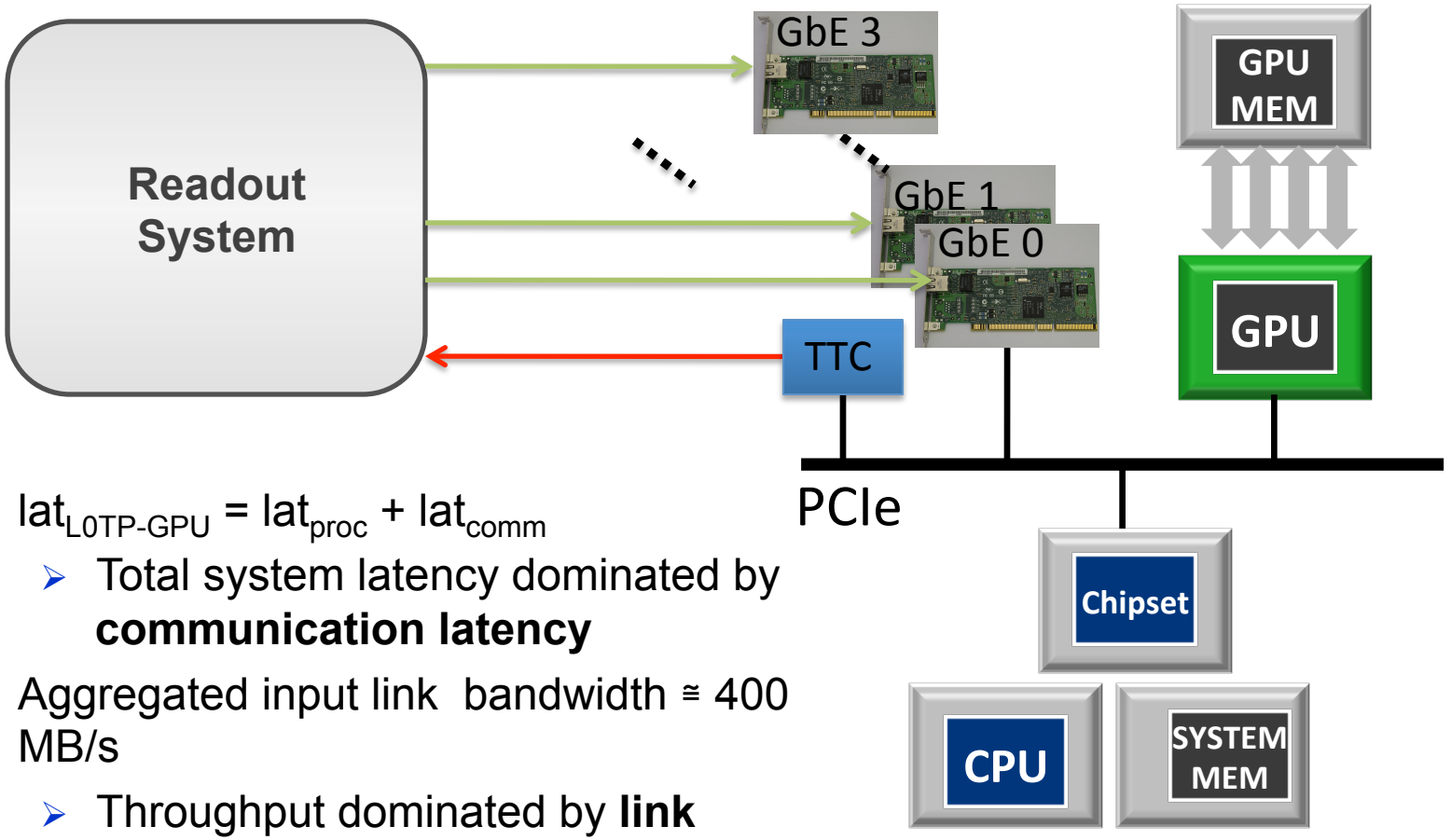


NA62 RICH L0 Trigger Processor Requirements

- Network Protocols/Topology: UDP over Point-to-Point (no switches) GbE.
- Throughput
  - Input event primitive data rate < 400MB/s (on 4 GbE links)
  - Output of trigger results < 50 MB/s (on 1 GbE link)
  - Output trigger signal on LVDS link
- System response latency < 1 ms
  - determined by the size of Readout System memory buffer storing event data candidated to be passed to higher trigger levels.



GPU-Based RICH Level 0 Trigger Processor (commodity NIC)



- $lat_{L0TP\text{-}GPU} = lat_{proc} + lat_{comm}$
  - Total system latency dominated by **communication latency**
- Aggregated input link bandwidth ≅ 400 MB/s
  - Throughput dominated by **link bandwidth**

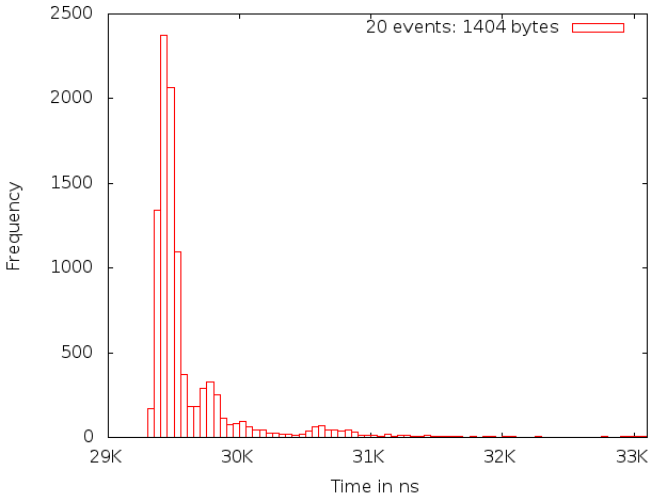
GPU-Based RICH Level 0 Trigger Processor – Processing Latency

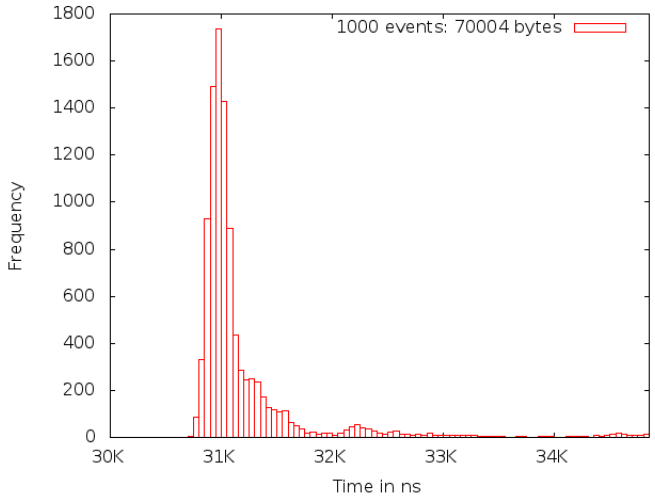
- **lat_{proc}** : time needed to perform rings pattern-matching on the GPU with input and output data on device memory.
- Test setup: supermicro X8DTG-DF motherboard (Intel Tylersburg chipset), dual Intel Xeon E5620, Nvidia S2050, Intel 82576 Gigabit Network Connection, CentOS 5.9, CUDA 4.2, Nvidia driver 310.19.
- Measurement:

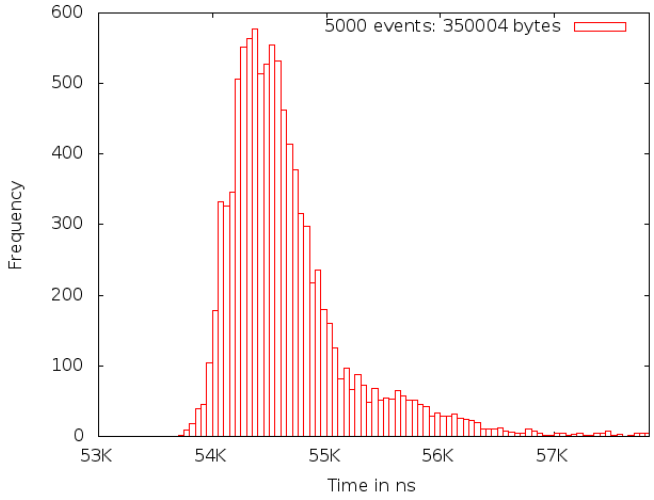
```
for (int sample=0; sample < MAXSAMPLES; sample++) {
  starting_cycles[sample]=get_cycles();
  math<<<blocksPerGrid, threadsPerBlock, smemSize>>>(data_gpu,
r_data_gpu, utility);
  cudaDeviceSynchronize();
  stopping_cycles[sample]=get_cycles();
}
```

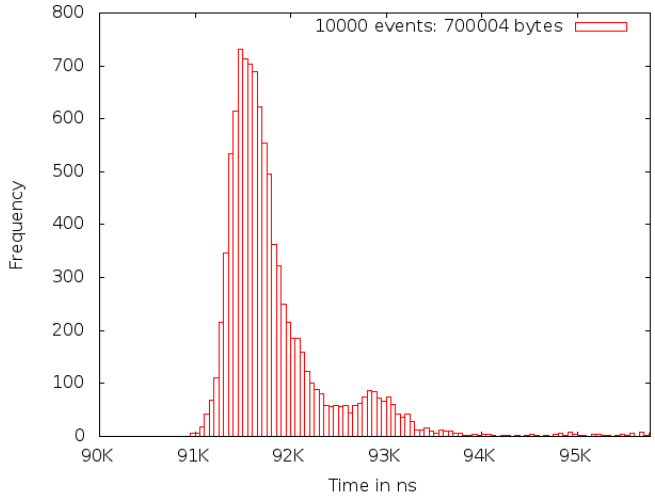
- Cycles are read from the x86 Time Stamp Counter

GPU-Based RICH Level 0 TP Processing Latency Distribution

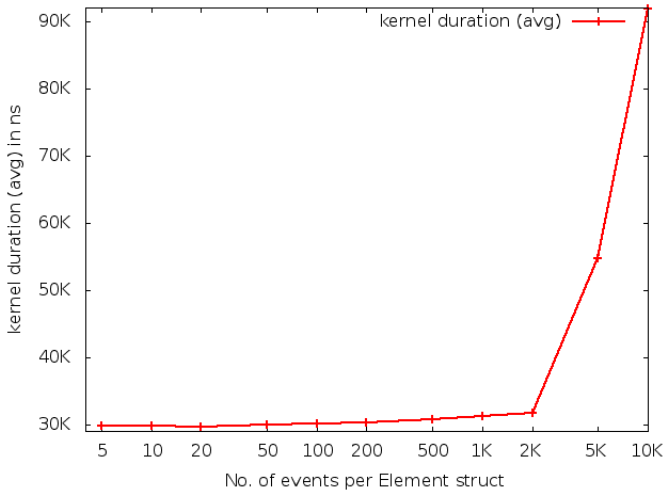


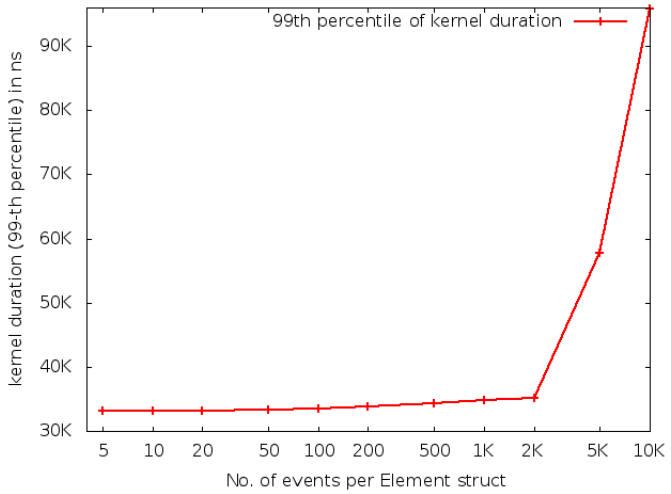






GPU-Based RICH Level 0 TP Processing Latency vs. Input Events

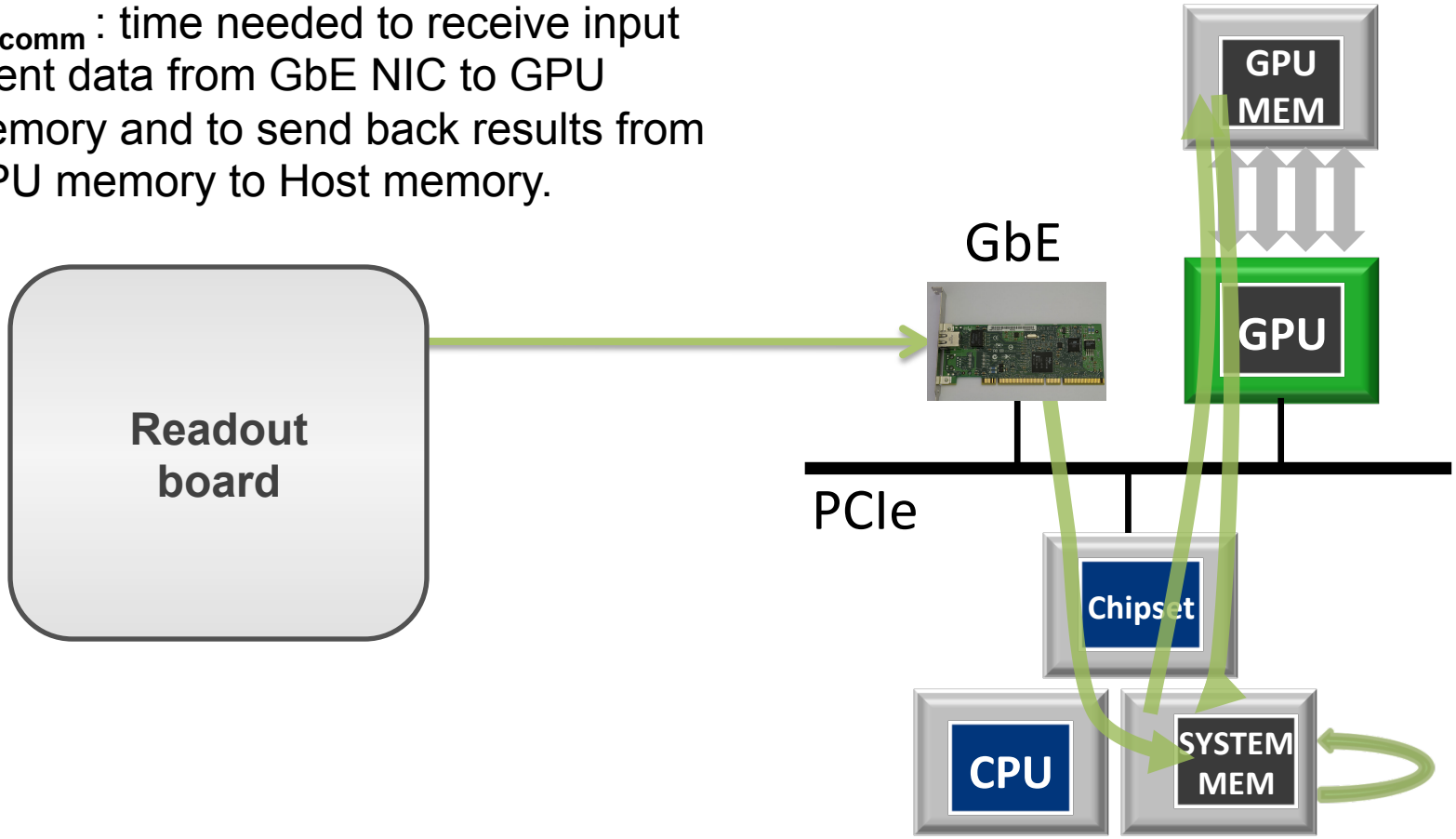




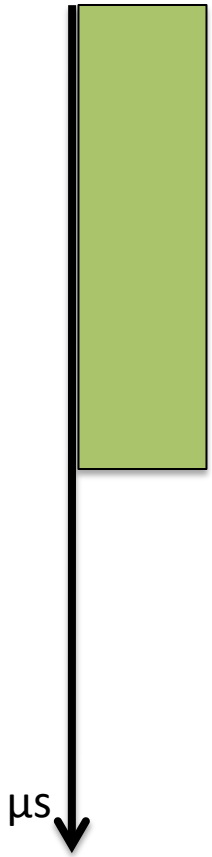
- $\mathbf{lat_{proc}}$ is stable
- max 1/10 of the time budget available
- what about moving events data to be processed, from RICH Readout boad to the GPU memory?

GPU-Based RICH Level 0 Trigger Processor – Communication Latency

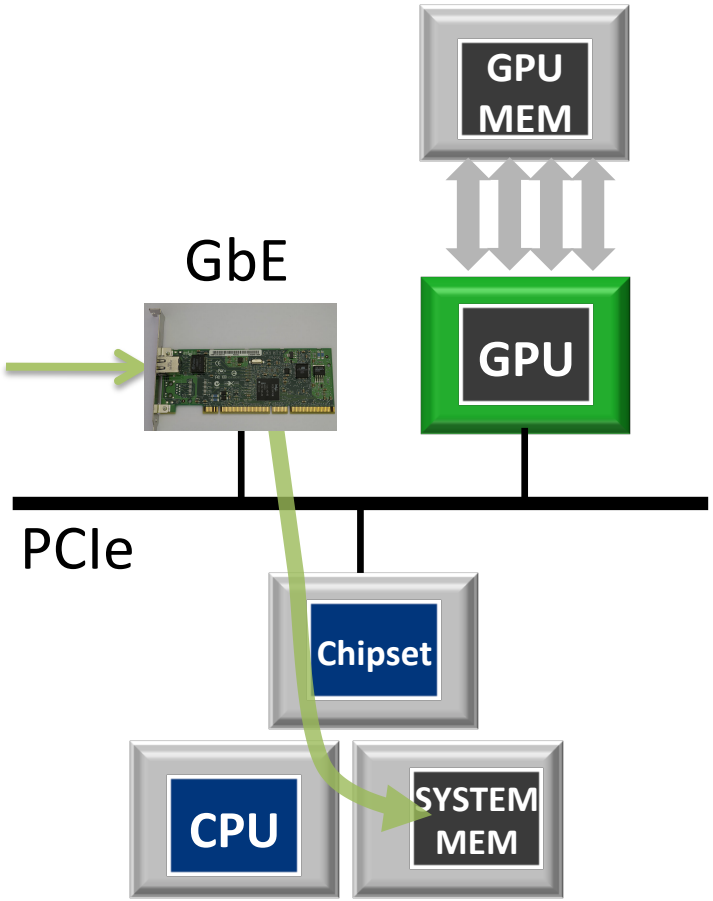
- **lat_{comm}** : time needed to receive input event data from GbE NIC to GPU memory and to send back results from GPU memory to Host memory.



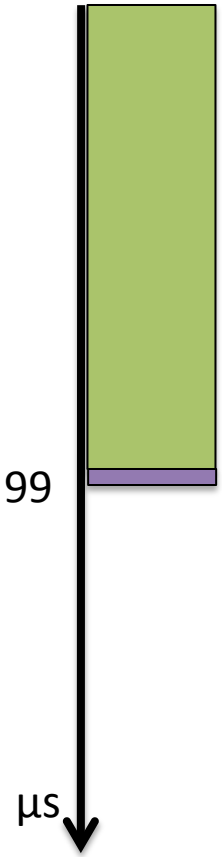
GPU-Based RICH Level 0 Trigger Processor – Communication Latency



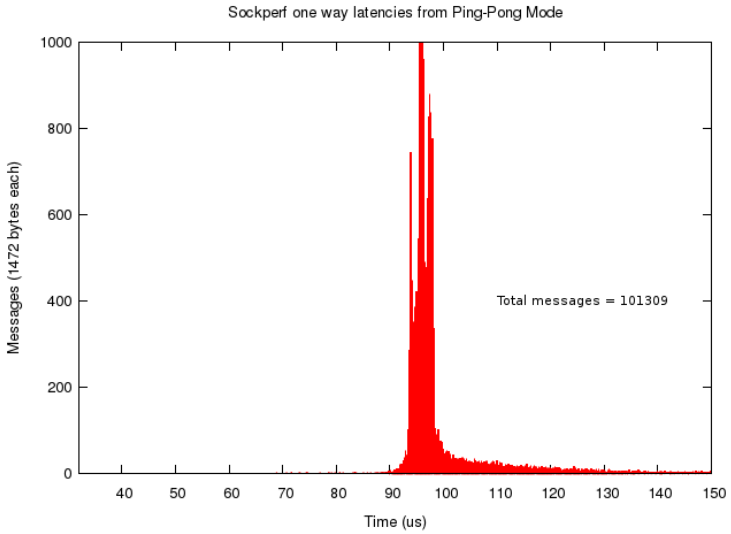
- 20 events data (1404 byte) sent from Readout board to the GbE NIC are stored in a receiving host kernel buffer.
- T=0 start of send operation on the Readout Board.

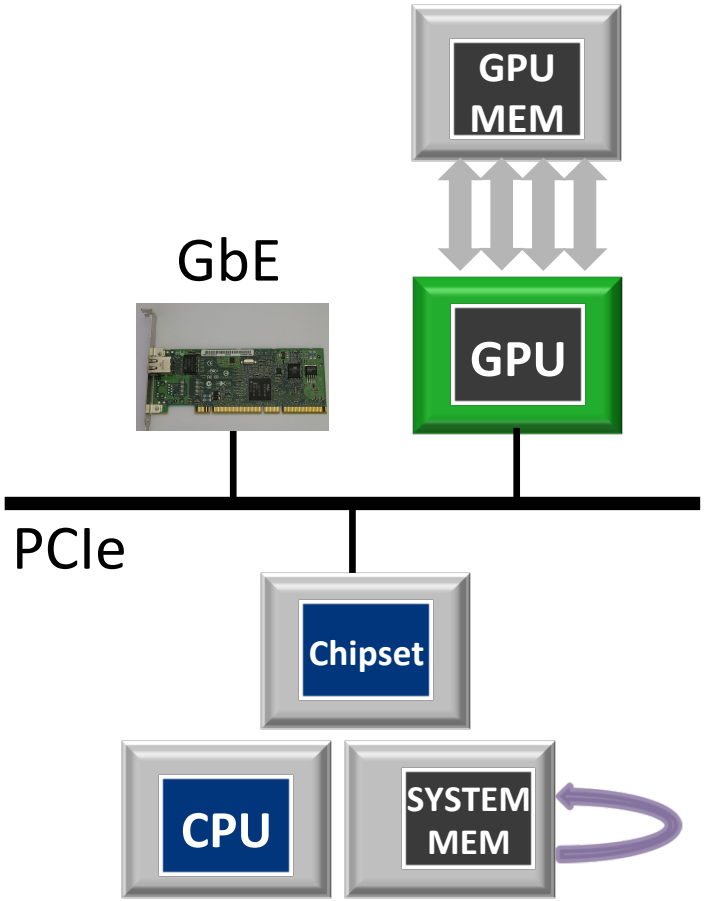


GPU-Based RICH Level 0 Trigger Processor – Communication Latency

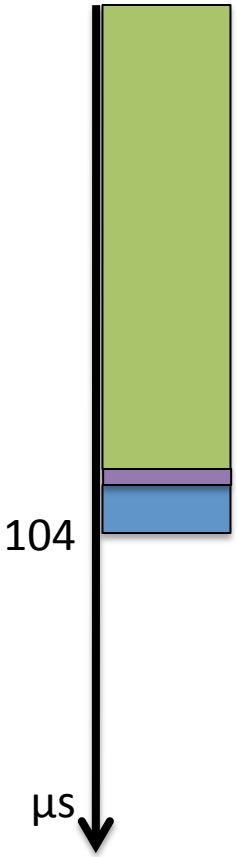


Data are copied from kernel buffer to a user space buffer.

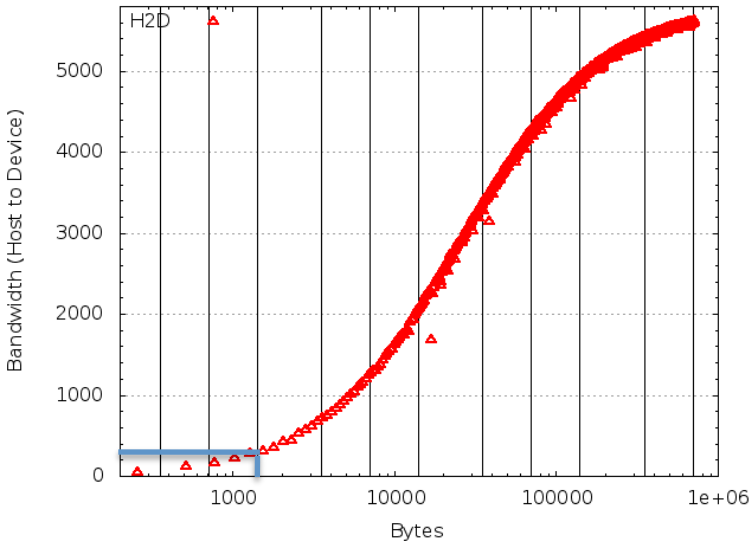


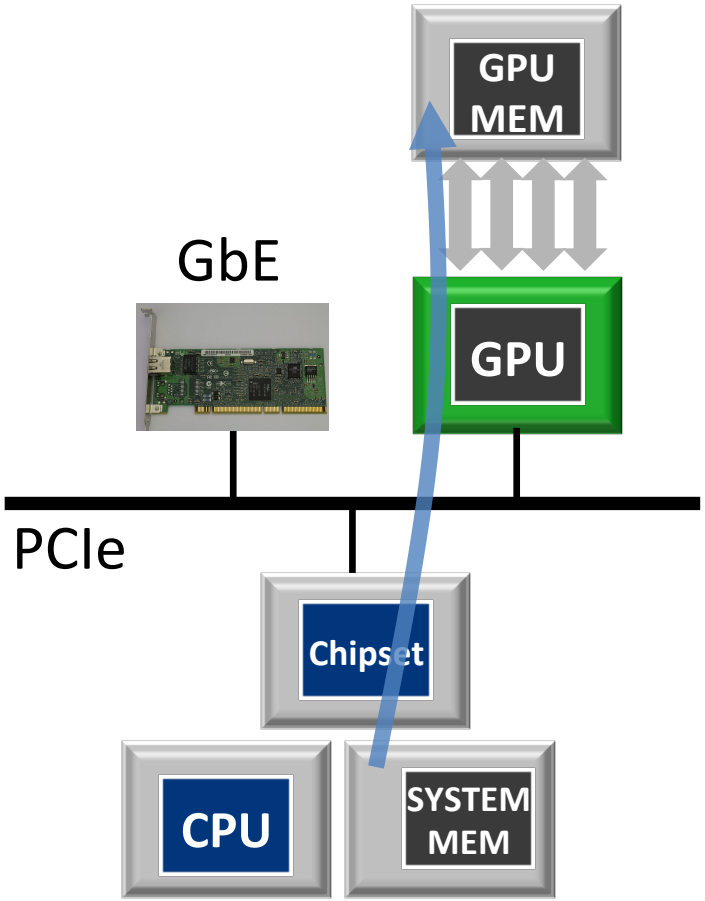


GPU-Based RICH Level 0 Trigger Processor – Communication Latency

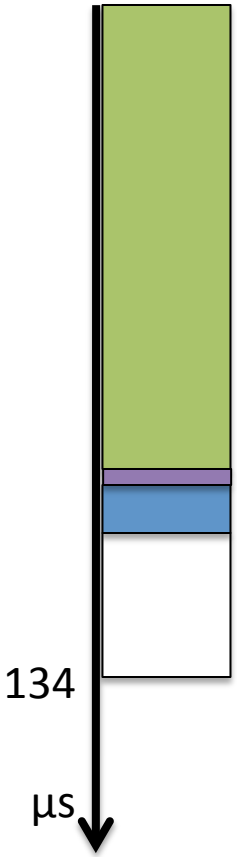


Data are copied from system memory to GPU memory

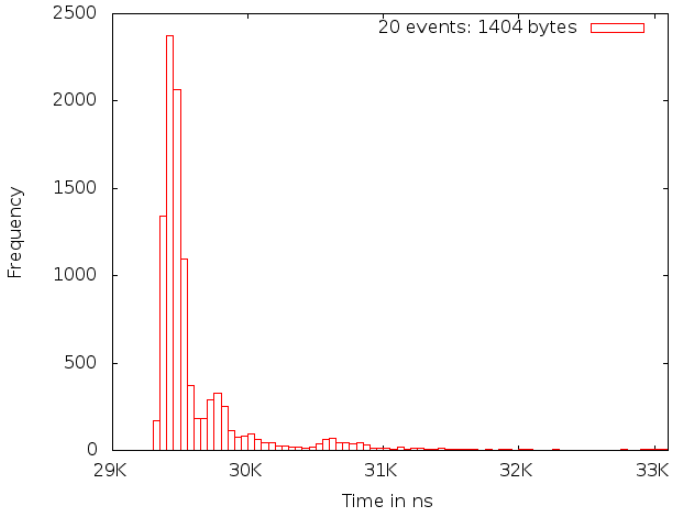


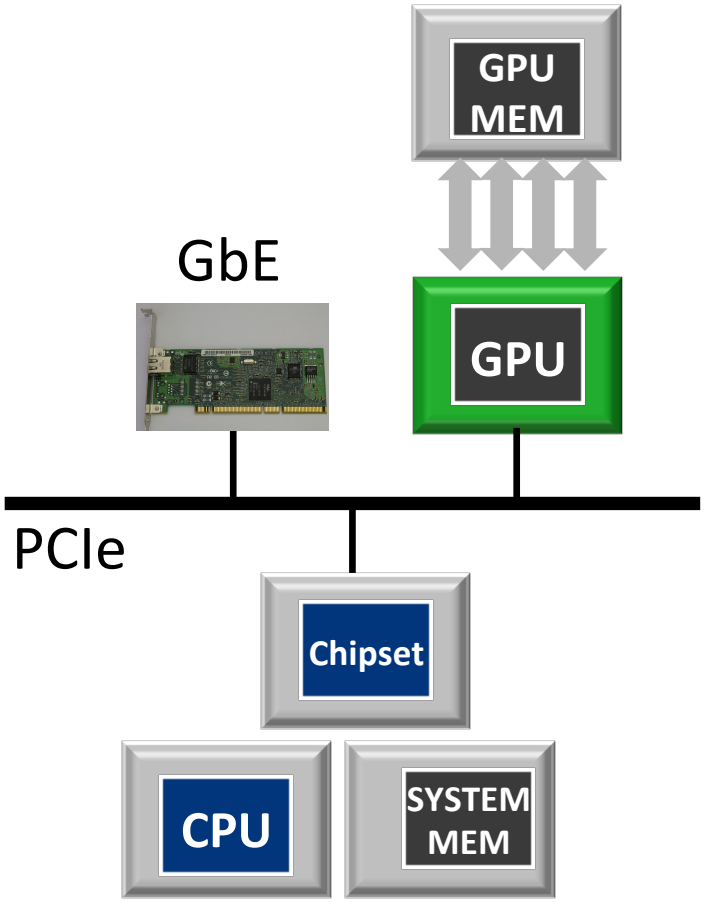


GPU-Based RICH Level 0 Trigger Processor – Communication Latency

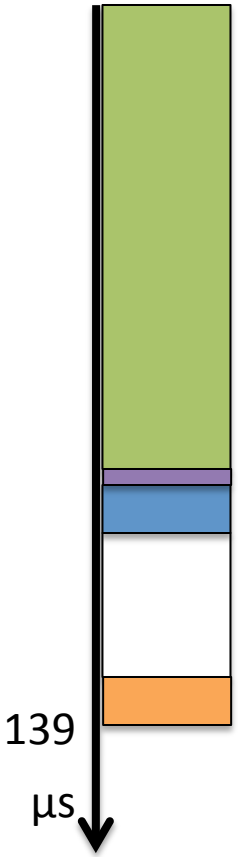


Ring pattern-matching GPU Kernel is executed, results are stored in device memory.

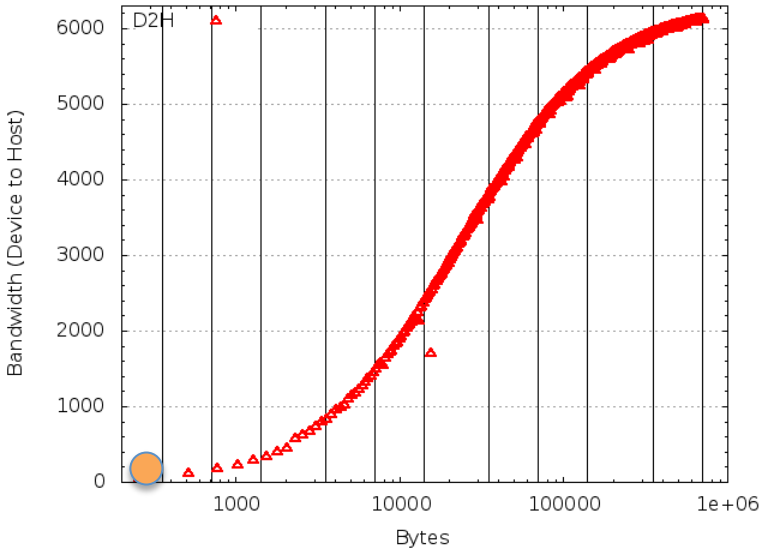


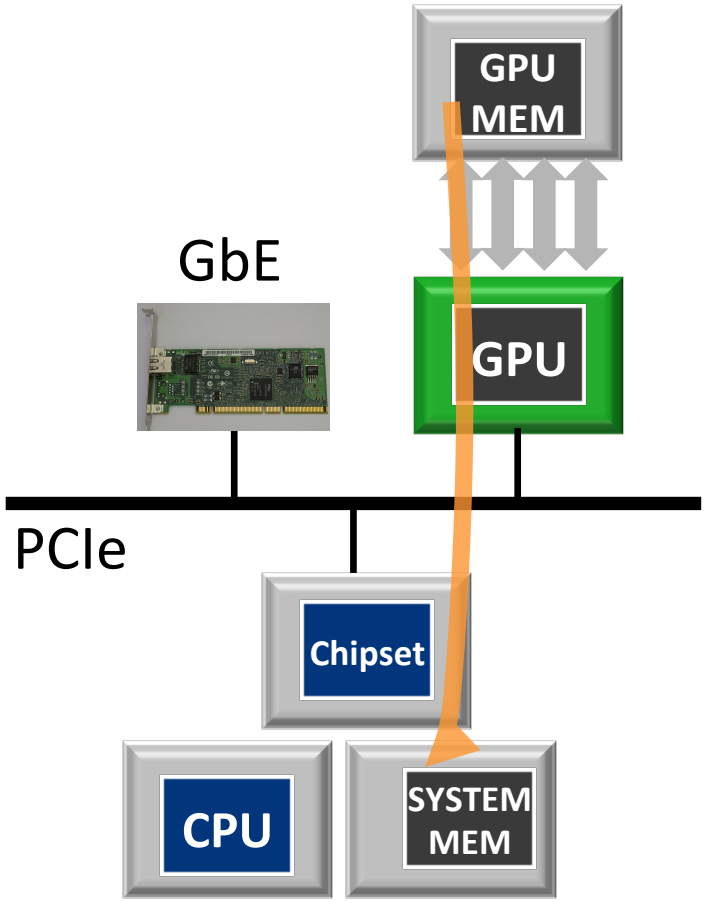


GPU-Based RICH Level 0 Trigger Processor – Communication Latency

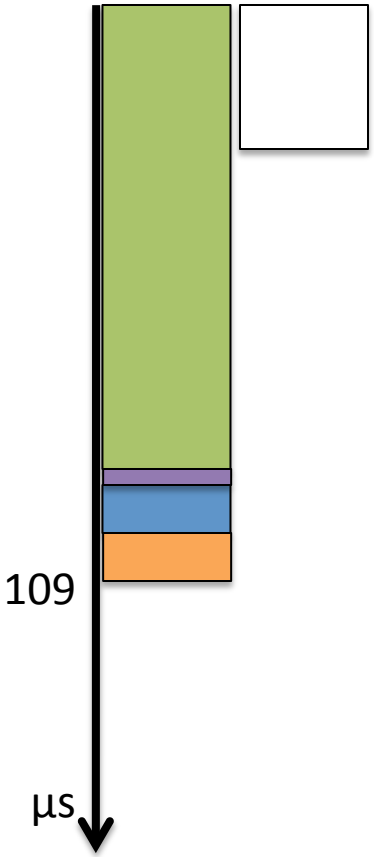


Results are copied from GPU memory to system memory (322 bytes – 20 results)

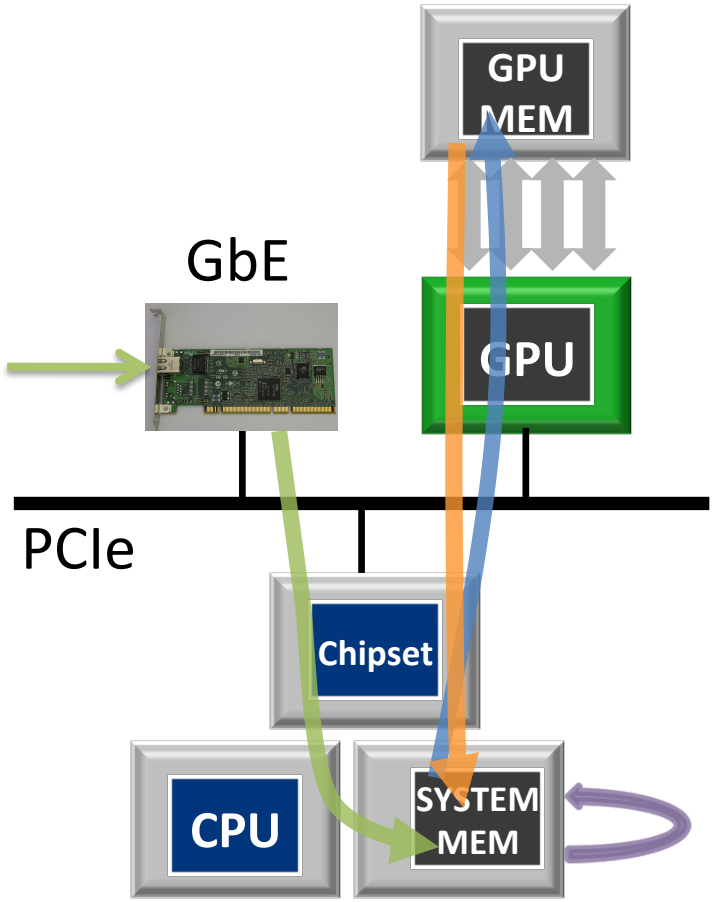




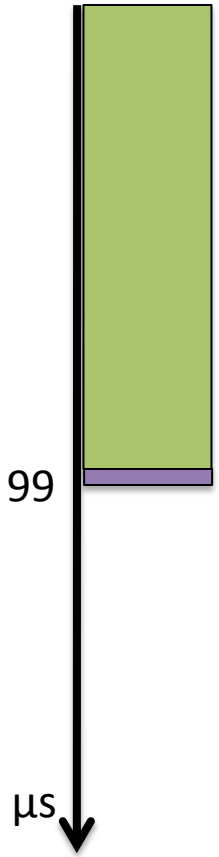
GPU-Based RICH Level 0 Trigger Processor – Communication Latency



- lat_{comm} ≅ 110 µs
- 4 x lat_{proc}

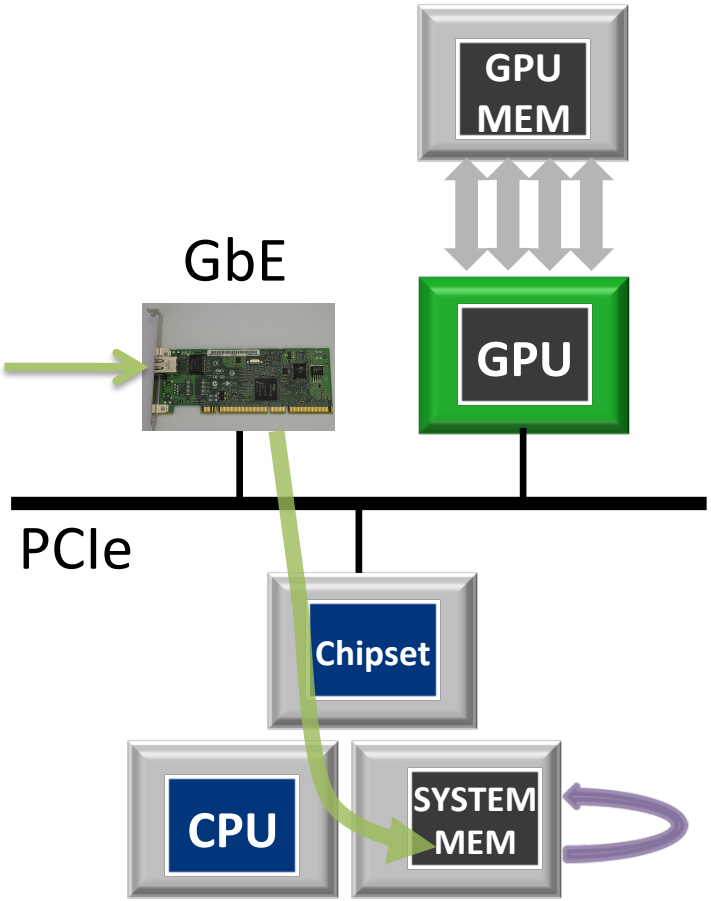


GPU-Based RICH Level 0 Trigger Processor – Communication Latency



➢Fluctuations on the GbE component of lat_{comm} may hinder the real-time requisite, even at low events count.

```
sockperf: Summary: Latency is 99.129 usec
sockperf: Total 100816 observations; each
percentile contains 1008.16 observations
sockperf: ---> <MAX> observation = 657.743
sockperf: ---> percentile 99.99 = 474.758
sockperf: ---> percentile 99.90 = 201.321
sockperf: ---> percentile 99.50 = 163.819
sockperf: ---> percentile 99.00 = 149.694
sockperf: ---> percentile 95.00 = 116.730
sockperf: ---> percentile 90.00 = 105.027
sockperf: ---> percentile 75.00 = 97.578
sockperf: ---> percentile 50.00 = 96.023
sockperf: ---> percentile 25.00 = 95.775
sockperf: ---> <MIN> observation = 64.141
```



NaNet

- Problem: lower communication latency and its fluctuations.

- How?
  1. Injecting directly data from the NIC into the GPU memory with no intermediate buffering.
  2. Offloading the CPU from network stack protocol management, avoiding OS jitter effects.

- NaNet solution:
  - Use the APEnet+ FPGA-based NIC implementing GPUDirect RDMA.
  - Add a network stack protocol management offloading engine to the logic (UDP Offloading Engine).

APEnet+

3D Torus Network:

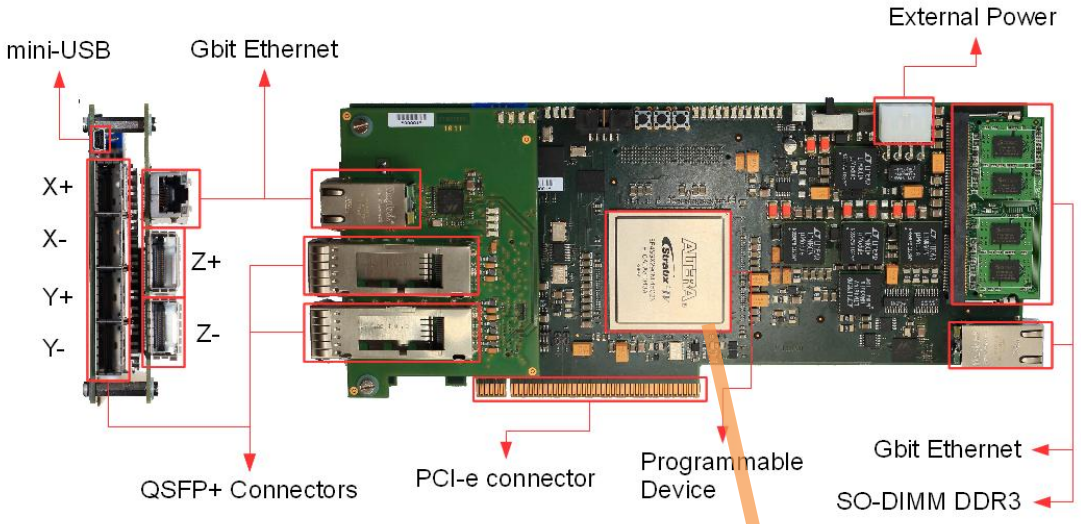
- Scalable (today up to 32K nodes)
- Direct Network: no external switches.

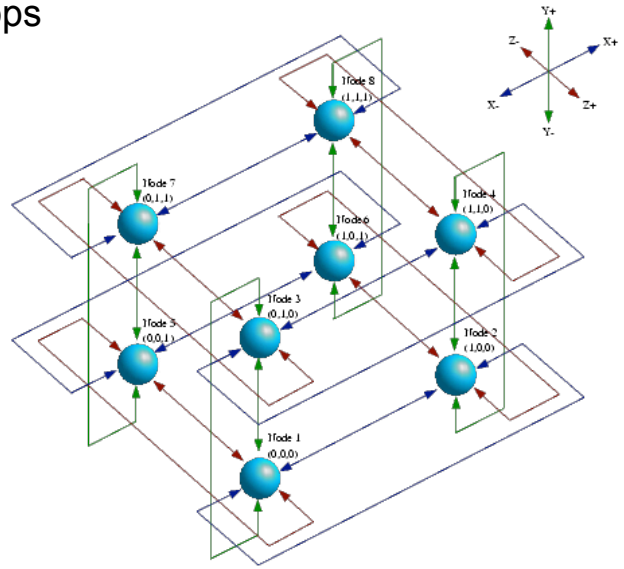
APEnet+ Card:

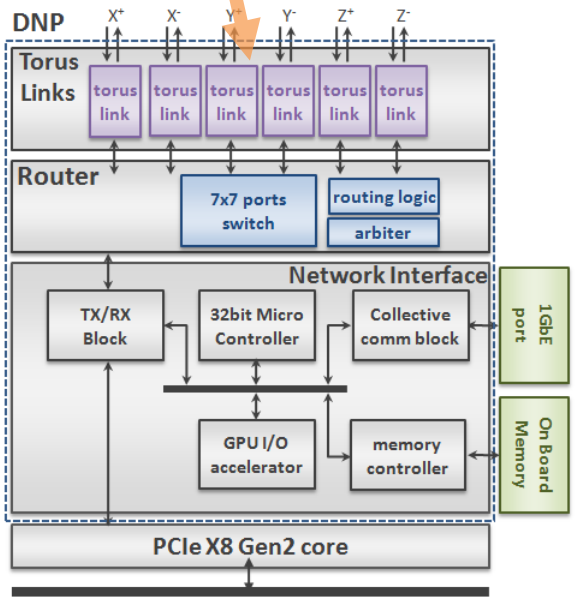
- FPGA based (ALTERA EP4SGX290)
- PCI Express x16 slot, signaling capabilities for up to dual x8 Gen2 (peak 4+4 GB/s)
- Single I/O slot width, 4 torus links, 2-d (secondary piggy-back double slot width, 6 links, 3-d torus topology)
- Fully bidirectional torus links, 34 Gbps
- Industry standard QSFP
- A DDR3 SODIMM bank

APEnet+ Logic:

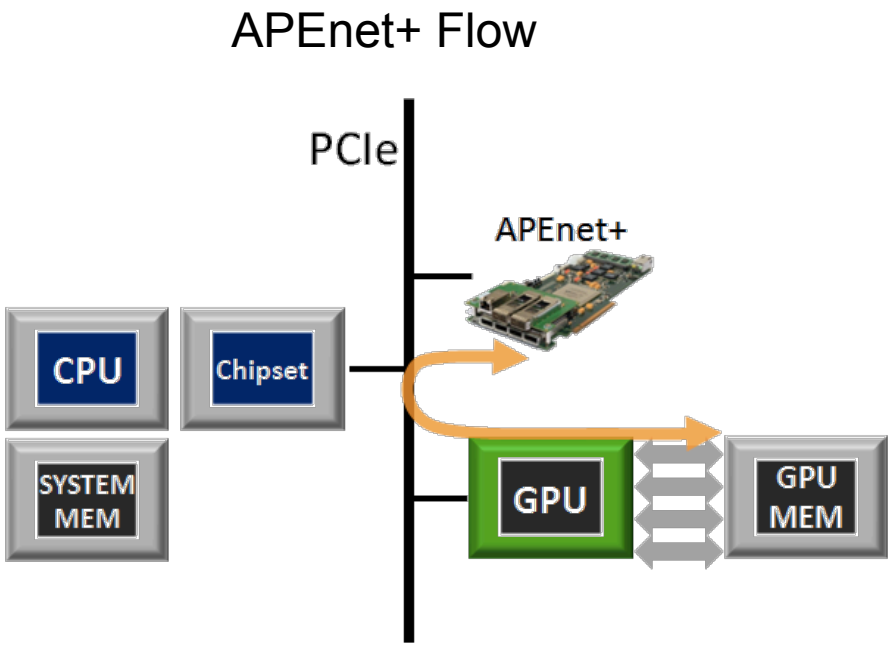
- Torus Link
- Router
- Network Interface
  - NIOS II 32 bit microcontroller
  - RDMA engine
  - GPU I/O accelerator.
- PCI x8 Gen2 Core







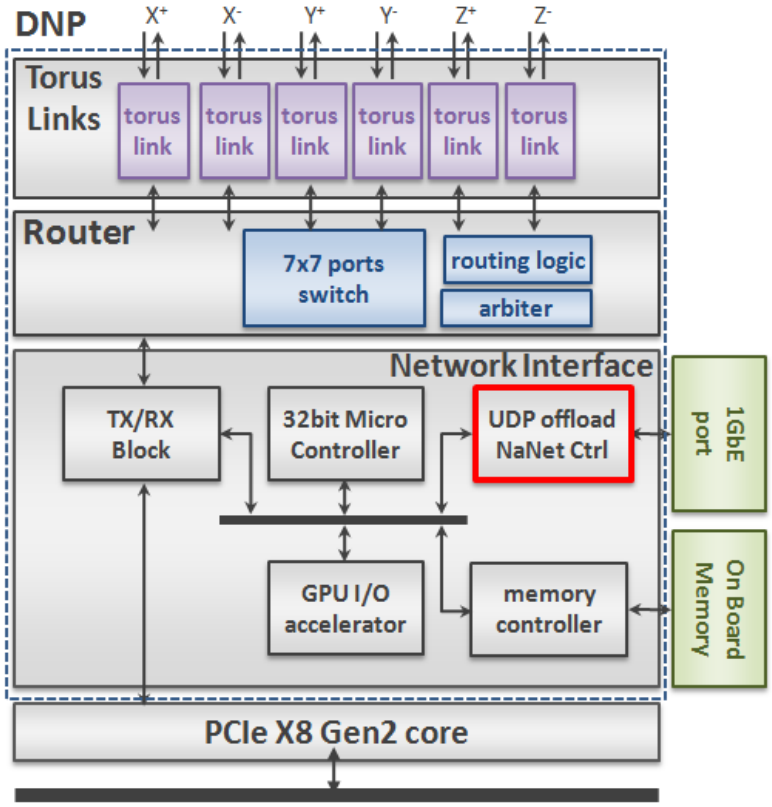
APEnet+ GPUDirect P2P Support



- PCIe P2P protocol between Nvidia Fermi/Kepler and APEnet+
  - First non-Nvidia device supporting it in **2012**.
  - Joint development with Nvidia.
  - APEnet+ board acts as a peer.
- No bounce buffers on host. APEnet+ can target GPU memory with no CPU involvement.
- GPUDirect allows direct data exchange on the PCIe bus.
- Real zero copy, inter-node GPU-to-host, host-to-GPU and GPU-to-GPU.
- Latency reduction for small messages.

NaNet Architecture

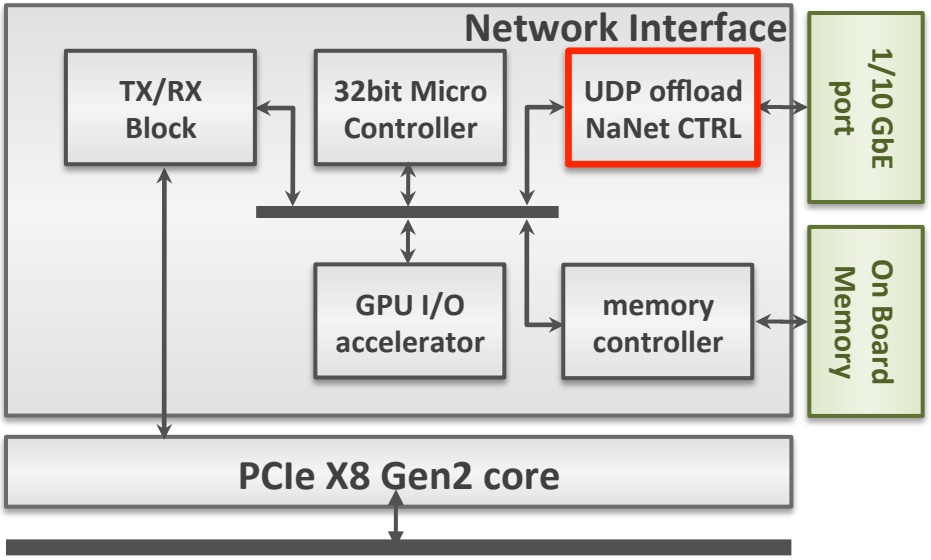
- NaNet is based on APEnet+
- Multiple link tech support
  - apelink bidir 34 Gbps
  - 1 GbE
  - 10 GbE SFP+
- New features:
  - UDP offload: extract the payload from UDP packets
  - NaNet Controller: encapsulate the UDP payload in a newly forged APEnet+ packet and send it to the RX NI logic



NaNet UDP Offloader

NiosII UDP Offload :

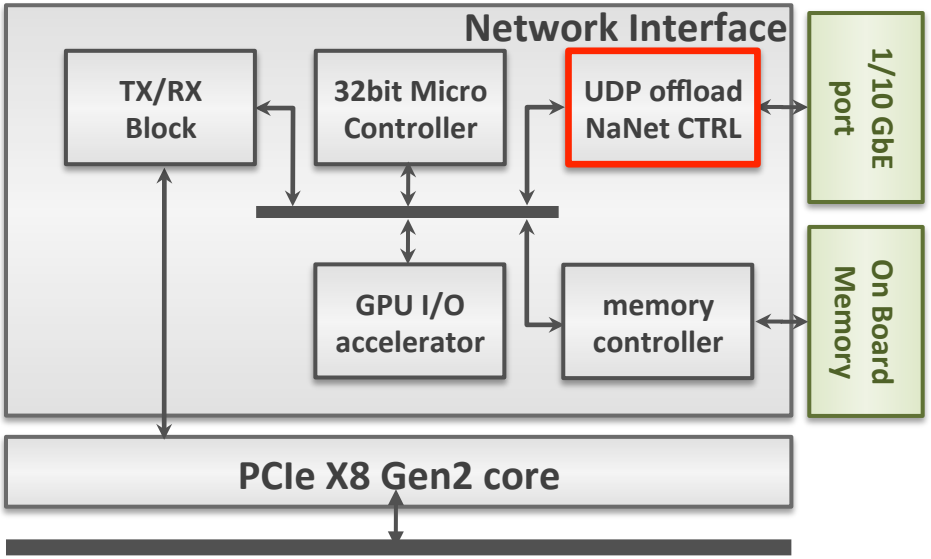
- Open IP: http://www.alterawiki.com/wiki/Nios_II_UDP_Offload_Example
- It implements a method for offloading the UDP packet traffic from the Nios II.
- It collects data coming from the Avalon Streaming Interface of the 1/10 GbE MAC.
- It redirects UDP packets into an hardware processing data path.
- Current implementation provides a single 32-bit width channel.
- 6.4 gbps (working on a 64 bit/12.8 gbps for 10 GbE)
- The output of the UDP Offload is the PRBS packet (Size + Payload)
- Ported to Altera Stratix IV EP4SGX230 (the project was based on Stratix II 2SGX90), clk@200MHz (instead of 35 MHz).



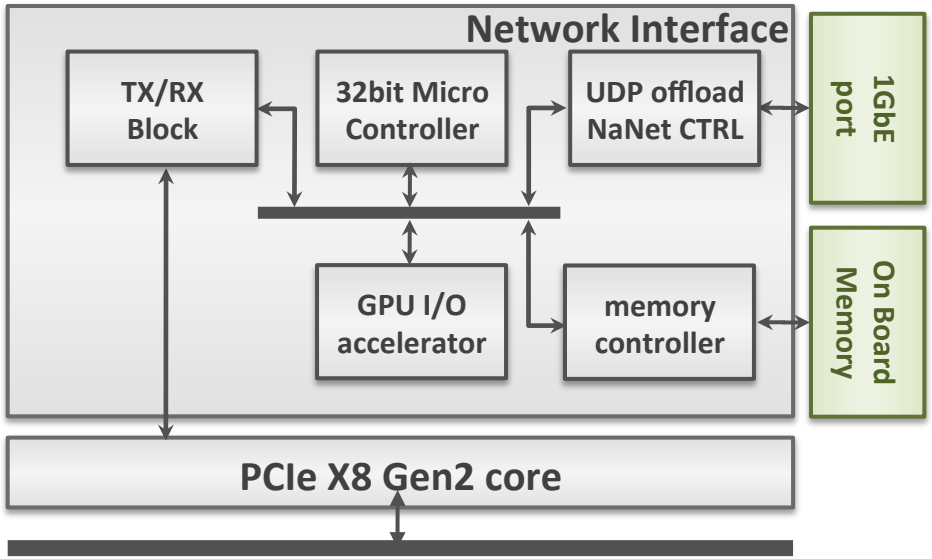
NaNet Controller

NaNet Controller:

- It manages the 1/10 GbE flow by encapsulating packets in the APEnet+ packet protocol (Header, Payload, Footer)
- It implements an Avalon Streaming Interface
- It generates the Header for the incoming data, analyzing the PRBS packet and several configuration registers.
- It parallelizes 32-bit data words coming from the Nios II subsystem into 128-bit APEnet+ data words.
- It redirects data-packets towards the corresponding FIFO (one for the Header/ Footer and another for the Payload)

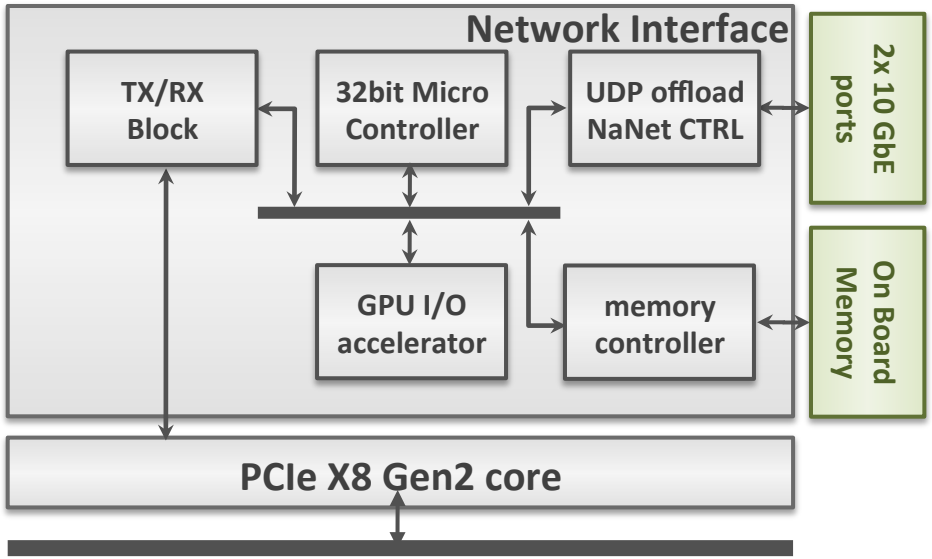


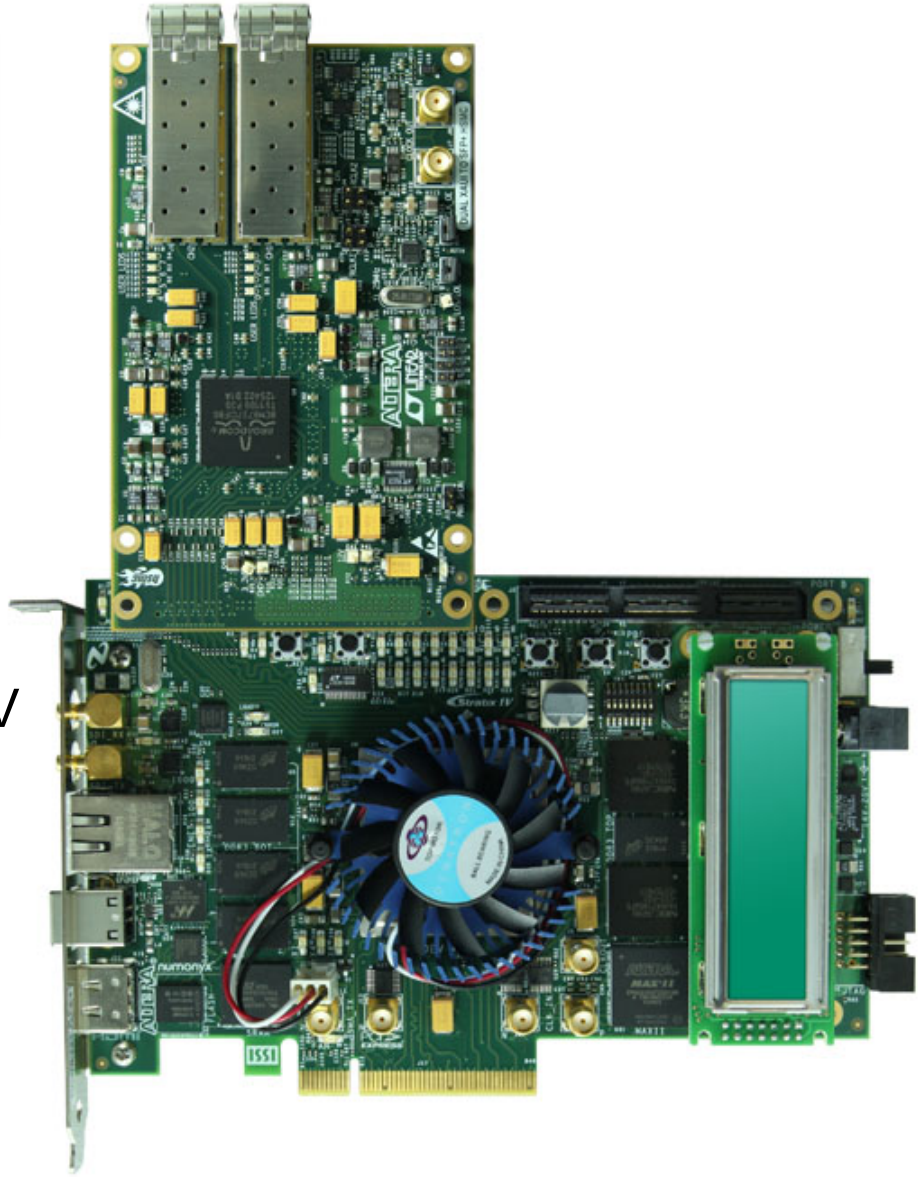
NaNet–1 (1 GbE)



- Implemented on the Altera Stratix IV dev board
- PHY Marvell 88E1111
- HAL based Nios II microcontroller firmware (no OS)
  - Configuration
  - GbE driving
  - GPU destination memory buffers management

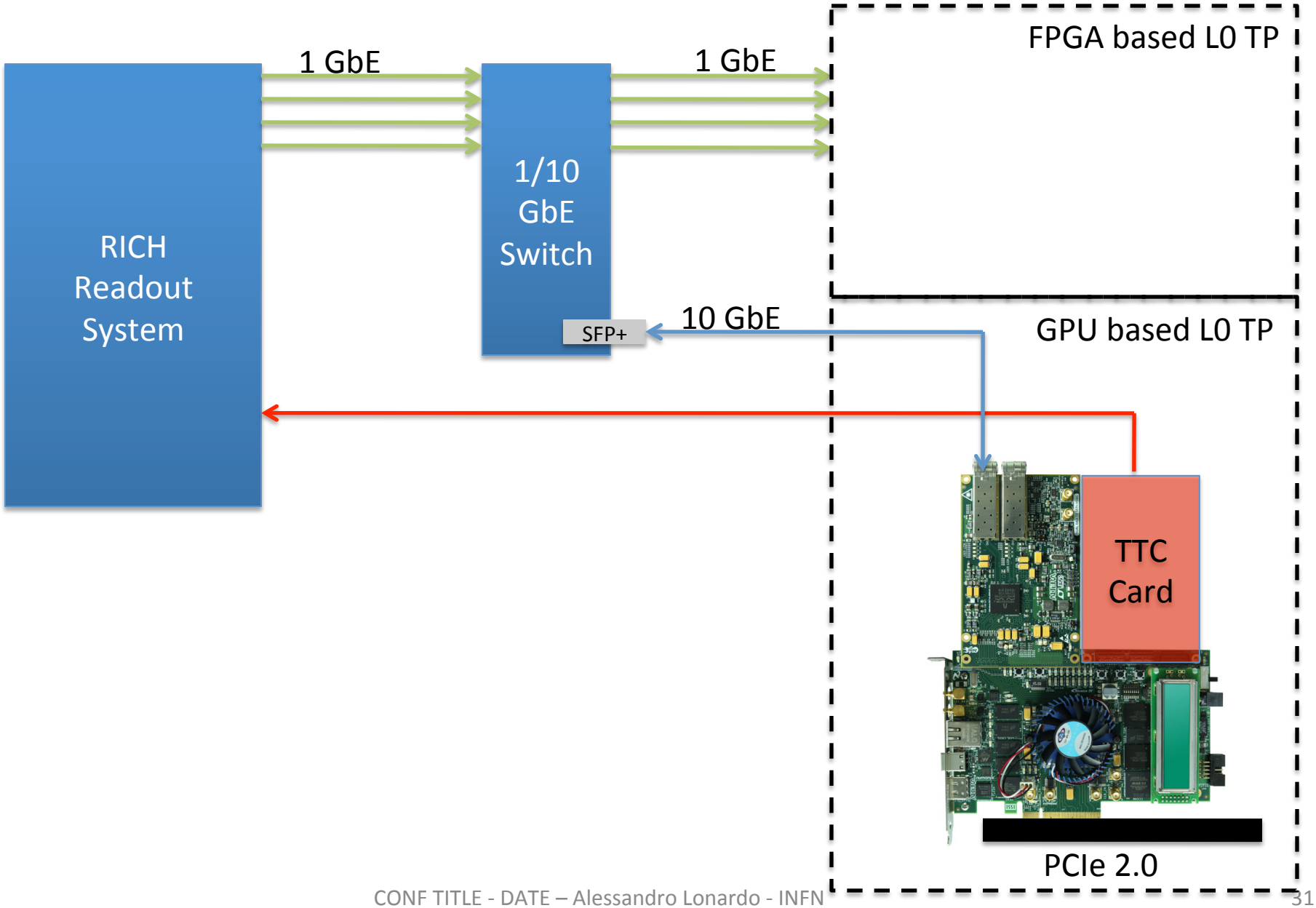
NaNet-10 (dual 10 GbE) work in progress



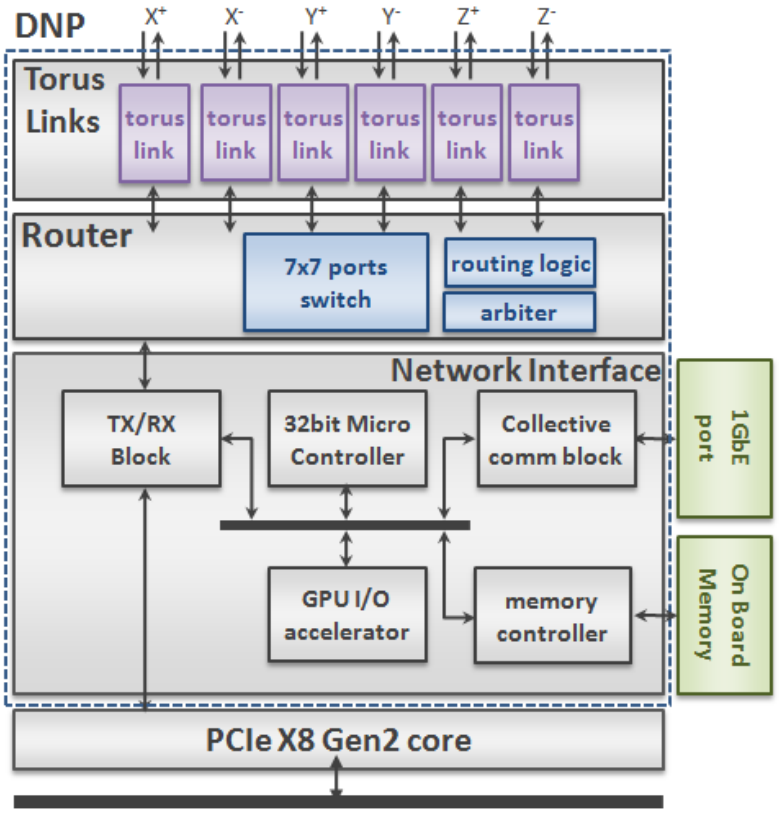


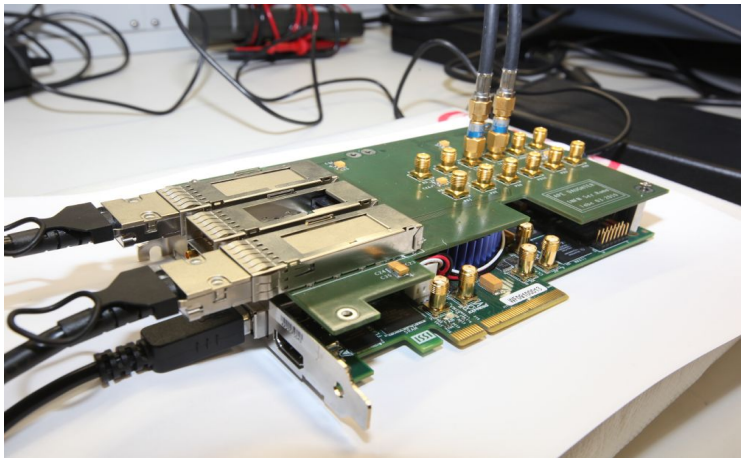
- Implemented on the Altera Stratix IV dev board + Terasic HSMC Dual XAUI to SFP+ daughtercard
- BROADCOM *BCM8727* a dual-channel 10-GbE SFI-to-XAUI transceiver

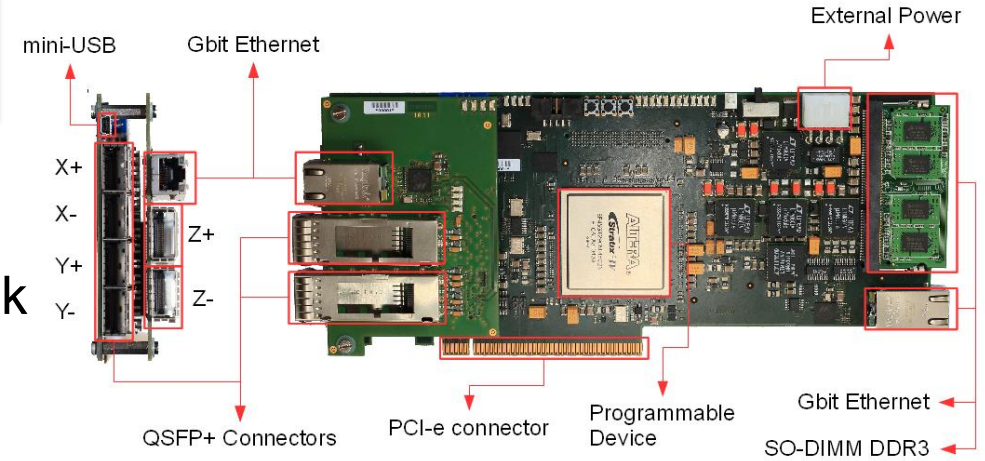
NaNet-10 for NA62 RICH L0 GPU Trigger Processor



NaNet–Apelink (up to 6 34-Gbps channels)



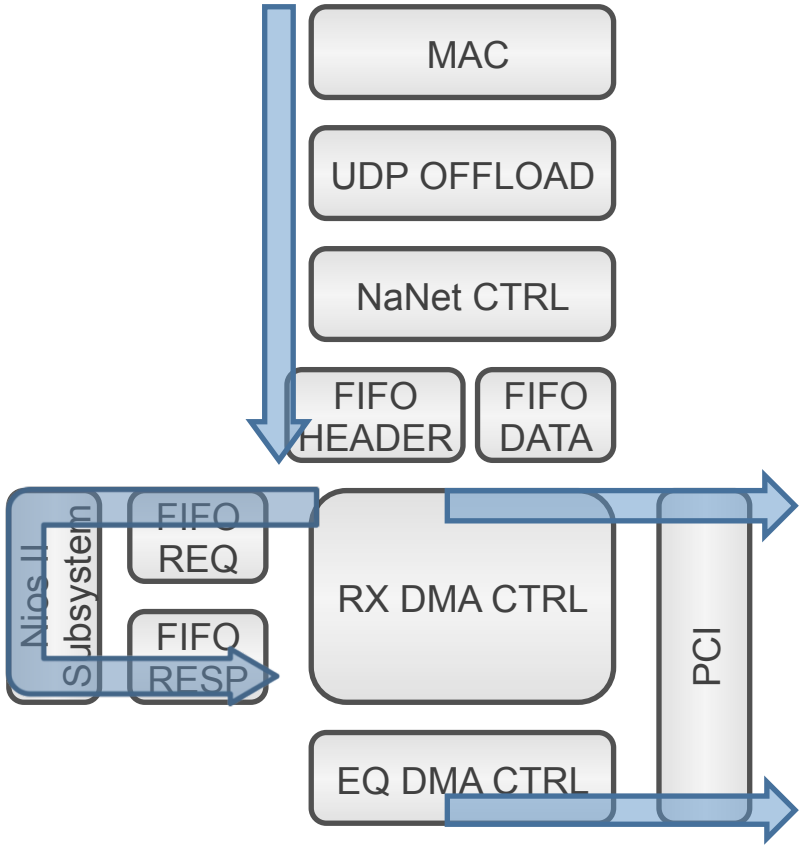




- Altera Stratix IV dev board +3 Link Daughter card
- APEnet+ 4 Link board (+ 2 Link Daughter card)

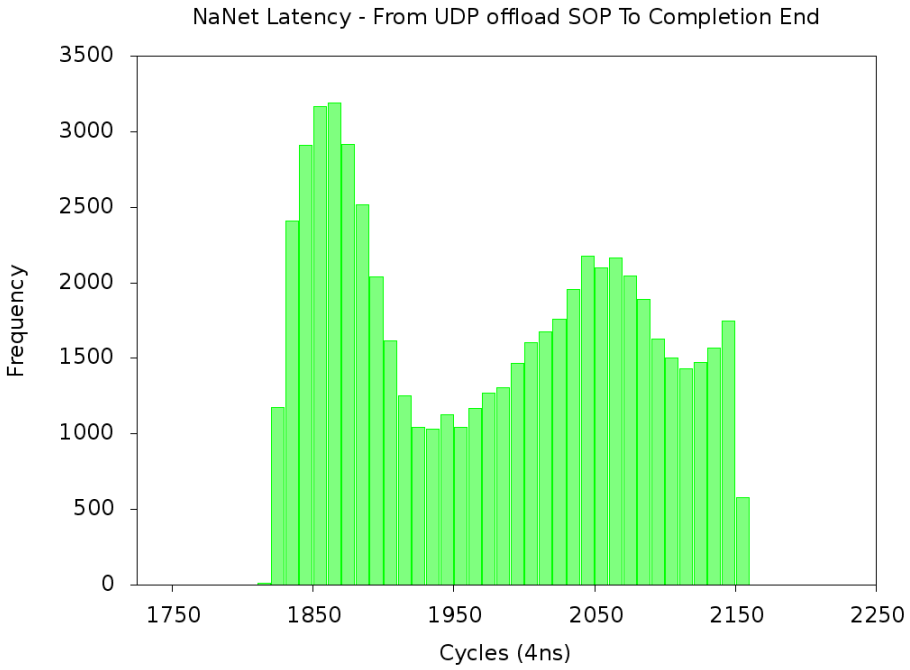
NaNet-1 Data Flow

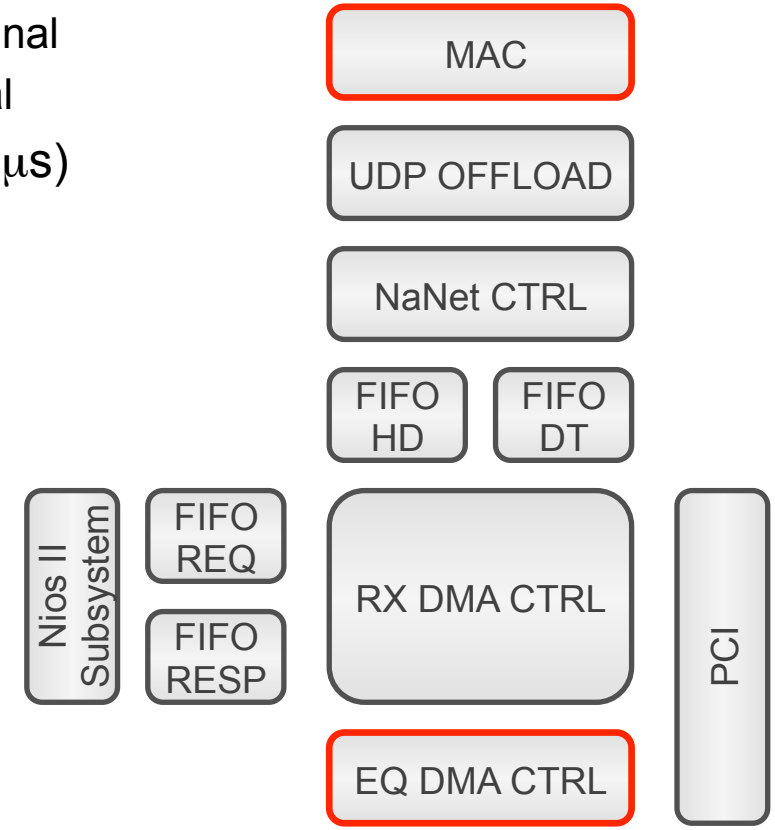
- TSE MAC, UDP OFFLOAD and NaNet CTRL manage the GbE data flow and encapsulate data in the APEnet+ protocol.
- RX DMA CTRL:
  - It manages CPU/GPU memory write process, providing hardware support for the Remote Direct Memory Access (RDMA) protocol
  - It communicates with the μC :
    - It generates the request for the Nios necessary to perform the destination buffer address generation (**FIFO REQ**)
    - It exploits the instruction collected in the **FIFO RESP** to instantiate the memory write process.
- Nios II handles all the details pertaining to buffers registered by the application to implement a zero-copy approach of the RDMA protocol (**OUT of the data stream**).
- EQ DMA CTRL generates a DMA write transfer to communicate the completion of the CPU/GPU memory write process.
- **A Performance Counter is used to analyze the latency of the GbE data flow**



UDP Packet Latency inside NaNet-1

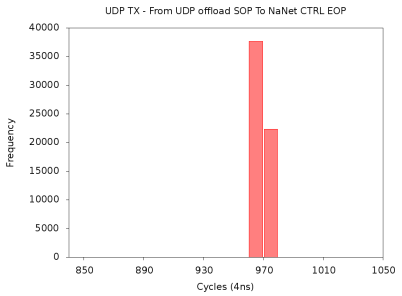
- Total Latency of a 1472 bytes UDP Packet through the hardware path:
  - From the UDP OFFLOAD Start of Packet signal
  - To the EQ DMA CTRL Completion End signal
- Appreciable latency variability (7.3 μs ÷ 8.6 μs)
- Sustained Bandwidth ~119.7 MB/s

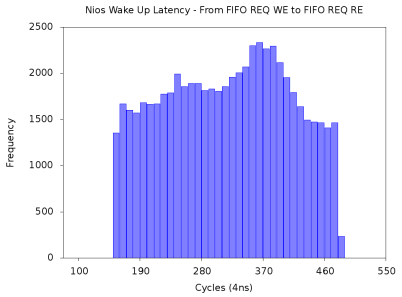


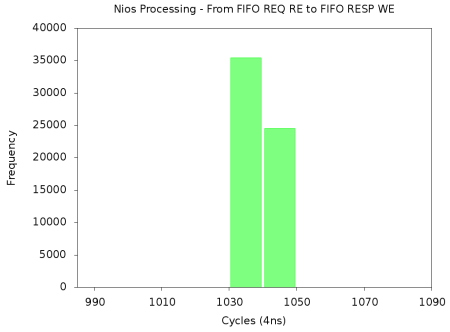


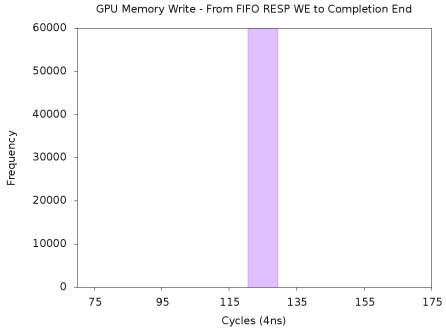
UDP Packet Latency Inside NaNet–1 Detailed Analysis

- Fine grained analysis of the UDP packet hardware path to understand the latency variability
- UDP TX (3.9 μs, stable)
  - From the UDP offload Start of Packet
  - To the NaNet CTRL End Of Packet
- Nios Wake Up (0.6 μs ÷ 2.0 μs, **culprit**)
  - From the FIFO REQ Write Enable
  - From the FIFO REQ Read Enable
- Nios Processing (4.0 μs, stable)
  - From the FIFO REQ Read Enable
  - From the FIFO RESP Write Enable
- GPU Memory Write (0.5 μs, stable)
  - From the FIFO REQ Read Enable
  - From the FIFO RESP Write Enable



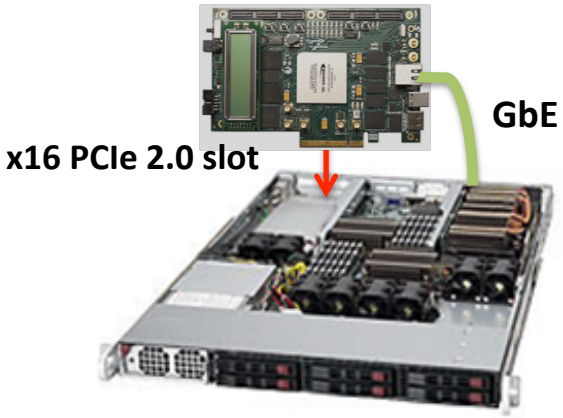






NaNet-1
Test & Benchmark Setup

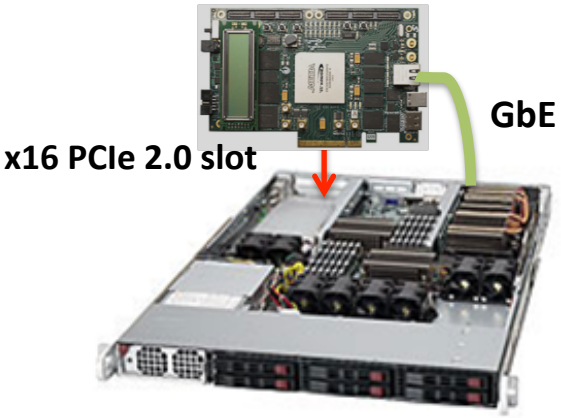
- Supermicro SuperServer 6016GT-TF
  - X8DTG-DF motherboard (Intel Tylersburg chipset)
  - dual Intel Xeon E5620
  - Intel 82576 Gigabit Network Connection
  - Nvidia Fermi S2050
  - CentOS 5.9, CUDA 4.2, Nvidia driver 310.19.
- NaNet board in x16 PCIe 2.0 slot
- NaNet GbE interface directly connected to one host GbE interface
- Common time reference between sender and receiver (they are the same host!).
- Ease data integrity tests.



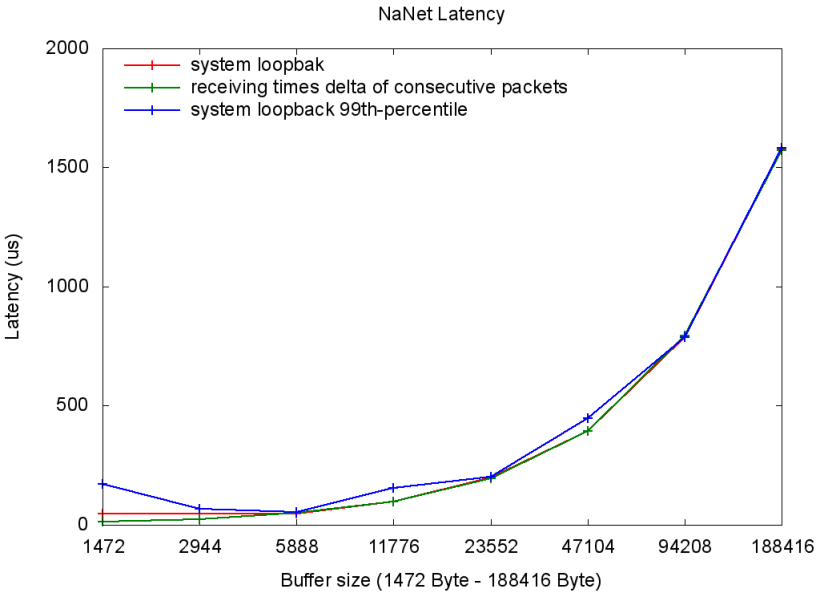
NaNet-1 Latency Benchmark

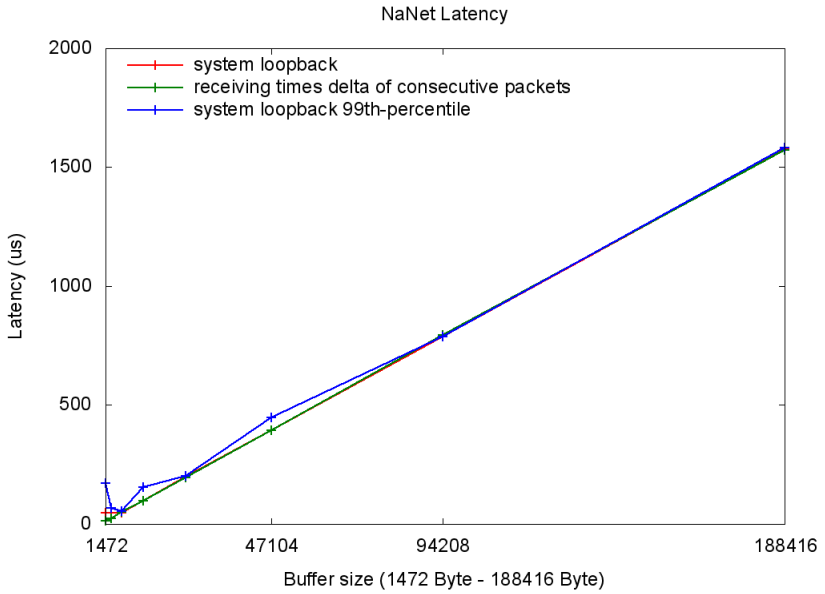
A single host program:

- Allocates and registers (pin) *N* GPU receive buffers of size *P* x1472 byte (the max UDP payload size)
- In the main loop:
  - Reads TSC cycles (*cycles_bs*)
  - Sends *P* UDP packet with 1472 byte payload size over the host GbE intf
  - Waits (no timeout) for a received buffer event
  - Reads TSC cycles (*cycles_ar*)
  - Records *latency_cycles* = *cycles_ar* - *cycles_bs*
- Dumps recorded latencies

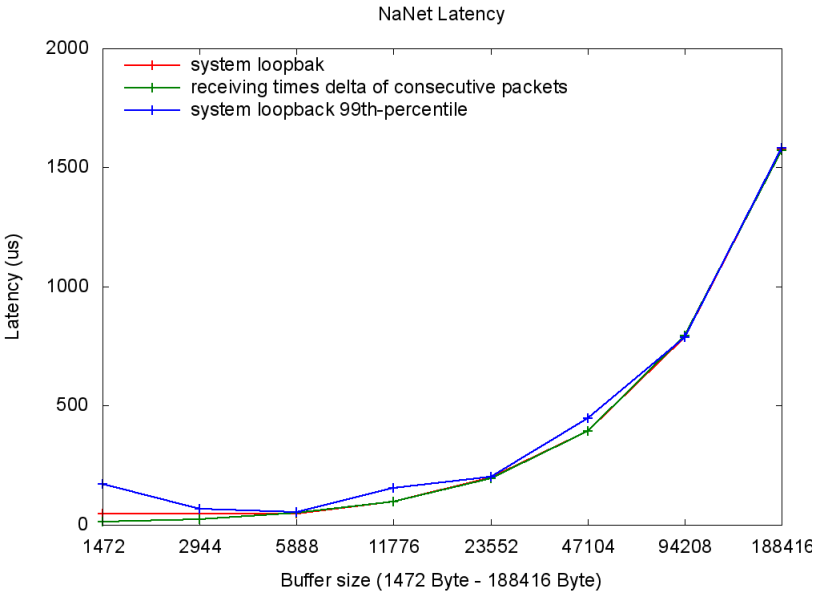


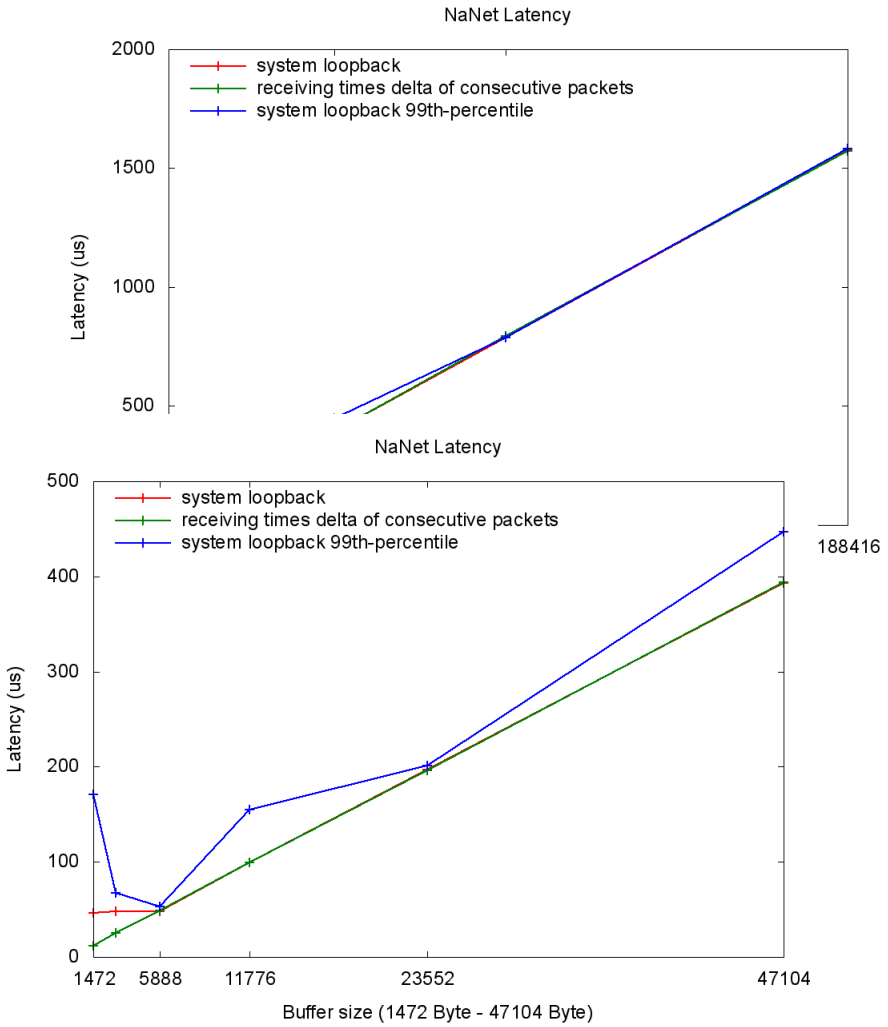
NaNet-1
Latency Benchmark



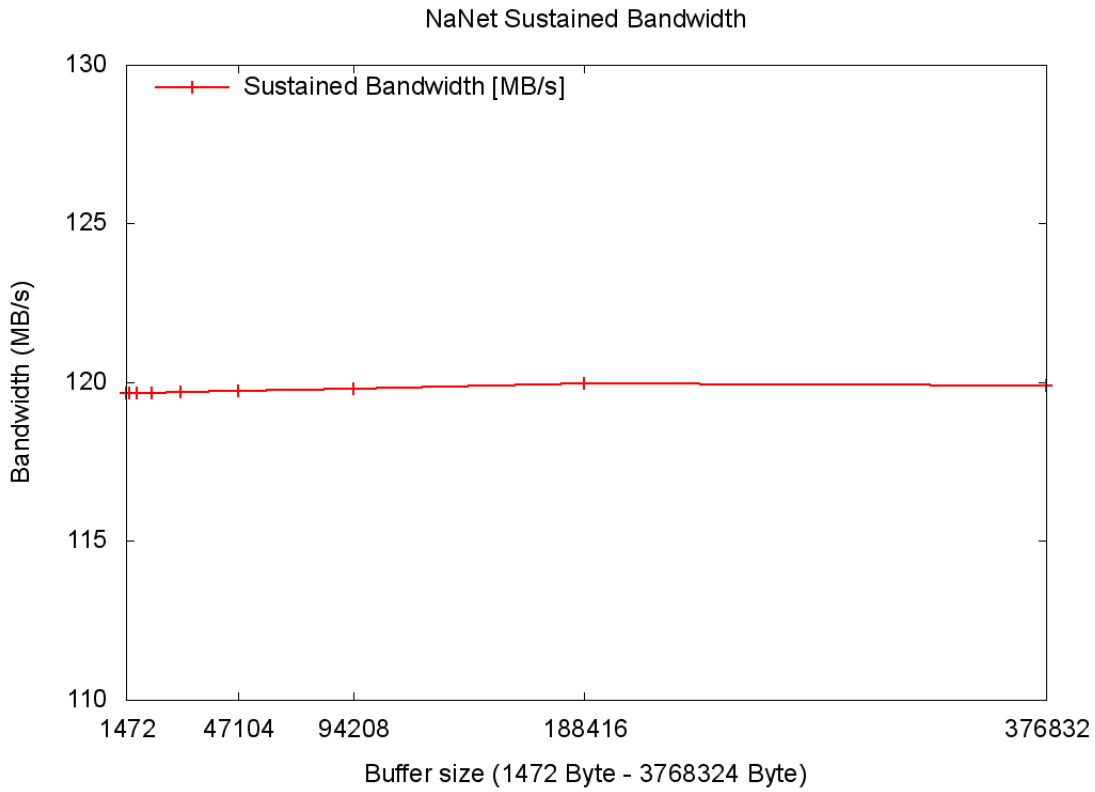


NaNet-1
Latency Benchmark



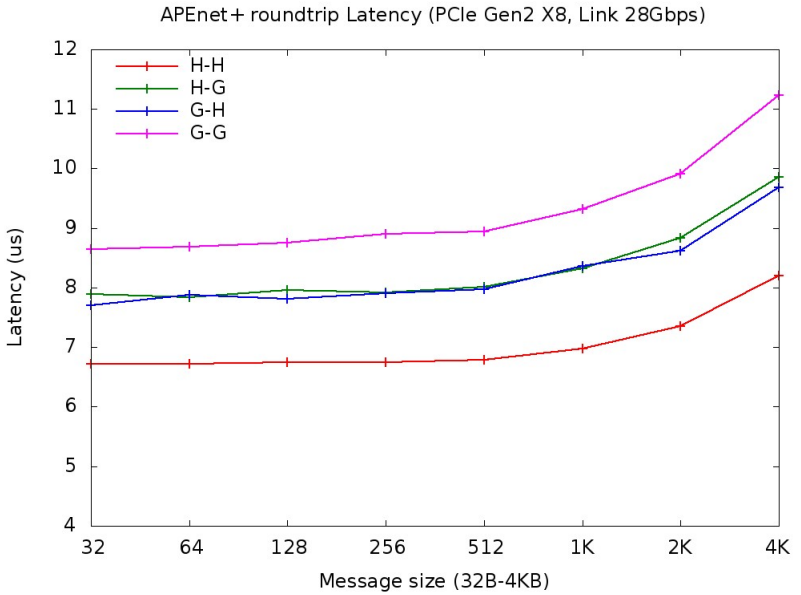


NaNet-1
Bandwidth Benchmark

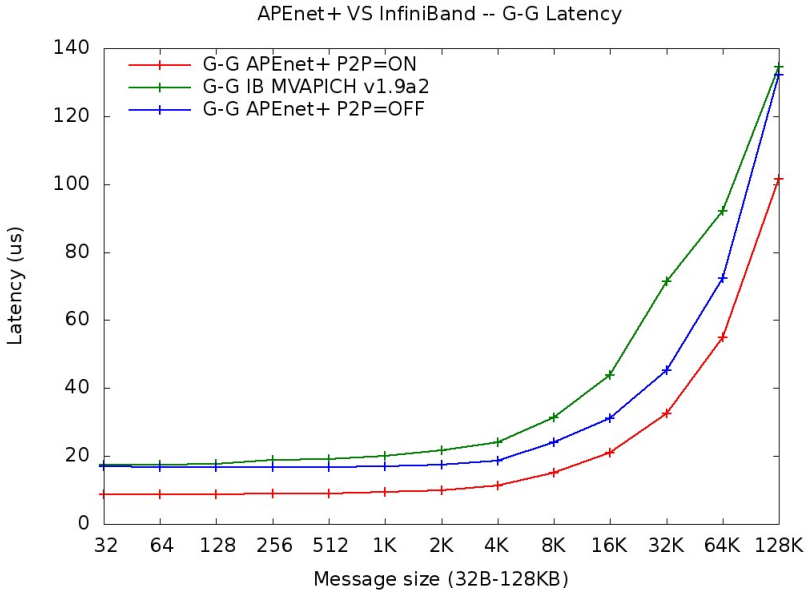


NaNet-Apelink Latency Benchmark

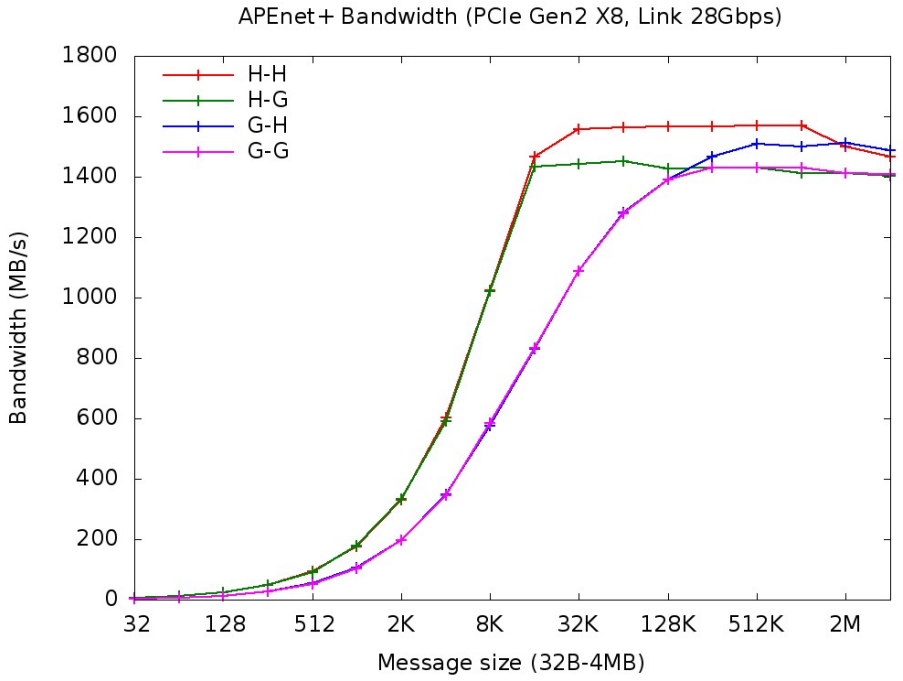
- The latency is estimated as half the round trip time in a ping-pong test
- ~ 8-10 µs on GPU-GPU test
- World record for GPU-GPU
- NaNet case represented by H-G

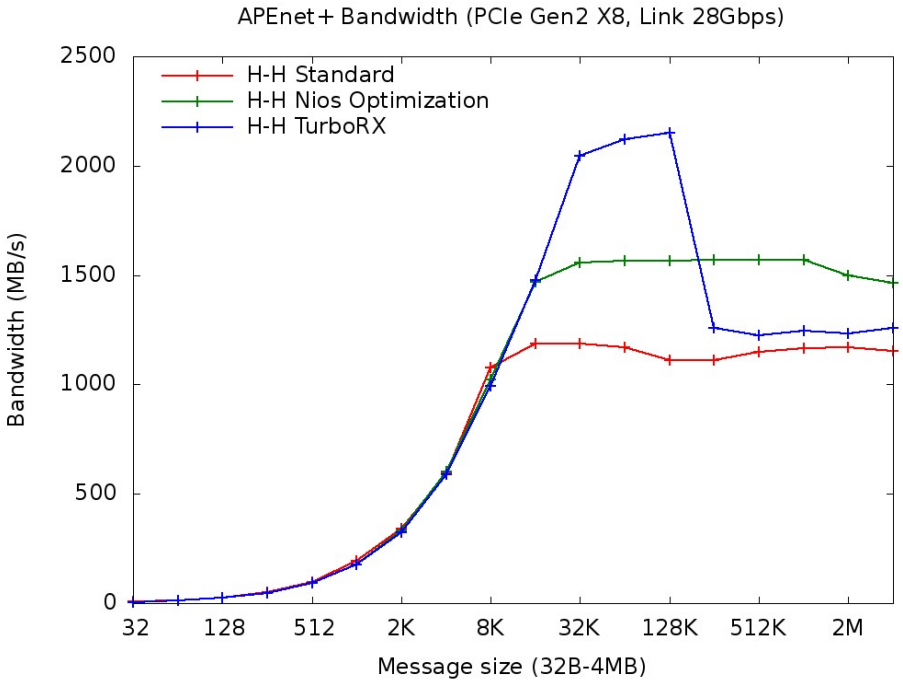


- APEnet+ G-G latency is lower than IB up to 128KB
- APEnet+ P2P latency ~8.2 µs
- APEnet+ staging latency ~16.8 µs
- MVAPICH/IB latency ~17.4 µs



NaNet-Apelink Bandwidth Benchmark





- Virtual to Physical Address Translation Implemented in the μC
  - Host RX ~ 1.6 GB/s
  - GPU RX ~ 1.4 GB/s (switching GPU P2P window before writing)
  - Limited by the RX processing
- GPU TX curves:
  - P2P read protocol overhead

- Virtual to Physical Address Translation Implemented in HW
  - Max Bandwidth **~2.2GB/s** (physical link limit, loopback test ~2.4GB/s)
  - Strong impact on FPGA memory resources! For now limited up to 128KB
  - First implementation. Host RX only!

UNICA (Unified Network Interface Card Adaptor)

- FPGA
  - Implementazione del controllo dello stack di trasmissione del protocollo
  - Pre-elaborazione dei dati consistente nella riorganizzazione degli stessi in stream idonei all'elaborazione tramite GPU
- Gestione della sincronizzazione della risposta di trigger con il clock dell'esperimento NA62 (interfaccia TTC)
  - Daughter card (connessa tramite HSMC) con chip TTCrq → realizzabile (ma occupa molti pin della FPGA se si usano tutti i segnali di controllo)
  - Alternativa (preferita ma da verificare): usare 4 connettori SMA per trasmettere in LVDS lo stream TTC da decodificare poi nella FPGA (protocollo TTC + segnali di synch)
- Ricezione time-stamp dell'evento
- La risposta della scheda UNICA può consistere nella produzione di primitive complesse passate ad un successivo livello di elaborazione oppure nella risposta di trigger finale da inviare, dopo un'opportuna compensazione della latenza di risposta, ai moduli del sistema di read-out dell'esperimento

UNICA (Unified Network Interface Card Adaptor)

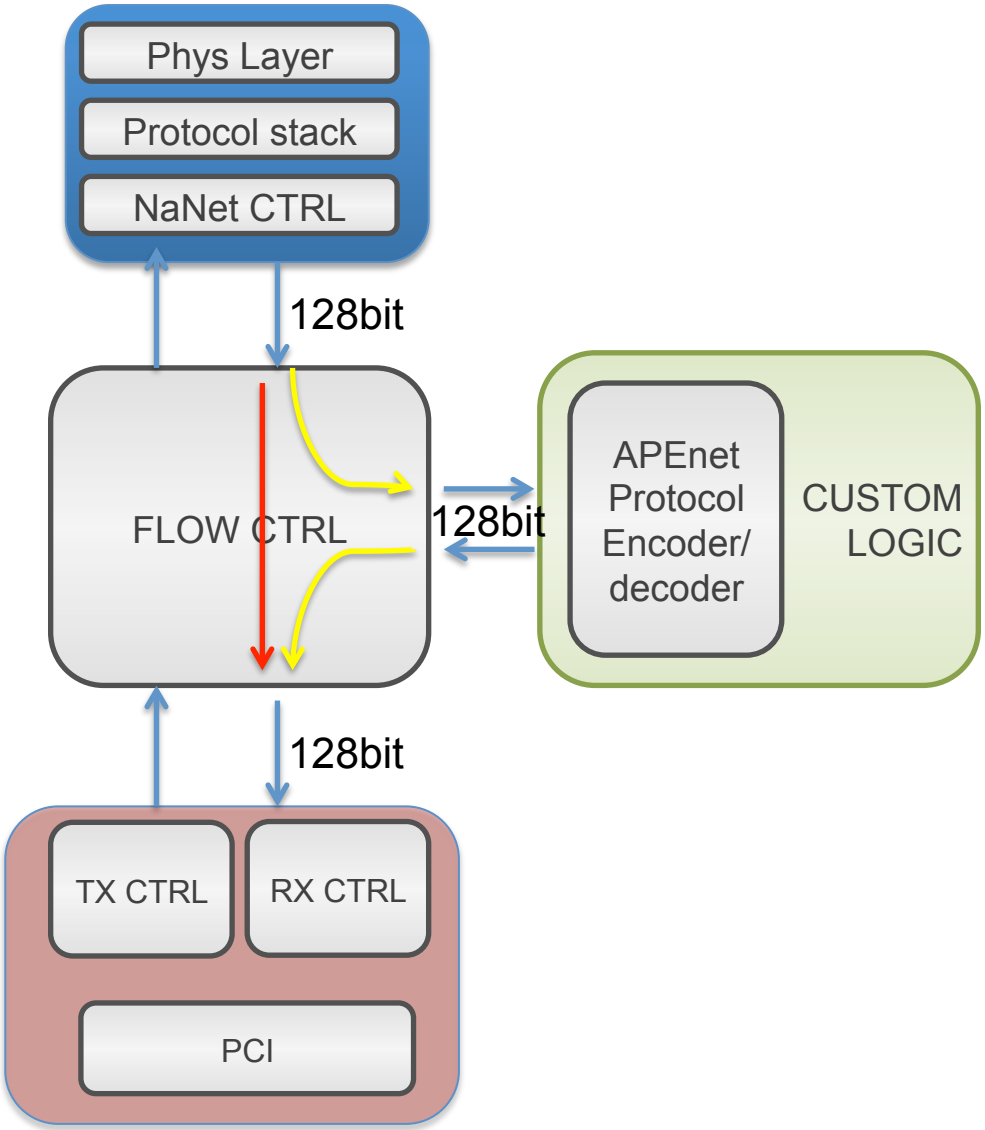
- FPGA
  - Implementazione del controllo dello stack di trasmissione del protocollo
  - Pre-elaborazione dei dati consistente nella riorganizzazione degli stessi in stream idonei all'elaborazione tramite GPU
- Gestione della sincronizzazione della risposta di trigger con il clock dell'esperimento NA62 (interfaccia TTC)
  - Daughter card (connessa tramite HSMC) con chip TTCrq → realizzabile (ma occupa molti pin della FPGA se si usano tutti i segnali di controllo)
  - Alternativa (preferita ma da verificare): usare 4 connettori SMA per trasmettere in LVDS lo stream TTC da decodificare poi nella FPGA (protocollo TTC + segnali di synch)
- Ricezione time-stamp dell'evento
- La risposta della scheda UNICA può consistere nella produzione di primitive complesse passate ad un successivo livello di elaborazione oppure nella risposta di trigger finale da inviare, dopo un'opportuna compensazione della latenza di risposta, ai moduli del sistema di read-out dell'esperimento

UNICA (Unified Network Interface Card Adaptor)

- FPGA
  - Implementazione del controllo dello stack di trasmissione del protocollo
  - Pre-elaborazione dei dati consistente nella riorganizzazione degli stessi in stream idonei all'elaborazione tramite GPU
- Gestione della sincronizzazione della risposta di trigger con il clock dell'esperimento NA62 (interfaccia TTC)
  - Daughter card (connessa tramite HSMC) con chip TTCrq → realizzabile (ma occupa molti pin della FPGA se si usano tutti i segnali di controllo)
  - Alternativa (preferita ma da verificare): usare 4 connettori SMA per trasmettere in LVDS lo stream TTC da decodificare poi nella FPGA (protocollo TTC + segnali di synch)
- Ricezione time-stamp dell'evento
- La risposta della scheda UNICA può consistere nella produzione di primitive complesse passate ad un successivo livello di elaborazione oppure nella risposta di trigger finale da inviare, dopo un'opportuna compensazione della latenza di risposta, ai moduli del sistema di read-out dell'esperimento

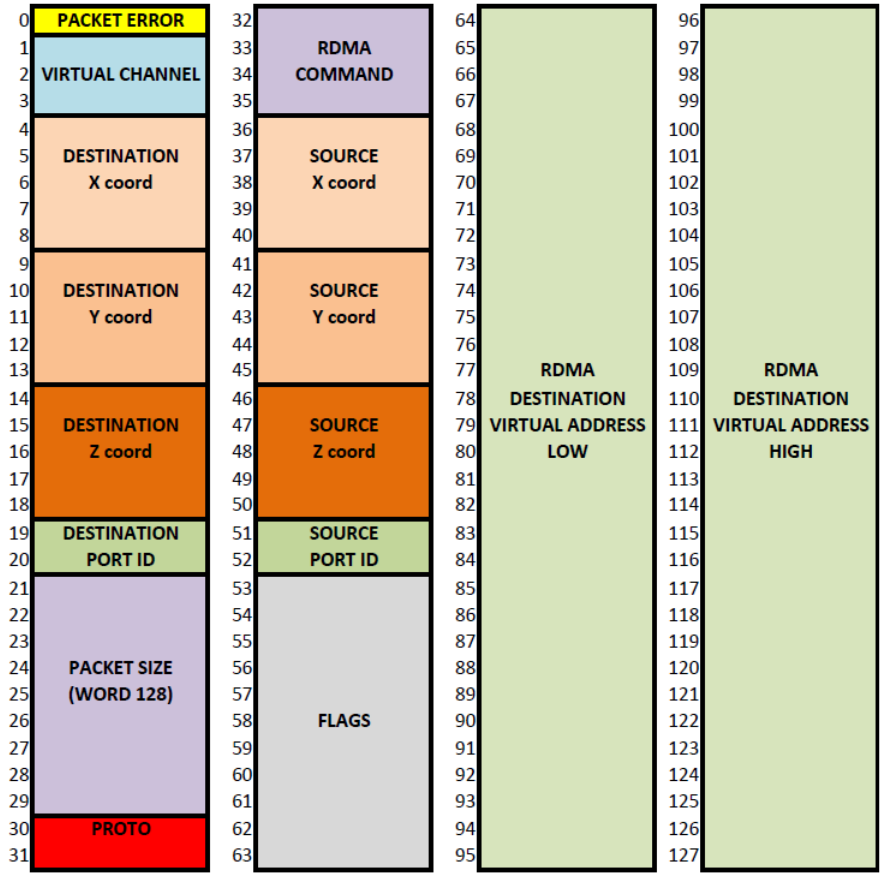
NaNet Data Flow

- NaNet CTRL manages the GbE flow by encapsulating packets in the APEnet+ packet protocol (Header, Payload, Footer)
- 128bit word
- Max payload size 4KB
- FLOW CTRL is a simple switch able to redirect the GbE flow towards the pre-elaborating data custom logic:
  - Re-use of VHDL code developed in the framework of NaNet project
  - Immediate analisys of the benefits obtained with the pre-elaboration of GPU data (Custom Logic)
  - NO additional Avalon Streaming Interface

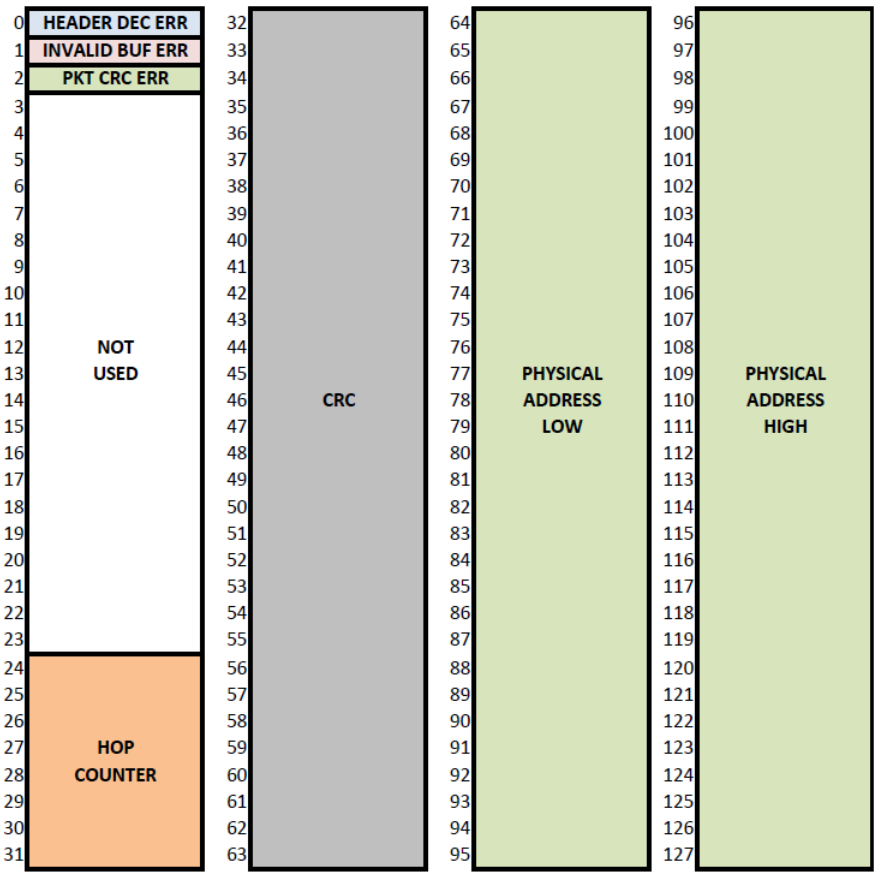


APEnet+ HEADER/FOOTER

HEADER



FOOTER



UNICA (Unified Network Interface Card Adaptor)

- FPGA
  - Implementazione del controllo dello stack di trasmissione del protocollo
  - Pre-elaborazione dei dati consistente nella riorganizzazione degli stessi in stream idonei all'elaborazione tramite GPU
- Gestione della sincronizzazione della risposta di trigger con il clock dell'esperimento NA62 (interfaccia TTC)
  - Daughter card (connessa tramite HSMC) con chip TTCrq → realizzabile (ma occupa molti pin della FPGA se si usano tutti i segnali di controllo)
  - Alternativa (preferita ma da verificare): usare 4 connettori SMA per trasmettere in LVDS lo stream TTC da decodificare poi nella FPGA (protocollo TTC + segnali di synch)
- Ricezione time-stamp dell'evento
- La risposta della scheda UNICA può consistere nella produzione di primitive complesse passate ad un successivo livello di elaborazione oppure nella risposta di trigger finale da inviare, dopo un'opportuna compensazione della latenza di risposta, ai moduli del sistema di read-out dell'esperimento

UNICA (Unified Network Interface Card Adaptor)

- TTC
  - FPGA → TTC Cosa deve comunicare la FPGA al TTC?
  - TTC → FPGA Cose riceve la FPGA dal TTC?
  - Come si interfaccia il TTC con la custom logic di pre-elaborazione?
- TRIGGER
- TRIGGER GPU
  - cosa viene trasferito dalla GPU alla FPGA


THANK YOU