Istituto Nazionale di Fisica Nucleare
Sezione di Padova

Istituto Nazionale di Fisica Nucleare
Laboratori Nazionali di Legnaro

FEROCE

Front-End Rdma Over Converged Ethernet

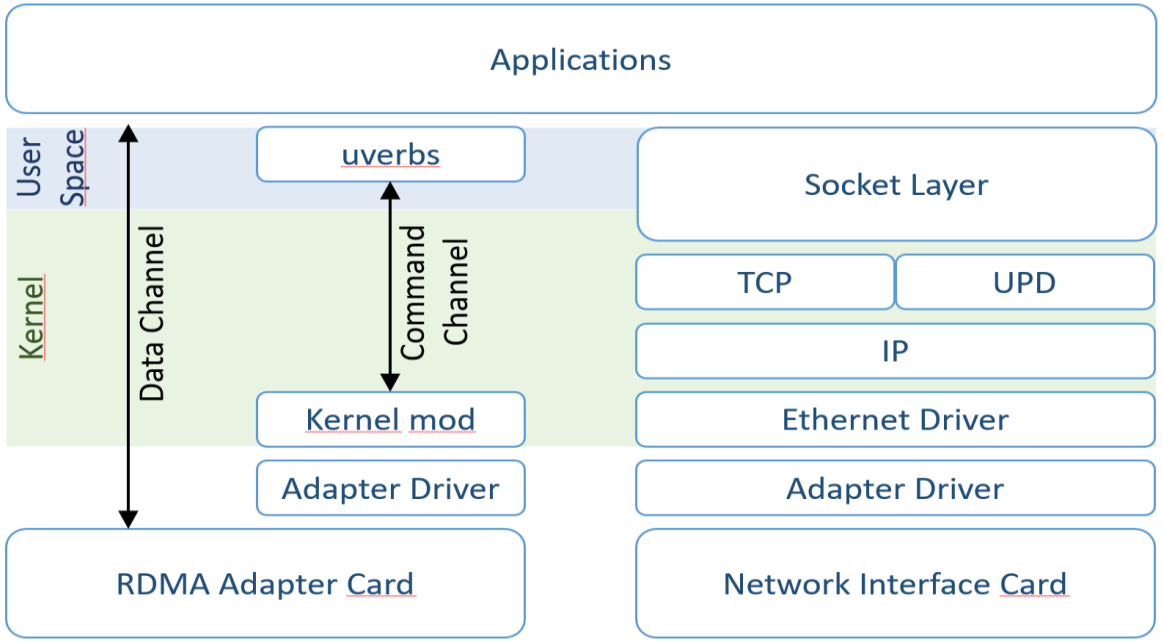
Area di ricerca: Rivelatori, Elettronica, Calcolo

Responsabile nazionale: Andrea Triossi (Unipd – INFN Padova)

Unita partecipanti: INFN sezione di Padova
Laboratori Nazionali di Legnaro

Objectives

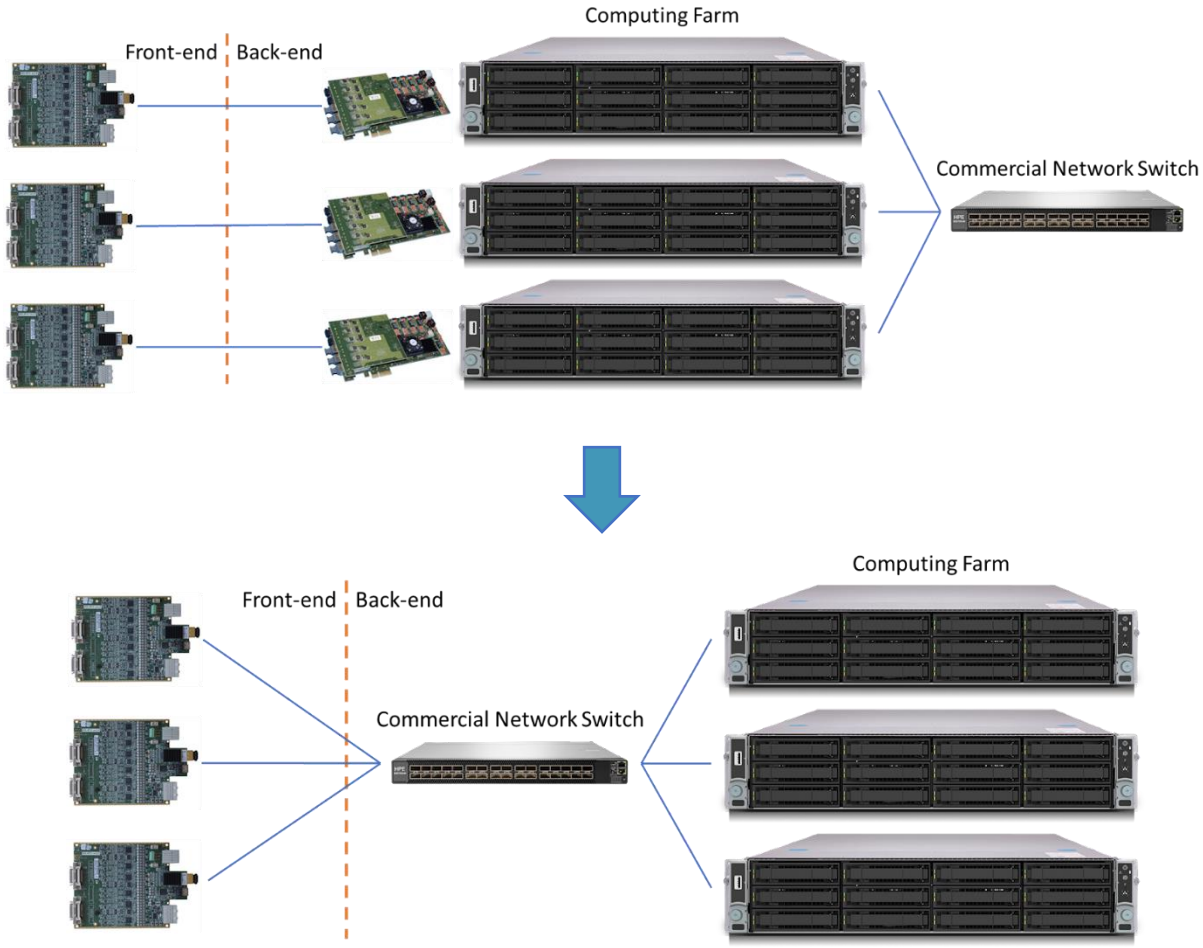
- Processing power is important as an efficient data movement
- In a DAQ system a large fraction of CPU is engaged in networking
  - Data manipulation (several copies)
  - Latency increase and throughput reduction
- Zero-copy is obtained by adding RDMA layer to the network stack



- FEROCE wants to move the adoption of the network protocol to the data producer
  - Front-end initiates the RDMA transfer
  - No point-to-point connection between front-end and back-end
  - Dynamical switching routing according to node availability

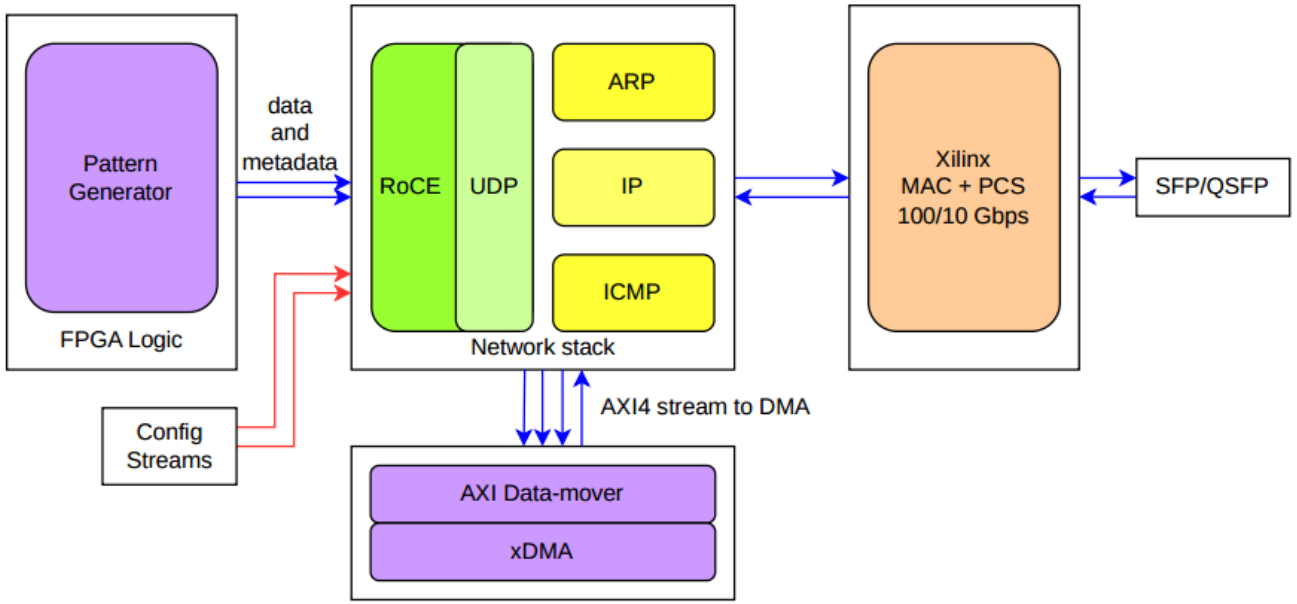
Methodology

- Several network stacks implementing RDMA
  - InfiniBand, RoCE, iWARP…
- RoCE (RDMA over Converged Ethernet)
  - Based on Ethernet networks
  - Industry-standard
  - Multi-vendor ecosystem
  - RoCE v2 packet switching (layer 2 and 3)
- FPGA are already used for implementing network stacks
  - Data center
  - ATLAS



Study of the exiting libraries

- Open-source libraries
  - ETH Zurich Network Stack
    - Entirely written in HLS
    - 10/100 Gbps via Xilinx 10G and 100G MAC IPs
    - xDMA, DDR4 memory and recently HBM support



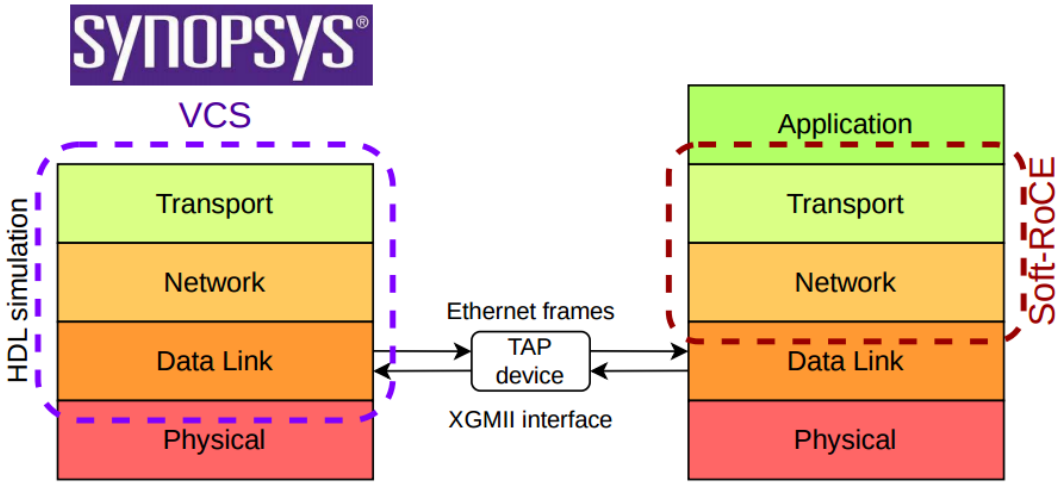
- This module writes/reads data to/from the Host machine's memory through the xDMA
- An AXI data-mover is used to translate the AXI4 stream into AXI memory mapped (and viceversa)
- Queue Pair's information is exchanged using AXI stream ports
- A debug port is present to send data directly from the FPGA logic TX DATA port

Example RDMA WRITE from FPGA

- Generate the Queue Pair parameter from the receiver side and allocate memory
  - Remote QP number
  - Remote Virtual Address
  - Registration key (to access the memory)
  - Remote Packet Sequence Number (PSN) needed to check stream's structure
- Send such parameter to FPGA on a sideband channel (e.g. ipbus/TCP-IP)
- Pack the parameters in the configuration AXI streams ports
- Stream data through the data AXI stream port
- Notify somehow the receiver that the transfer is completed (Immediate message/sideband)
- Dump the buffer

Testing of UDP, TCP and RoCE V2

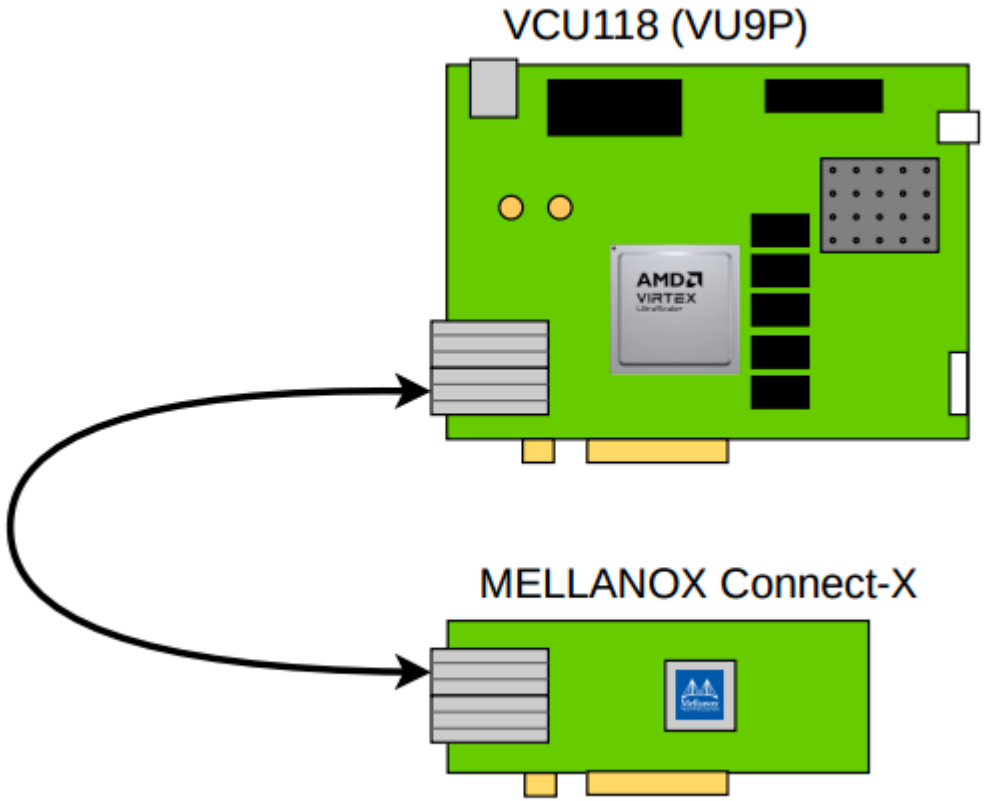
- Dynamic VCS simulation
  - TUN/TAP
  - DPI-C (Direct Programming Interface) used to link C functions with RTL simulation
  - Modifications in MAC and in DDR interface
  - RDMA WRITE tested successfully with a 10G MAC
  - RoCE firmware simulator sends data to Soft-RoCE end-point



- Results presented at two international conferences TIPP and TWEPP
  - Proceeding here

Hardware implementation

- UDP and TCP stacks deployment on VCU118
  - Good bandwidth results for TCP (> 60Gbps with two connections)
  - Hard to close the timing (even in HLS doesn't close)
  - Porting to Vivado 22.2 was needed (intelligent runs)
- RoCE v2 stacks deployment on VCU118
  - Several issues were found before successfully transmit data (also present in simulation)
    - New CRC32 module developed (RTL, not HLS) see backup slides for details
  - Occupancy:

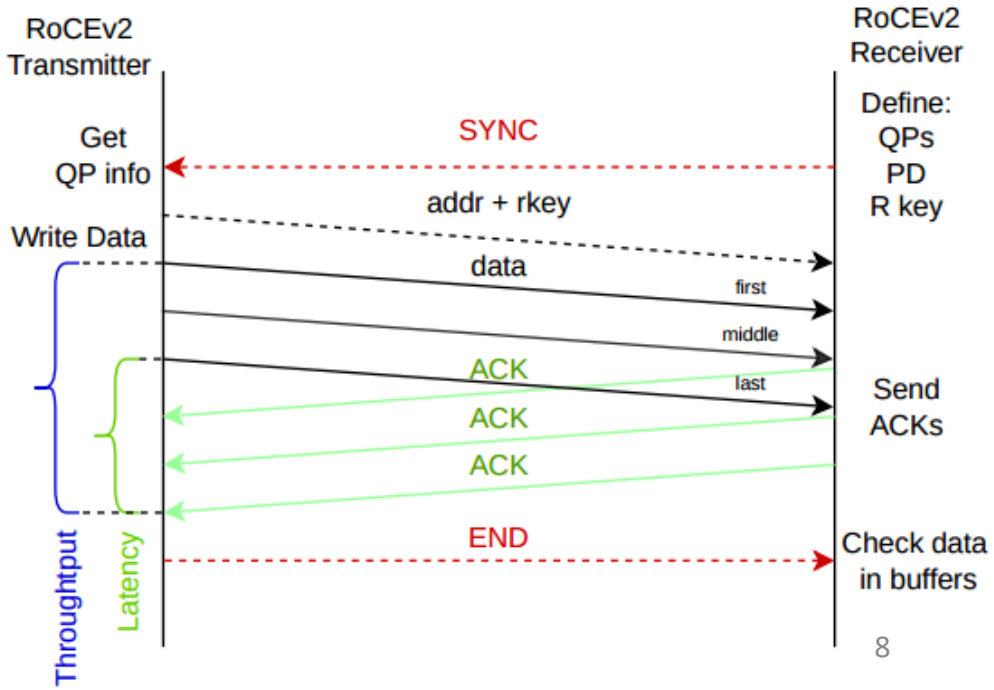


RoCEv2 (no retrans)			TX only estimate (no retrans)		
LUT [k]	FF [k]	BRAM	LUT [k]	FF [k]	BRAM
38	49	93	~ 15	~ 15	~ 30

Latency and throughput

- Latency is computed averaging the time from each packet's transfer start and its acknowledge (round trip)
- Throughput is computed considering the payload size and the time from first packet sent and last acknowledge received
- First transfer in the QP must be small (warm-up), it's needed to for caching the QP info (NIC side), otherwise packet loss is observed
- After that, no packet loss is observed, need more tests though

Transfer size	Throughput [Gbps]	Latency [μs]
8 kB	30.7	1.6
32 kB	62.8	2.3
64 kB	73.3	3.1
128 kB	80.5	3.3
512 kB	92.9	12.4
5.12 MB	95.8	12.4
51.2 MB	96.0	12.4
512 MB	96.0	12.4

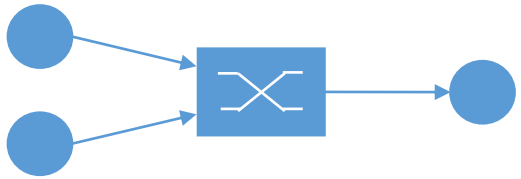


Networking activities

- Learned to use ibverbs for writing RDMA capable applications
  - Building testing environment, based on two virtual machines with soft-roce
  - First communication test between two applications, using rdma connection manager (RDMA CM) for connection handling
    - RDMA write from one VM memory to the other
  - CM removal, manual handling of the connections and out-of-band communication
  - Removal of one of the VM, substitution with dynamic simulation
- Verified the same operations also with real hardware
  - Test between two servers equipped with Connect-X 6, the same application works
    - Point-to-point connection
  - linux-rdma/perftest package used for performance measurement
    - For instance, `ib_write_bw` used for measuring the bandwidth of an RDMA WRITE
    - Test of several configuration of message size and number of messages
    - Measured up to 93 Gbps
  - Tuning of the host parameters (e.g. NIC Buffer, MTU, etc.)

Ongoing efforts

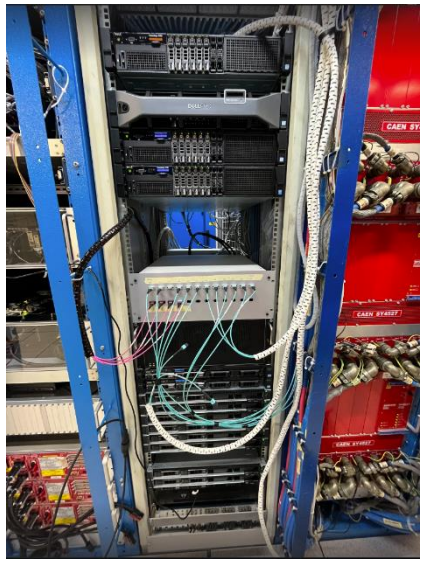
- Congestion tests on a 10G network
  - Two senders / one receiver link saturation
  - Flow Control or RoceV2 congestion management throttle the sender without triggering retransmission
  - Next steps
    - Repeat tests on a 100G network (Nvidia switch received)



- Rewrite RoCE TX module at RTL, only for RDMA WRITE (with immediate)
  - Open-source network stack for low level layers (UDP, IP, ARP…)
  - Tested in dynamic simulation towards soft-ROCE
  - GitHub code
  - Talk accepted at CHEP 2024
  - Next steps
    - Add re-transmission module (depending on 100G congestion test)
    - Rewrite RoCE RX module at RTL, to decode ACK/NACK/CNP packets
    - Porting of the ROCE network stack on Microchip FPGA (evaluation board received)

Prospective applications

- CMS (CSN1)
  - L1T Scouting is a project aiming at acquiring the L1 primitives at the full bunch crossing rate
  - It is meant for HL-LHC but a demonstrator based on commercial electronic is already deployed
  - At present as DAQ link it adopts a light version of the TCP/IP protocol at 100G



Research group

- Padova
  - Marco Bellato (0.1)
  - Antonio Bergnoli (0.2)
  - Gabriele Bortolato (0.3)
  - Daniele Mengoni (0.15)
  - Matteo Migliorini (0.3)
  - Fabio Montecassiano (0.2)
  - Jacopo Pazzini (0.15)
  - Andrea Triossi (0.3)
  - Marco Zanetti (0.1)
- LNL
  - Damiano Bortolato (0.15)

Total FTE: 1.95

Project organization

- Two working package
  - WP1 Front-end firmware core
  - WP2 Application layer and emulation
- Four milestones

DONE	~~31 Dec 2023~~	M1.1	Setting up of a global-oriented simulation environment for a small appliance of the RoCE stack (WP1)
DONE	~~31 Dec 2023~~	M1.2	Testing a RoCE network based on COTS products (WP2)
ON TRACK	31 Dec 2024	M2	Complete test of the scalable RoCE firmware stack for front-end (WP1+WP2)
	31 Dec 2025	M3	Complete test of the developed firmware on a radiation tolerant front-end FPGA (WP1+WP2)

		2023			2024			2025		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
WP1	**Front-end firmware core**	Development light RoCE		Simulation light RoCE	Implementing different data paths		Test FE RoCE on network	Porting to flash technology		Test on flash FPGA
WP2	**Application layer and emulation**	Soft RoCE	Setting up RoCE	Test of RoCE network	Acquisition system		Test FE RoCE on network	GPUdirect	Test of GPUdirect	Test on flash FPGA

Richiesta finanziaria

Anno	Missioni	Inventariabile	Consumo	Totale
2025	**1.5**	**10**		**11.5**

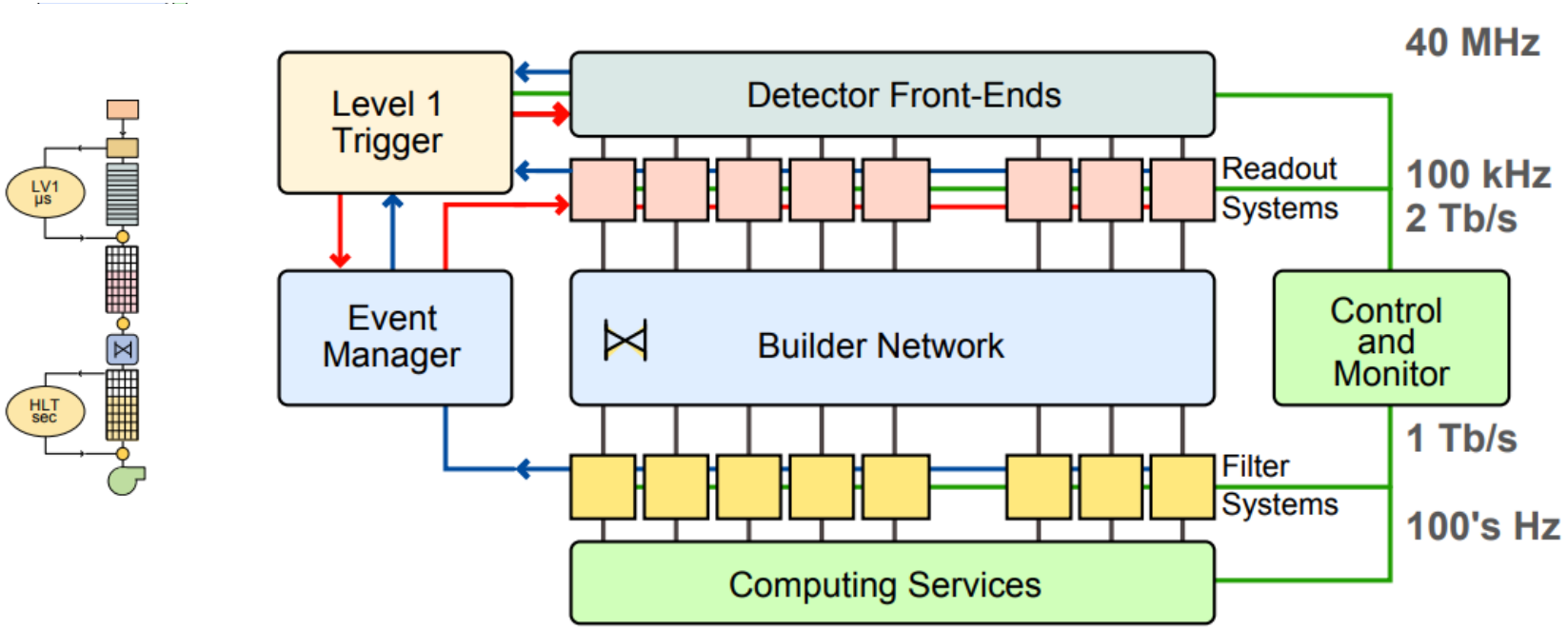
La parte di apparati è dettagliata come segue:

2025 1x GPU NVIDIA 10kE

Backup

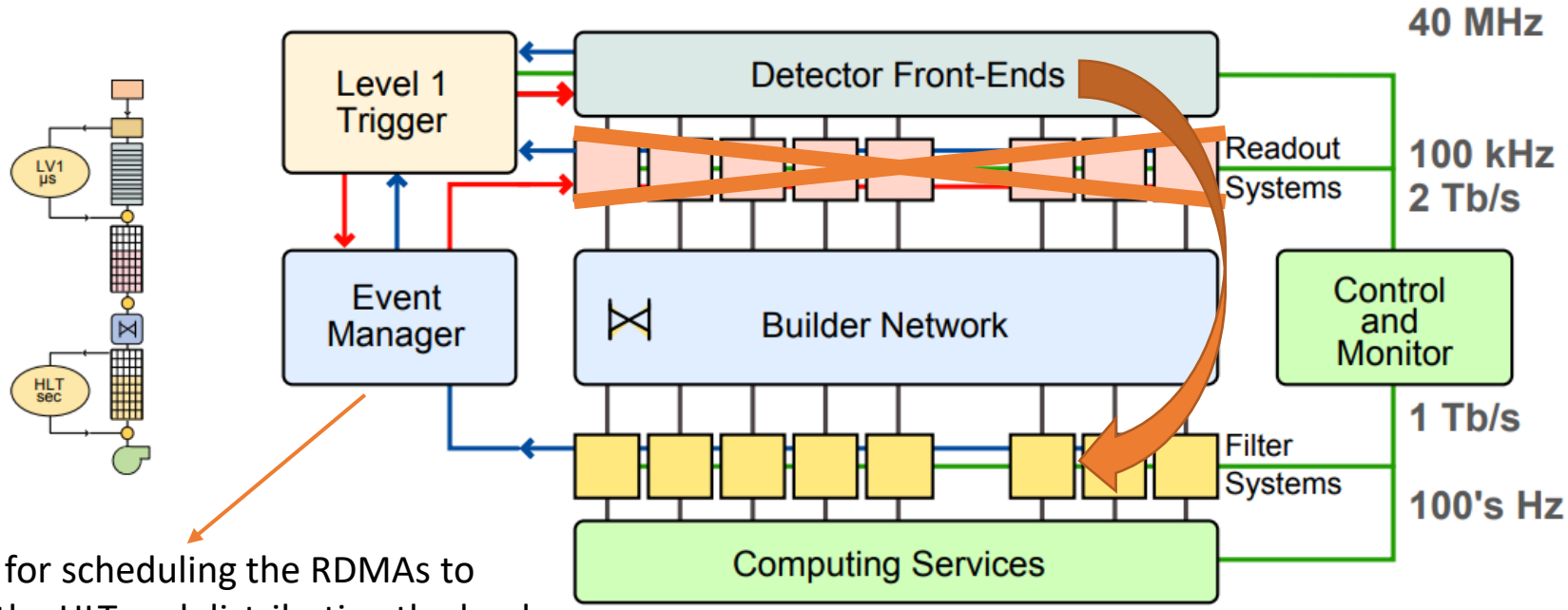
Methodology

- For instance, two levels CMS TDAQ system
  - Several layers of hardware (FE, BE, Read out units, Event manager, Filter units)



Methodology

- RDMA+Ethernet (RoCE) would simplify
  - Cost reduction -> no need of custom back-end electronics
  - Scalability -> add commercial switch
  - Commercial standard -> inherit of low-level software



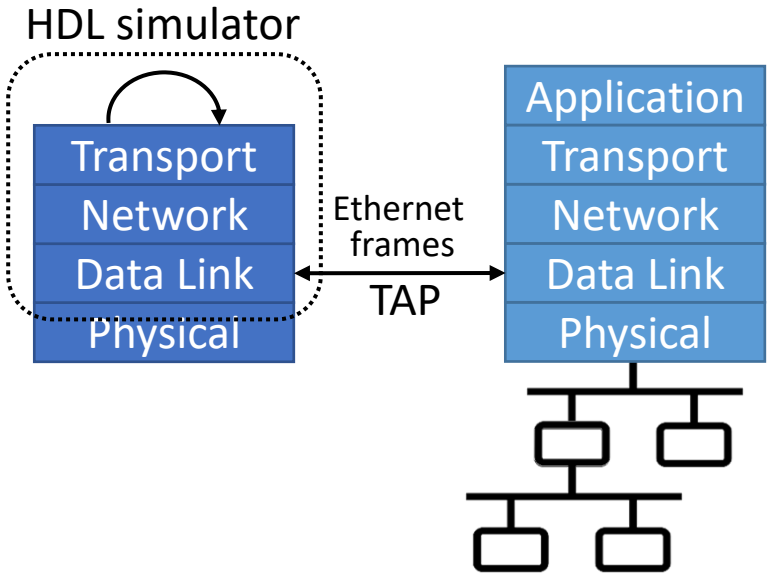
Event Manager remains for scheduling the RDMAs to the available servers of the HLT and distributing the load

FW activities

- Commercial RoCE cores don’t fulfill our requirements
  - Large footprint -> mono direction, only one transfer type (send)
  - Large memory for retransmission -> private network, reliable UDP
- Development of the firmware block needed by a RoCE endpoint
  - Small appliance of RoCE stack
    - Small scale FPGAs
    - No external memory or soft-core processor
  - Setup a global-oriented simulation environment
    - Language heterogeneity
    - Dynamic simulation (next slide)
  - Test of the RoCE stack on evaluation platform
  - Capability of scale down the datapath
    - 100GbE/10GbE/1GbE
    - Many small distributed endpoints
    - Commercial switch port aggregation
  - Porting the front-end RoCE core to flash-based FPGA (radiation environment)
    - No technology dependent IPs or proprietary compilation tools

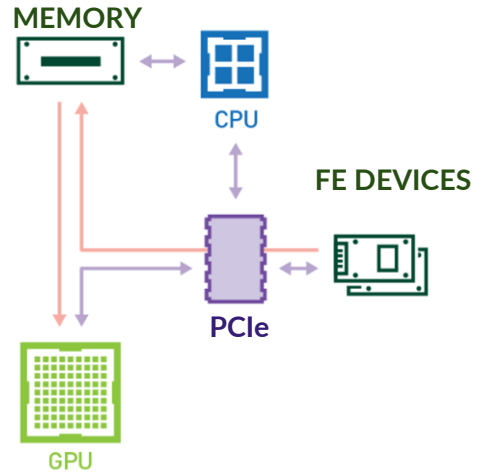
Injection of live traffic into digital simulator

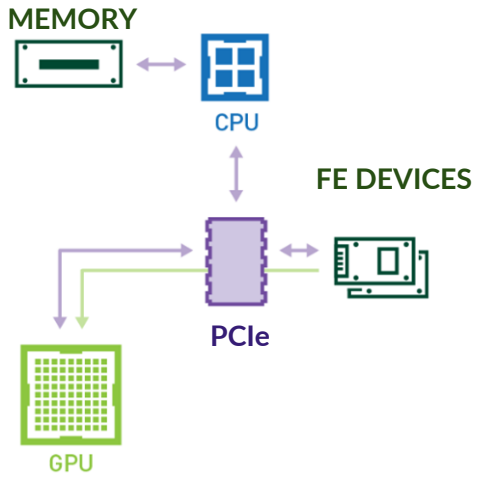
- Works only on Linux machines
  - Tun/Tap based
  - Tap is a raw ethernet interface into user space
- The method uses DPI-C interface of SystemVerilog
  - Imports C++ classed into SystemVerilog
  - Exports SystemVerilog tasks into C++ programs
- A C++ program opens Tap interface and passes raw ethernet input packet to SystemVerilog testbench
- SystemVerilog testbench passes raw ethernet output packets to C++ program
- A client program reads and writes to Tap interface

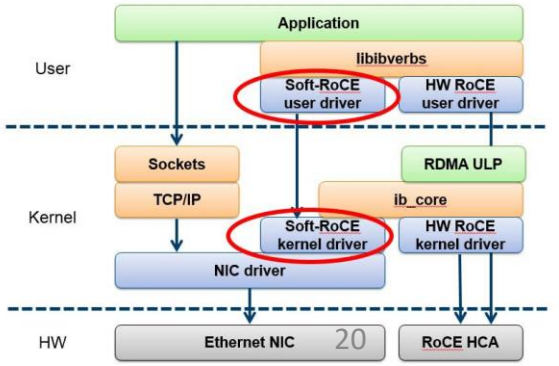


Networking activities

- Real-time processing of streaming data from FE devices with zero-copy access to memory
- Unburdening CPU by serving data directly to GPU memory
  - RDMA to GPU (GPUDirect)
  - Enabling CUDA-based applications on front-end data
- Expected activities
  - Emulating the RoCE transport in SW (via Soft-RoCE drivers) to enable testing of RDMA over Ethernet-enabled servers before the RoCE networking HW procurement
  - Deployment of networking hardware devices and qualification of the infrastructure in terms of throughput, latency and congestion avoidance
  - Test the commercial-available SW to move and control the data flow from RoCE to GPU







RoCEv2

G. Bortolato

Four different connection types

Type	ACK/NAK protocol	Private
RC	Yes	Yes
UC	No	Yes
RD	Yes	No
UD	No	No

Multiple message types

- RDMA WRITE
- SEND
- RDMA READ
- ATOMIC

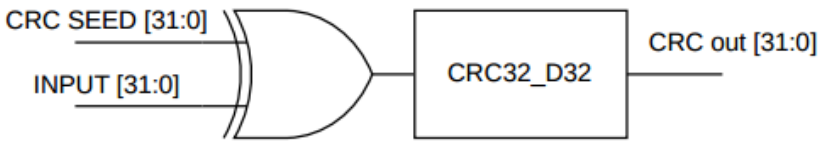
RDMA WRITE

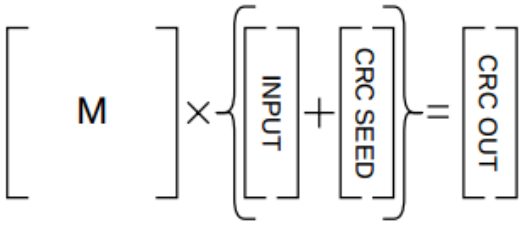
CRC32 operation

G. Bortolato

CRC32 computation can be seen as matrix multiplication:

$$CRC = M \times (I + S)$$





Where

- M is the matrix related to the CRC computation (generated form the polynomial)
- I is the 32-bit input
- S is the CRC seed or initial value (usually set to 0xFFFFFFFF)
- $\times$ is the AND operation
- $+$ is the XOR operation

CRC32 operation

G. Bortolato

$$\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{031} \\ a_{10} & a_{11} & \cdots & a_{131} \\ \vdots & \vdots & \ddots & \vdots \\ a_{310} & a_{311} & \cdots & a_{3131} \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{31} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{31} \end{bmatrix}$$

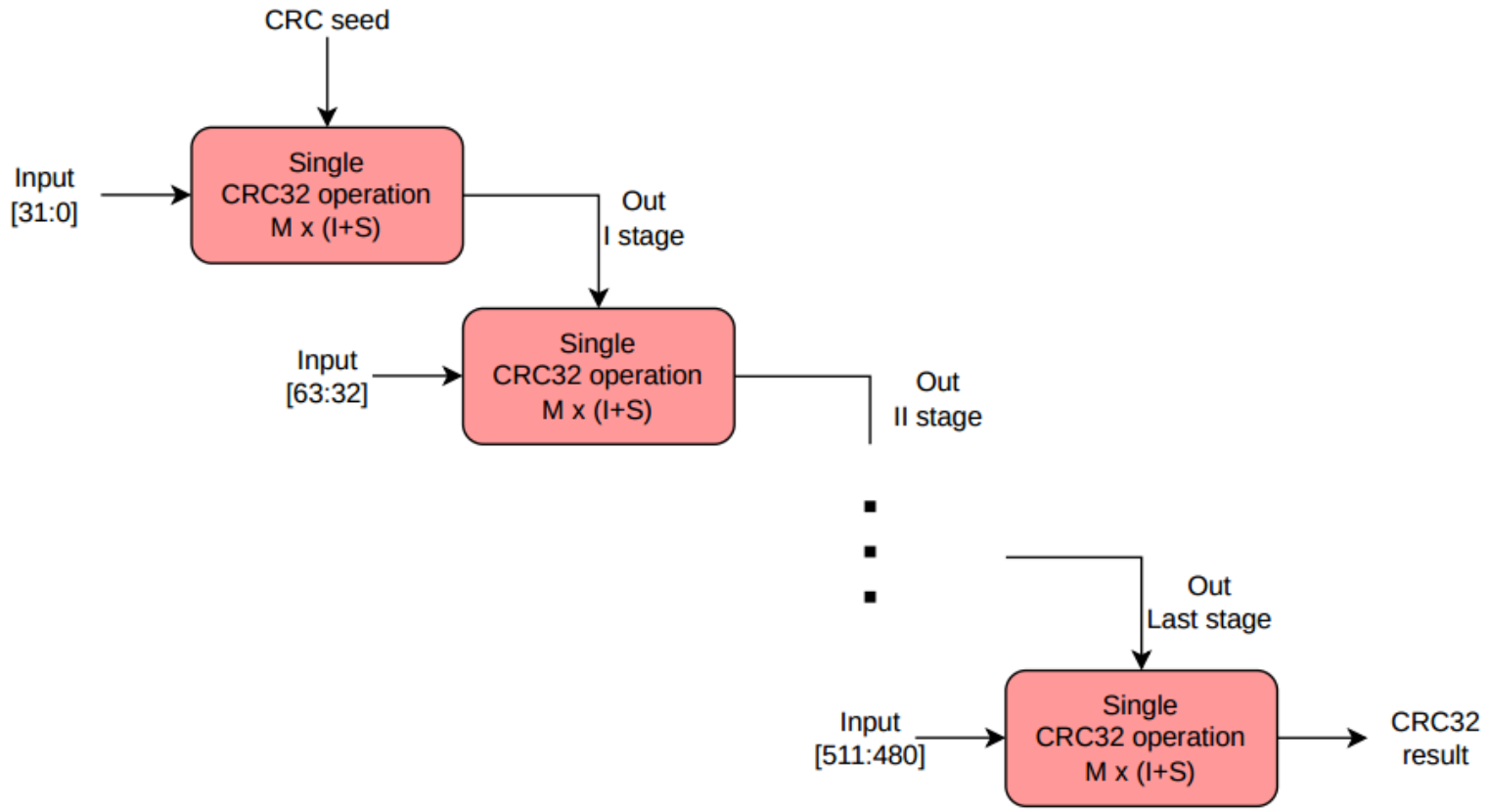
$$c_i = a_{i0} \wedge b_0 \oplus a_{i1} \wedge b_1 \oplus \cdots \oplus a_{in} \wedge b_n = \sum_{k=0}^{31} a_{ik} \wedge b_k$$

Where the matrix is generated starting from the polynomyal, endianess and shift direction

CRC32 operation

G. Bortolato

Now if we start applying the operation to subsequent 32-bit data we obtain the diagram below.



CRC32 operation

G. Bortolato

For example let's consider the computation for a 64-bit data:

$$CRC_{tot} = M \times (M \times (I_0 + S) + I_1)$$

Where

- M is the CRC32 matrix
- I_0 and I_1 are the 32-bit data slices
- S is the CRC seed, usually set to 0xFFFFFFFF

Expanding the product:

$$CRC_{tot} = M \times (M \times (I_0 + S) + I_1) = M^{(2)} \times I_0 + M^{(1)} \times I_1 + M^{(2)} \times S$$

Now for a 512-bit data vector

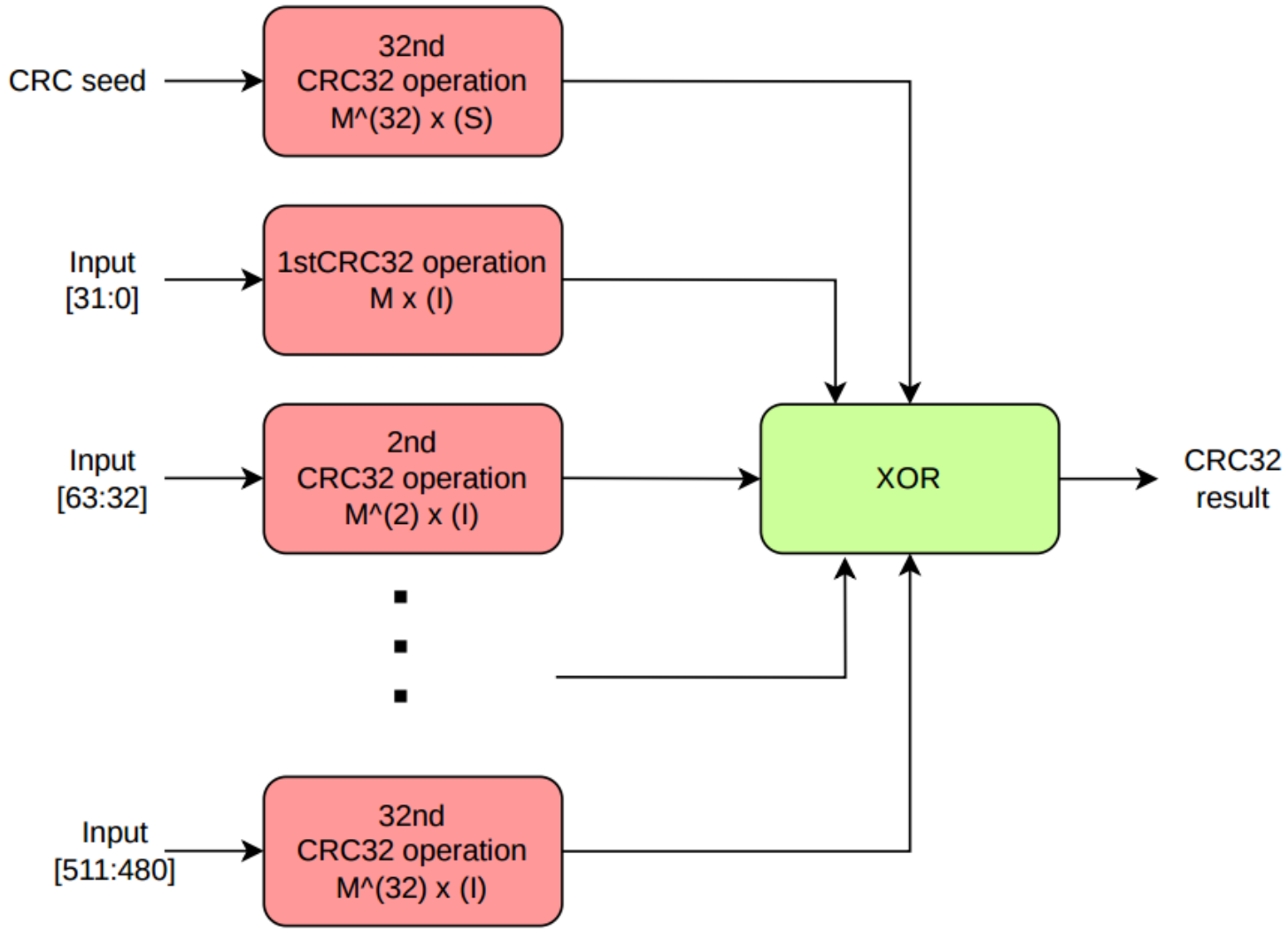
$$CRC_{tot} = M \times I_{15} + M^{(2)} \times I_{14} + M^{(3)} \times I_{13} + \cdots + M^{(16)} \times I_0 + M^{(16)} \times S$$

Or written in a shorter manner:

$$CRC_{tot} = \left(\sum_{(i=0)}^{15} M^{(i+1)} \times I_{15-i} \right) + M^{(16)} \times S$$

CRC32 operation parallel

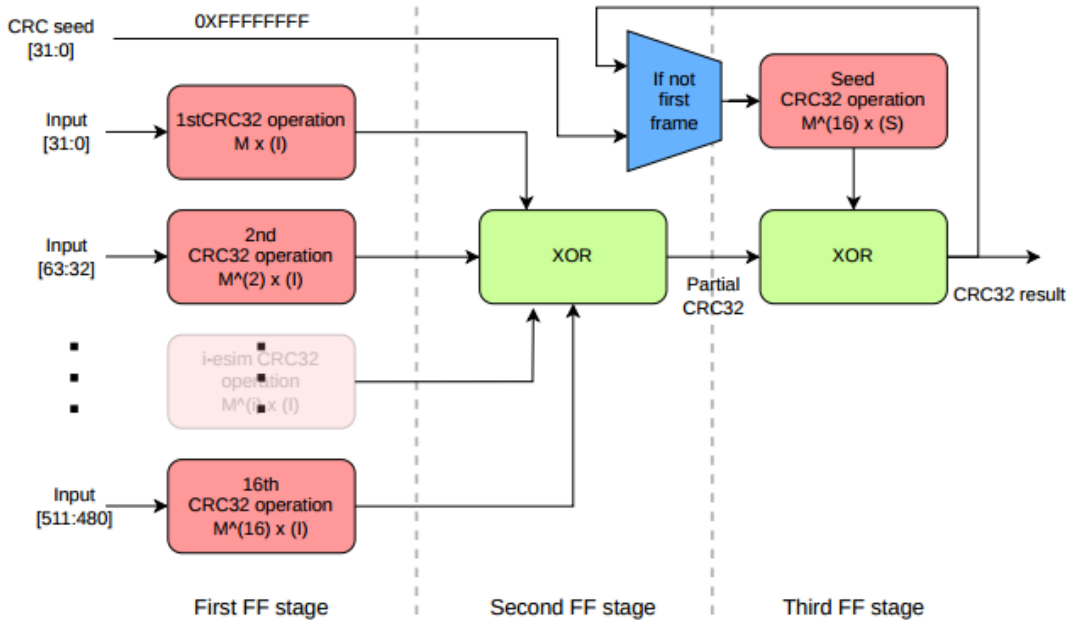
G. Bortolato



My CRC implementation

G. Bortolato

With the last equation we can pre-compute the various matrices and apply them tho the various slices. the result will be simply the XOR of the results.
Such computation can be pipelined to achieve the $\sim$ 322 MHz target frequency.
In my design 3 stages were used [4], possibility to stream with different keep values (still need to be multiple of 32).



[4] Matrix multiplications with data slices, XOR, XOR with CRC SEED result

Matrix-Matrix multiplication

G. Bortolato

The generating matrix to the n^{th} power is obtained with the matrix-matrix multiplication applied n times.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

$$c_{ij} = a_{i1} \wedge b_{1j} \oplus a_{i2} \wedge b_{2j} \oplus \cdots \oplus a_{in} \wedge b_{nj} = \sum_{k=1}^{n} a_{ik} \wedge b_{kj}$$

This computation is done in VHDL, where the matrices are computed at compile time.