Front-End Rdma Over Converged Ethernet, real-time firmware simulation

BELLATO Marco[1], BERGNOLI Antonio[1], BORTOLATO Damiano[2], BORTOLATO Gabriele[1,a], MENGONI Daniele[1,a], MIGLIORINI Matteo[1,a], MONTECASSIANO Fabio[1], PAZZINI Jacopo[1,a], TRIOSSI Andrea[1,a], VENTURA Sandro[1], ZANETTI Marco[1,a]

[1] INFN sezione Padova, [2] INFN LNL, [a] Università degli studi di Padva

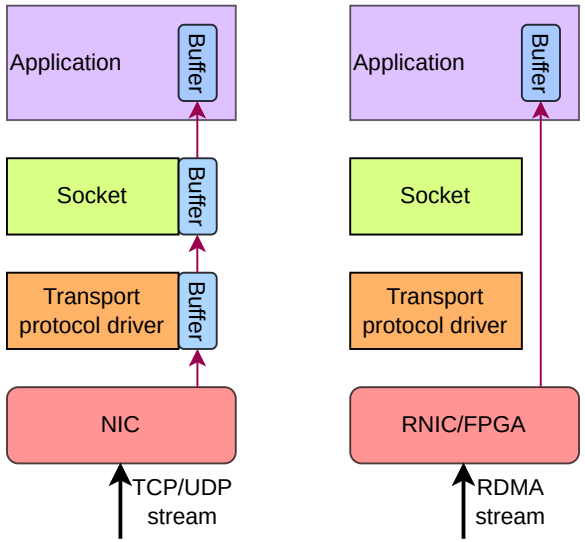
Finanziato dall'Unione europea NextGenerationEU

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Istituto Nazionale di Fisica Nucleare

Introduction on RDMA and RoCE

In a DAQ system a large fraction of CPU resources is engaged in networking rather than in data processing; common network stacks that take care of network traffic usually manipulate data through several copies.



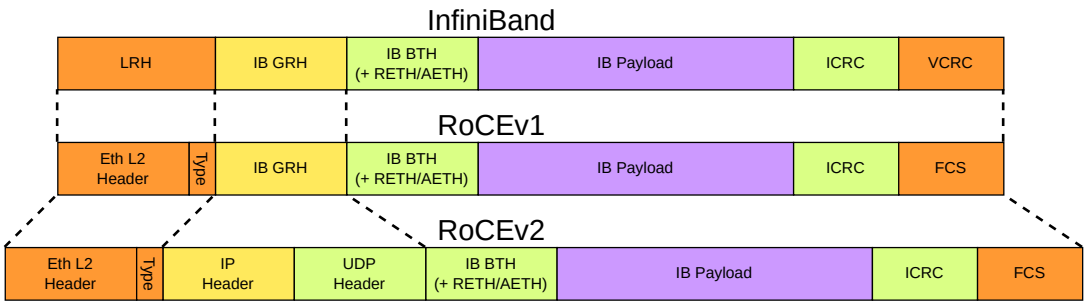
Remote Direct Memory Access (RDMA), as the name suggests, allows read and write operations directly in the target machine(s). This implies no OS involvement allowing high-throughput and low-latency applications.

This requires RDMA enabled NICs on both ends (RNIC) that performs the DMA, reducing the CPU load.

RDMA protocols

Many RDMA flavours are available:

- Infiniband
- RoCEv1, introduce the Ethernet framing
- RoCEv2, add the UDP/IP transport protocol
- iWARP, congestion-aware protocols



RoCEv2 is the only industry-standard Ethernet-based RDMA solution with a multi-vendor ecosystem. For this reason it is chosen as target protocol.

What is FERoCE?

Front-End RDMA over Converged Ethernet

Constant trend in producing larger and larger dataset in almost every experimental physics field, new requirements arise form that:

- High throughput, low latency
- Efficient data movement

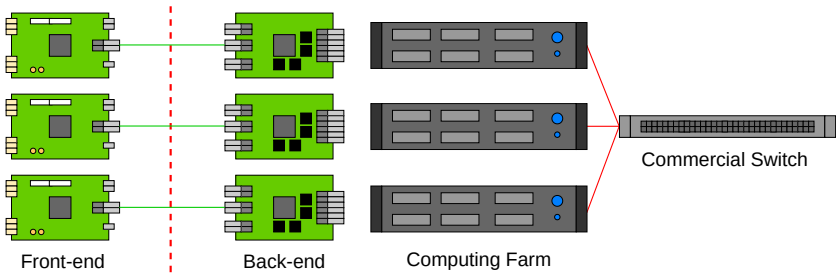
Such requirements lead to clever ideas:

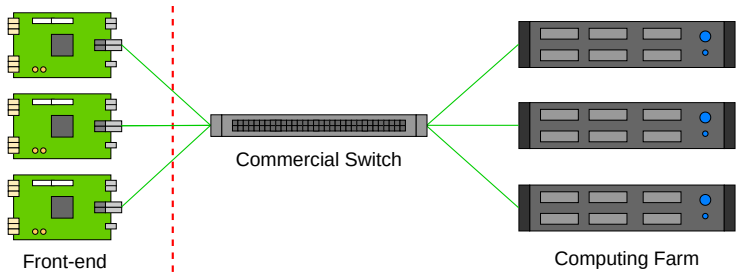
- Zero-copy protocols such as Ininiband or RoCE
- Move the network protocol directly in the front-end electronics (FPGA)
- Need to be scalable 1/10/100 Gbit/s
- Multi-vendor ecosystem Xilinx/Microchip/Altera

What can we achieve?

- Front-end initiates the RDMA transfer
- No point-to-point connection between front-end back-end
- Dynamical switching routing with COTS (lowering the costs and maintenance)

What is FERoCE?





Back-end boards required to get the data, and send it to the computing farms. This requires multiple custom cards and custom boards

Front-end boards send data already packaged within an ethernet frame allowing switching and routing. Choosing the right proper protocol allows the use of COTS switches

ETH RDMA network stack library has been chosen for the first prototype. Some of its characteristics:

- Entirely written in HLS (Vivado)
- It targets Xilinx FPGA with PCIe connection
- 10/100 GBit/s speeds

Real-time Firmware simulation

Why a dynamic firmware simulation is needed?

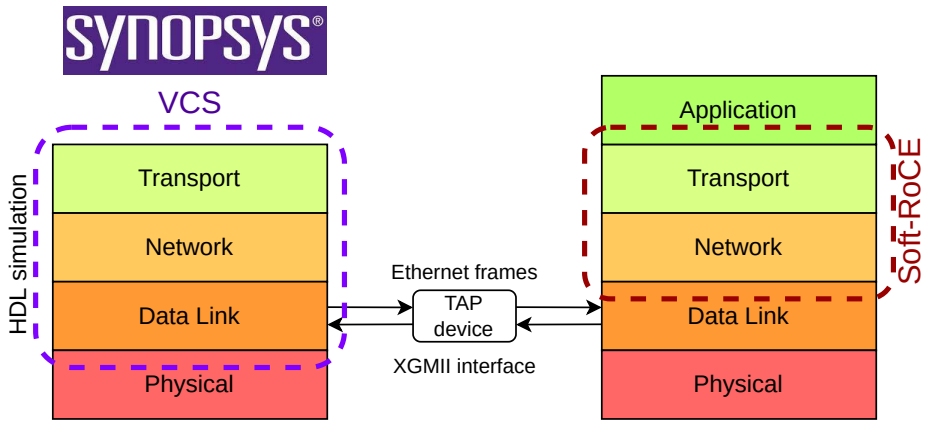
- Explore wider test-case phase space
- Feed raw ethernet frames directly to the code
- Simulate the HDL produced starting from the HLS code
- Capture frames with third party programs (e.g. Wireshark)
- Send frames to Soft-RoCE

Real-time Firmware simulation

Start form ETH network stack entirely developed in HLS. Functionalities and features must be understood: real-time firmware simulation with real network traffic.

- Works only on Linux machine: Tun/Tap devices
- It makes use of DPI-C interface of SystemVerilog: C++ code in our testbench!
- Tap device exchanges raw ethernet frames between simulation and Linux network stack
- We can capture such frames and study them

Simulation in Synopsys VCS.
XGMII interface directly form Xilinx MAC



Soft-RoCE used to capture and store in memory data sent.
Enable fast verification of the stack without going through sythesis/implementation every time.

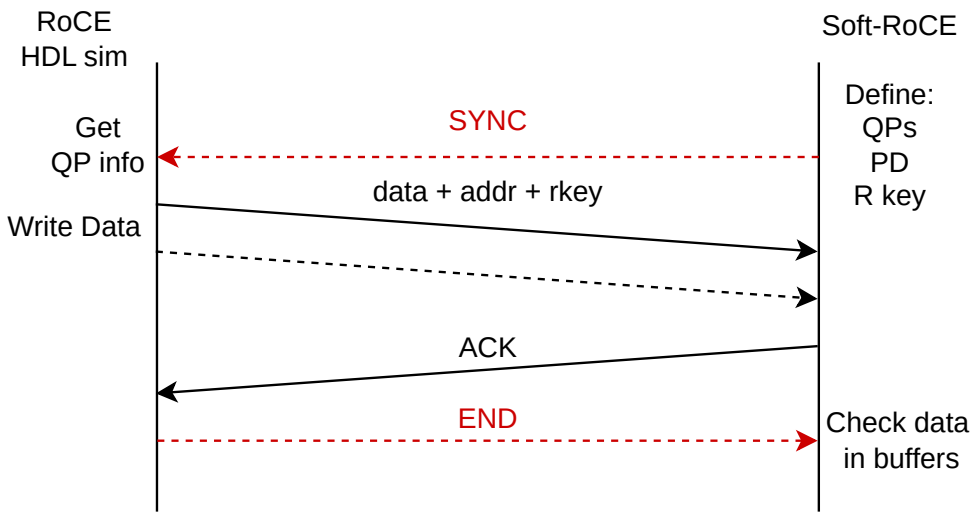
Once the stack has been verified the firmware can be built.

RoCE

RoCEv2 is a complex protocol, but not all its features are required for this project. RoCE supports many operations such as: RDMA SEND, RDMA WRITE, RDMA READ.

The goal is only to stream data and initiate the RDMA transfer, for this reason only RDMA WRITE is considered. Multiple steps need to take place in order to write data into the Soft-RoCE buffers:

- Create the connection: QP creation and memory allocation
- Set such variables in the RoCE stack
- Start writing
- End of operation



ETH TX engine details

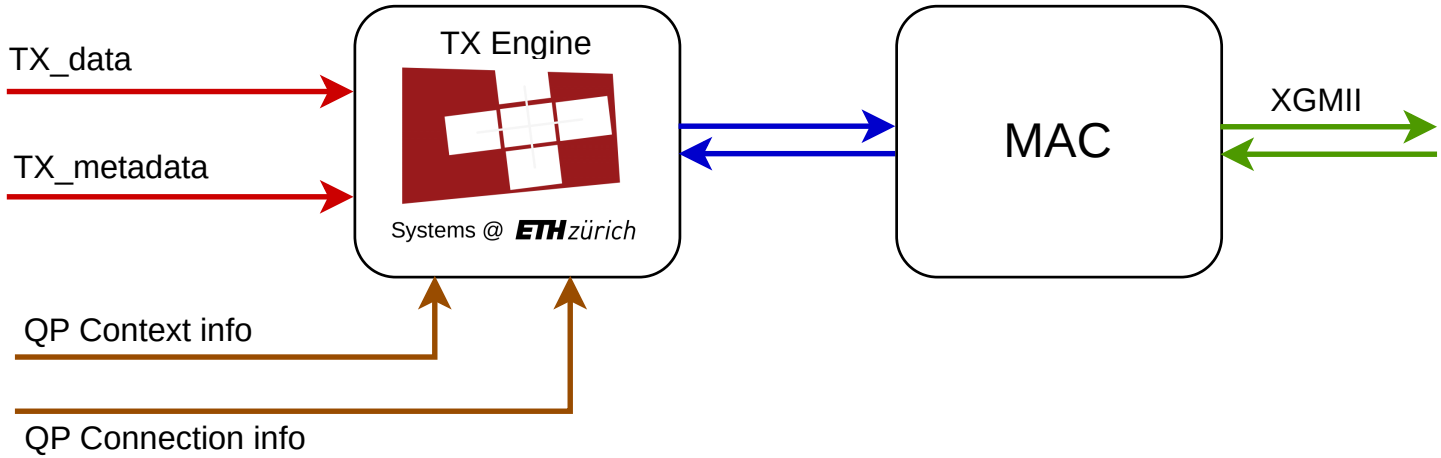
QP Context and connection info contains:

- QP numbers
- Remote and local PSNs
- Remote key
- Virtual address
- Remote IP address

TX metadata contains:

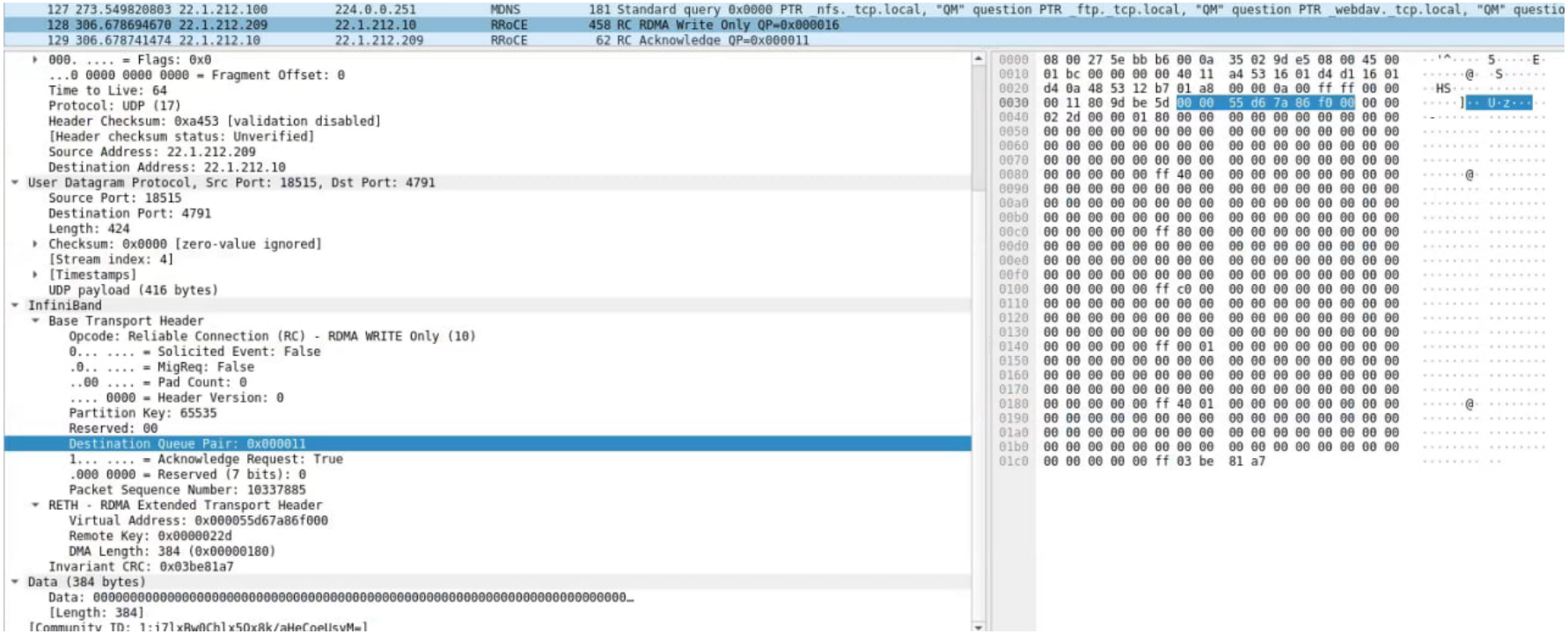
- operation type
- QP number
- Remote and local addresses
- DMA length

Once these three vectors are set a data transfer can be initiated.



Some results -Wireshark-

Used Wireshark to capture Ethernet frames coming out of the simulation.



In this frame we can check:

- Queue Pair number
- RDMA OP Code
- IP addresses
- Memory addresses

Summary and Outlook

Summry

- Developed a dynamic simulation
- Tested and verified ETH network stack
- Fed simulation RoCE data to Soft-RoCE end point

Outlook

- Cut ETH library to reduce the FPGA resource footprint
- Move from Xilinx HLS to a more agnostic HLS
- Delopy the light-RoCE in a Microchip FPGA