Integration of FPGA RDMA into the ATLAS Readout with FELIX in High Luminosity LHC

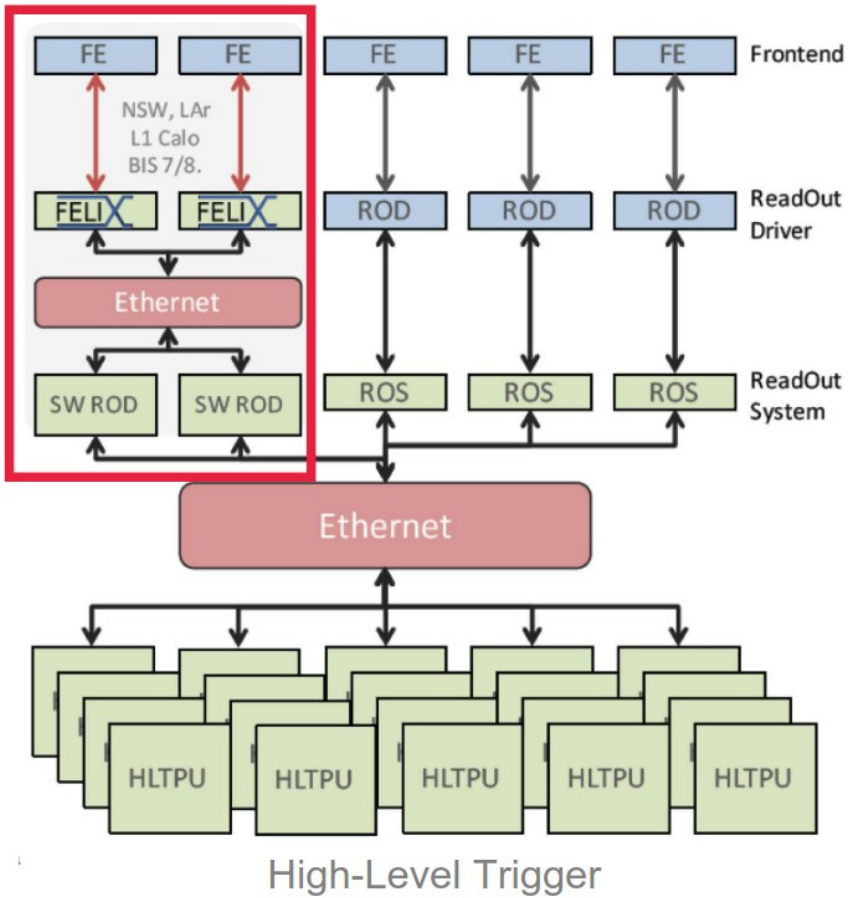
Matei-Eugen Vasile*, Sorin Martoiu, Gabriel Stoicea
Nayib Boukadida
Petru Micu, Alexandru Dumitru, Andrei-Alexandru Ulmamei,
Radu Hobincu, Cristina-Cerasela Iordache

TWEPP 2022 Topical Workshop on Electronics for Particle Physics

FELIX (Front-End LInk eXchange)

- The DAQ architecture until the end of **Run 2 (2015-2018)** was based on custom **Readout Drivers (ROD)** and **Readout Systems (ROS)** for each **ATLAS** subdetector
- The DAQ architecture in **Run 3** is introducing a new readout system based on **FELIX**[1] and **Software ROD (SW ROD)** for a number of subdetectors such as *NSW*, *LAr*, *L1 Calo* and *BIS 7/8*
- The purpose of **FELIX** + **SW ROD** is:
  - replace custom hardware and links
  - use standardized links
  - use commodity computer hardware
  - use commodity network infrastructure

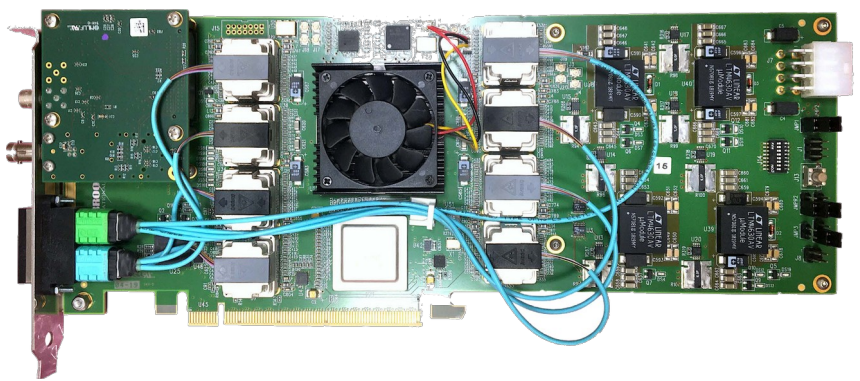
  in order to:
  - reduce hardware/firmware development effort
  - decouple data transport from data processing
- For **Run 4**, **FELIX** and a **successor to SW ROD** are planned to be used for all subdetectors



[1] *FELIX: the new ATLAS readout system from Run 3 to High Luminosity LHC,* Frans Schreuder (Nikhef), TWEPP 2021, https://indico.cern.ch/event/1019078/contributions/4494722/

FELIX boards

- FELIX Phase 1 / Run 3 hardware:
  - The FLX-712 card:
    - FPGA: Xilinx Kintex UltraScale KU115
    - 16-lane PCIe Gen3 for a bandwidth of up to 128 Gb/s
    - 4 or 8 MiniPODs to support 24 or 48 bidirectional optical links

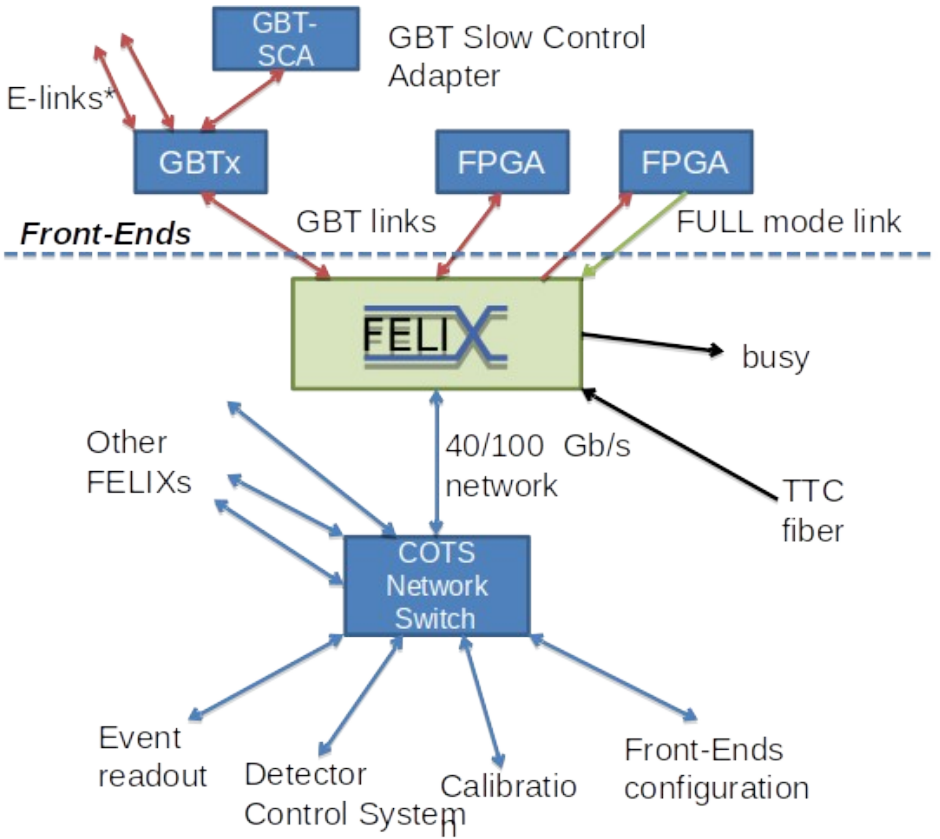


- FELIX Phase 2 / Run 4 hardware:
  - The FLX-181/182 card:
    - FPGA: Xilinx Versal Prime VP1802
    - 16-lane PCIe Gen4 for a bandwidth of up to 256 Gb/s
    - 24 Firefly optical links with up to 25 Gb/s
    - 4 FireFly optical links which can be used for a 100GbE port

FELIX & netio-next

- **FELIX** routes event, detector control, configuration, calibration and monitoring data
- Aggregates e-links
- Connects bidirectionally ATLAS detector Front-Ends and the DAQ system
- Detector independent
- It is built using:

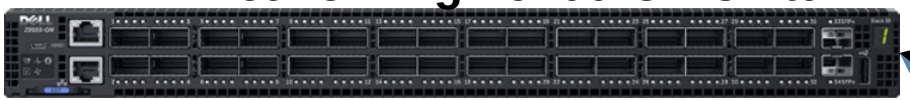
- **netio-next**[1] is a CERN-developed communication library based on libfabric
- Supports various RDMA implementations such as *Infiniband*/*RoCE*
- Implements a publisher/subscriber paradigm
- Uses *RDMA Send* messages for all its communication
- It is the network interface layer of FELIX ATLAS data acquisition in LHC Run 3 (2022)



[1] *Event-driven RDMA network communication in the ATLAS DAQ system with NetIO*, Jorn Schumacher (CERN), CHEP 2019, https://indico.cern.ch/event/773049/contributions/3473244/

Development setup

DELL Networking Z9100-ON switch



switch:
OS: Cumulus Linux 3.7.15
supports **RoCEv2**
Ports: 32x 100GbE QSFP28 and 2x 10 GbE SFP+

DELL PowerEdge T640

publisher & provider:

OS: CERN CentOS Linux release 7.9.2009 (Core)
RDMA: NVIDIA Mellanox ConnectX-5 EX Dual Port 40/100GbE QSFP28
FPGA board: XILINX Alveo U250

100 Gbps

100 Gbps

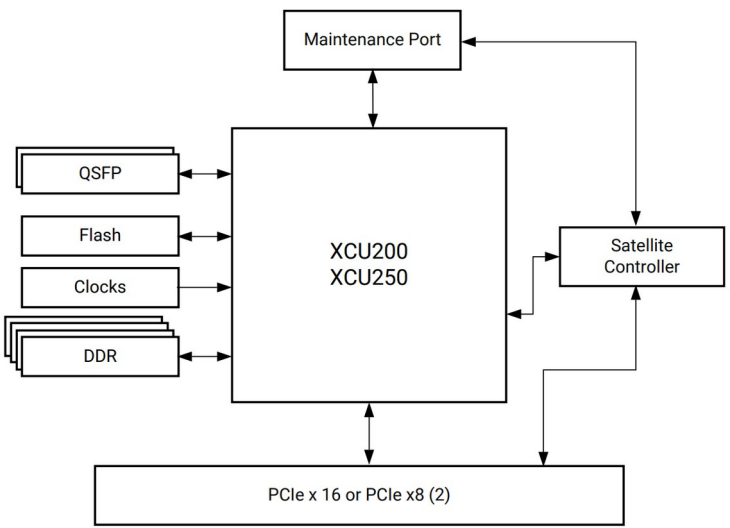
DELL PowerEdge R740XD

subscriber:

OS: CERN CentOS Linux release 7.9.2009 (Core)
RDMA: NVIDIA Mellanox ConnectX-5 EX Dual Port 40/100GbE QSFP28

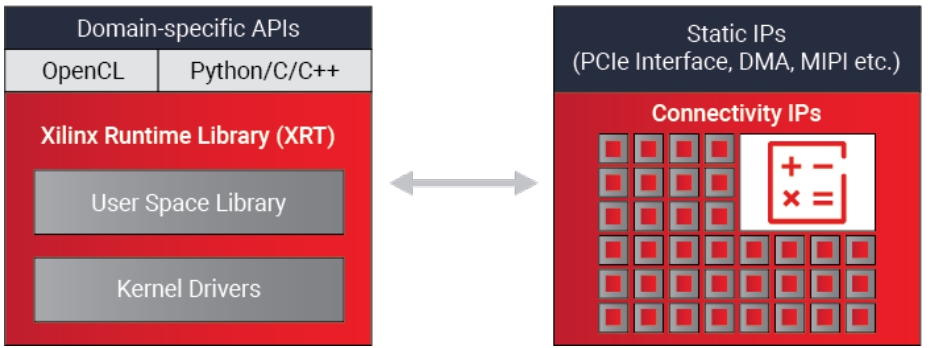
Hardware: Xilinx Alveo

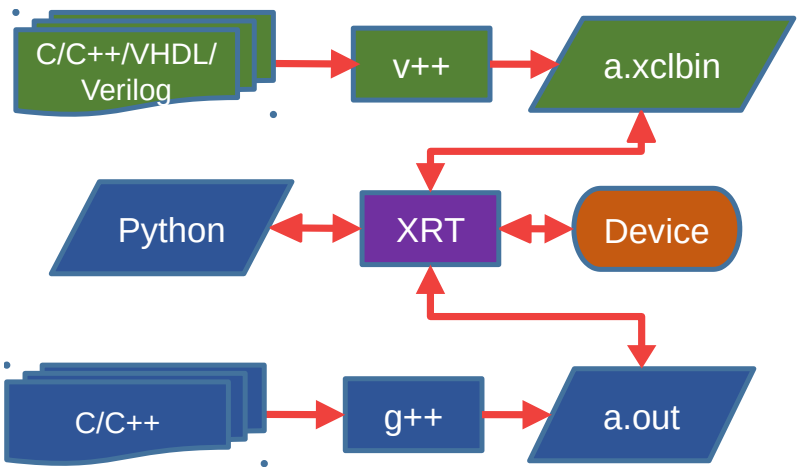
- **Xilinx Alveo U250** Data Center Accelerator Card
  - custom-built **UltraScale+** FPGA
  - exclusive to Alveo boards
- **2 x QSFP28** capable of **100GbE**
- **64 GB DRAM** with a bandwidth of 77 GB/s
- **PCI Express Gen3 x16**
- https://www.xilinx.com/products/boards-and-kits/alveo/u250.html



Hardware: XRT (Xilinx Runtime library)

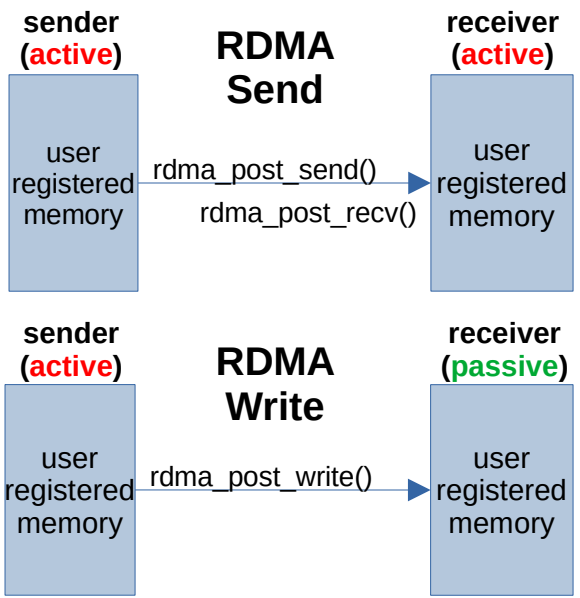
- **Xilinx Runtime library (XRT)** is a component of the *Vitis Unified Software Platform*
- Open Source:
  - Available on *GitHub* at https://github.com/Xilinx/XRT
- facilitates communication between
  - application code
  - accelerated-kernels deployed on the reconfigurable portion of **Alveo** accelerator cards, **Versal ACAP**s etc.
- https://www.xilinx.com/products/design-tools/vitis/xrt.html





Existing RDMA implementation (software)

- **RDMA** (Remote Direct Memory Access)
  - transfer data with little CPU involvement
- **Transport Services**:
  - *Reliable Connection (RC)*
    - Similar to *TCP/IP*
- **Transport Functions:**
  - *Send* – both ends do work
  - *Write* – only sender does work



- [Initial idea] At high loads:
  - PCIe bus could become a bottleneck
  - CPU could become heavily loaded
  - RDMA data transfer is implemented:
    - In software (using RDMA-capable NICs)
    - Using RDMA Send messages:
      - both endpoints need to be involved
- Potential solution (implemented feasibility study[1]):
  - Implemented RDMA in the FPGA
    - Transfer detector data directly to client from the FPGA board
    - Control plane unchanged: the NetIO publisher continues managing the subscriptions as before
    - Data plane changed: RDMA FPGA implementation
    - NetIO publisher forwards the connection parameters from the subscriber to the FPGA data provider

Existing RDMA implementation (hardware)

- Starting point:
  - **FPGA Network Stack**
  - Developed by **ETH Zurich**
  - Fully written in **HLS** (High-Level Synthesis)
  - Contains modules required for setting up network communication:
    - ARP
    - ICMP
    - UDP/IP
    - TCP/IP
    - ROCEv2 (RDMA over Converged Ethernet)
  - Available as **Open Source** on GitHub
    - https://github.com/fpgasystems/fpga-network-stack
    - https://github.com/fpgasystems/davos
- RDMA implementation uses is **RoCEv2 (RDMA over Converged Ethernet, version 2)**
  - allows the Infiniband-based RDMA to work over any routed IP-based network
- RoCEv2 core contains support for:
  - RDMA Write
  - RDMA Read
  - Retransmission
- Functionality added:
  - RDMA Send
  - Updated retransmission
  - Invariant CRC calculation (work in progress)
  - Connection management (work in progress)

Developed items

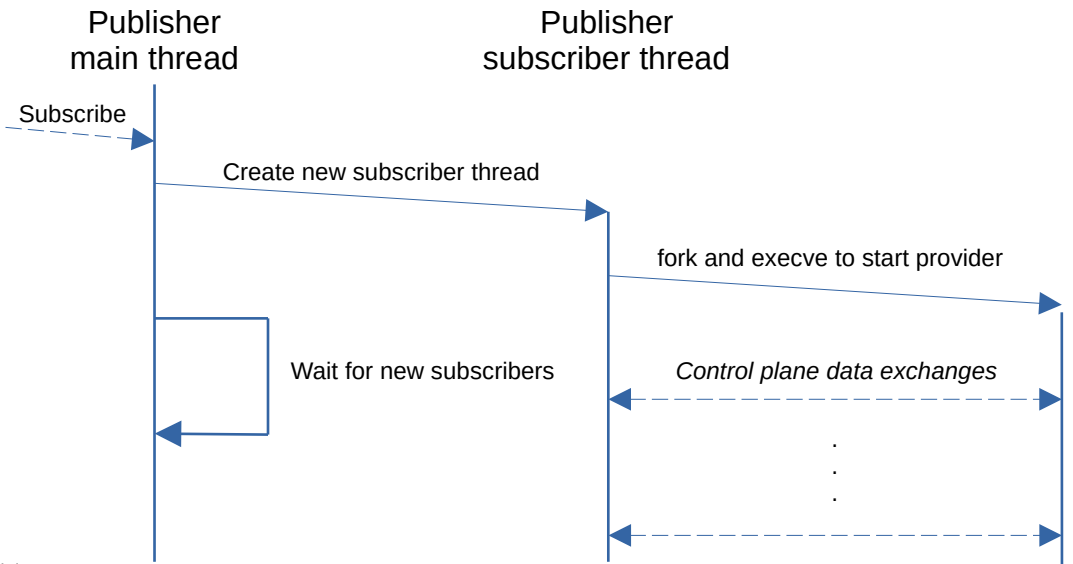
- Hardware design:
  - Ported the FPGA implementation to run on Xilinx Alveo boards
  - Implemented software client using XRT (Xilinx Runtime library)
  - Improved FPGA implementation of RDMA
- Software:
  - Migrated from using *netio-next* as a basis of the implementation to the FELIX distribution
  - Implemented a **full 3-way handshake** between the **subscriber** and the **provider** between which the RDMA connection is established
  - Implemented a modular solution for handling subscribers and providers on the publisher side:
    - Each subscriber is running on its own thread
    - Each provider is running in its own process (using *fork* and *execve*)
- Testing
  - Functional
  - Performance

New software features

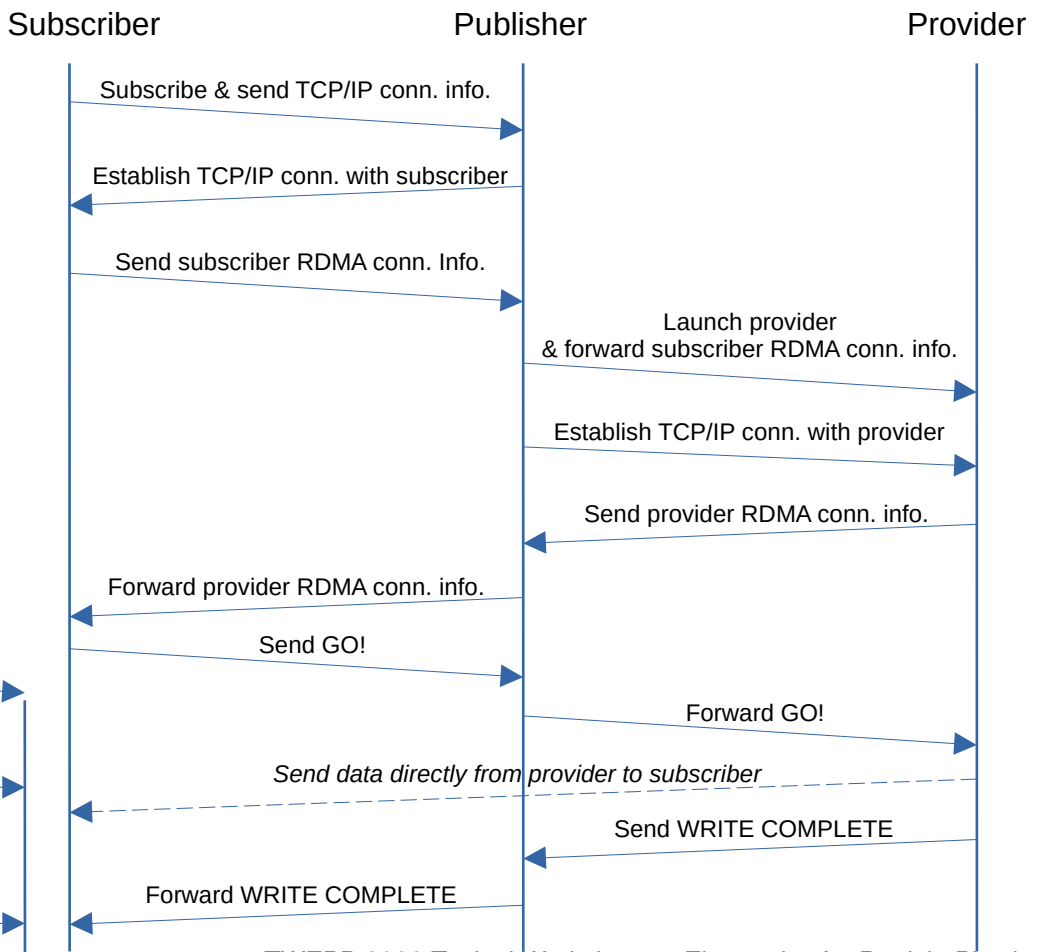
- Full 3-way handshake needed before any data can be moved between any 2 given endpoints
- TCP/IP connections between:
  - **subscriber & publisher**
  - **provider & publisher**

  in order to provide a way for the **subscriber & provider** to perform the 3-way handshake
- multi-threaded solution: a **thread** for each **subscriber**

subscriber & provider management:

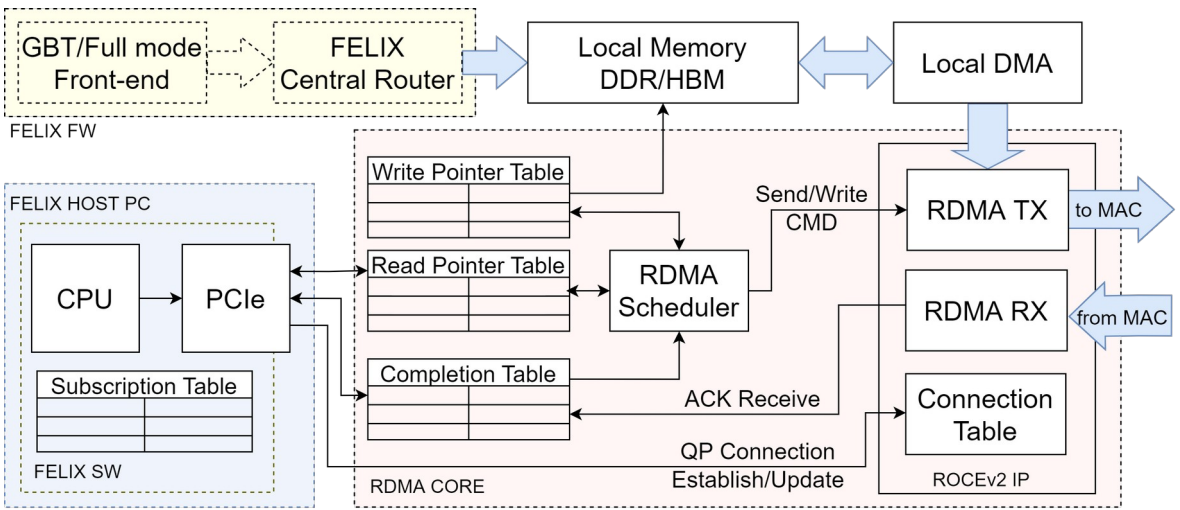


connection establishment with full handshake:



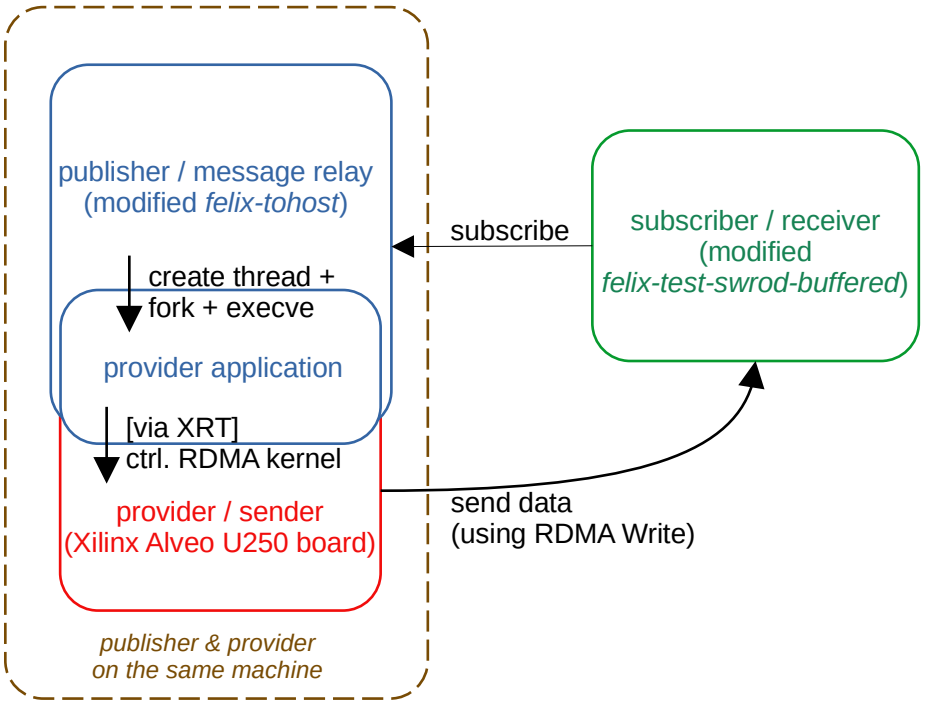
New hardware features: FPGA RDMA on Alveo & XRT

- Starting point:
  - The **Xilinx XUP Vitis Network Example**
    - Available on GitHub at https://github.com/Xilinx/xup_vitis_network_example
  - The existing FPGA RDMA implementation
- Refactoring of the FPGA RDMA implementation to work on the **Alveo** platform
  - Using free-running kernels
- Development of the software interacting with the **Alveo** kernels
  - Using the C++ XRT Native API
  - Basis of the FPGA RDMA **provider** used by the software framework

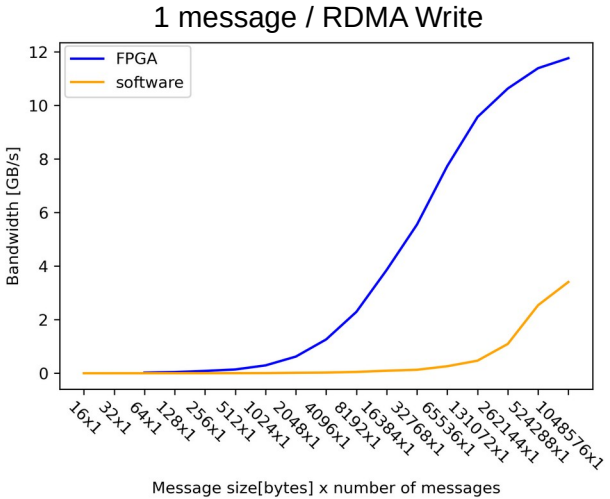


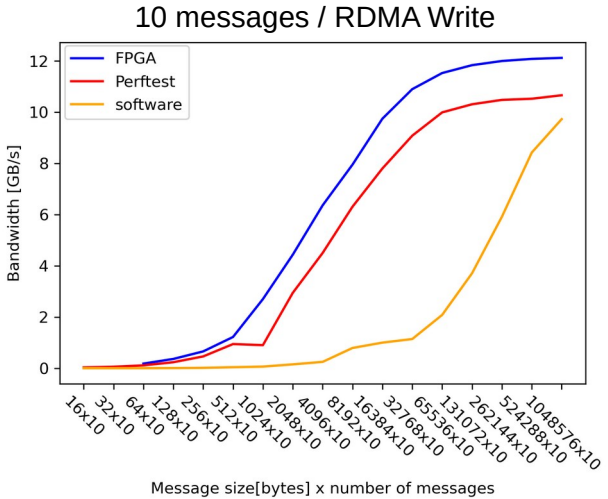
Test setup

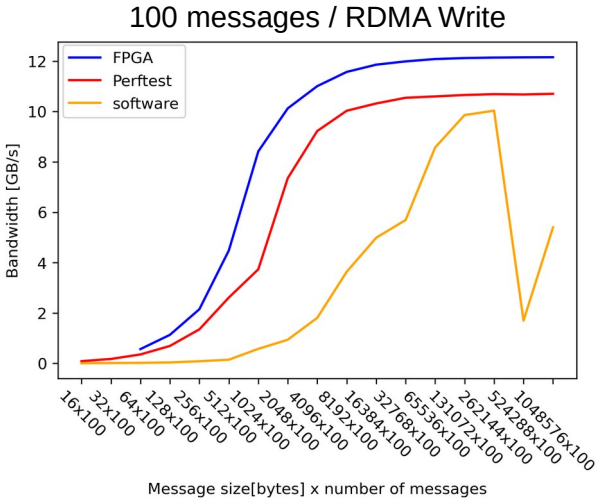
- The **receiver** is a **modified felix-test-swrod-buffered** from the FELIX Distribution
- The **sender** is made out of two parts:
  - The firmware running in the **FPGA**, on a **Xilinx Alveo U250** board
  - The software running on the PC hosting the FPGA, based on the **XRT** native interface
- A modified **felix-tohost**, also from the FELIX Distribution, is the **publisher,** acting as **relay** between the **sender** and the **receiver**
  - Receives subscription requests from the **sender** (the *subscriber*)
  - Starts the receiver **process** (the *provider*)
  - Establishes out-of-band *TCP/IP connections* with both **sender** and **receiver,** and **relays** control messages between them in both directions
    - The data is sent directly from the **sender** to the **receiver** over the network
- Transmission time is measured between the times when:
  - Transmission starts
  - Acknowledgement for the last message is received
- For now, only a 1 subscriber and 1 provider setup has been tested
- Perftest *ib_write_bw* used as reference

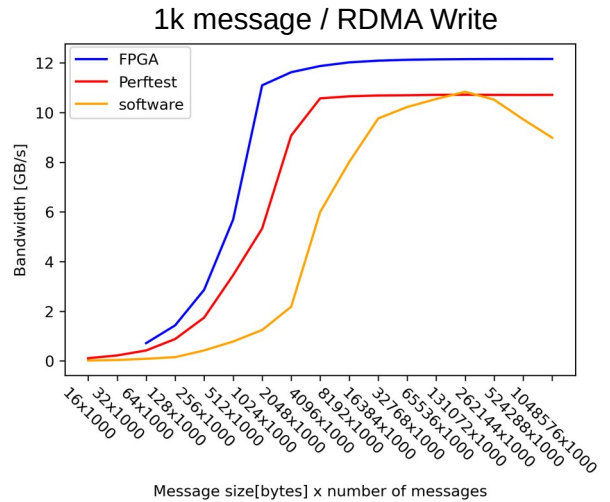


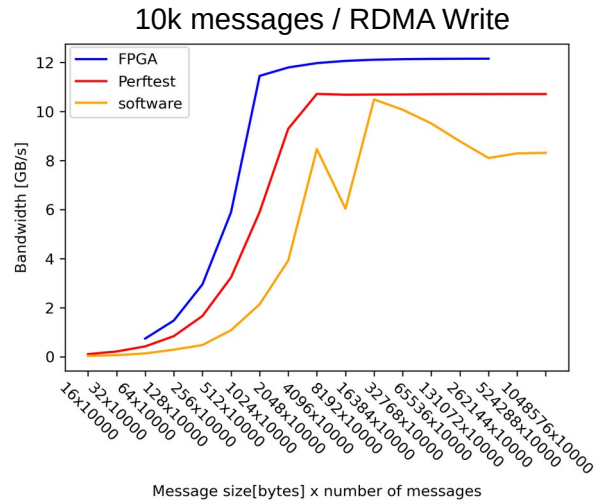
RDMA implementation performance (1)

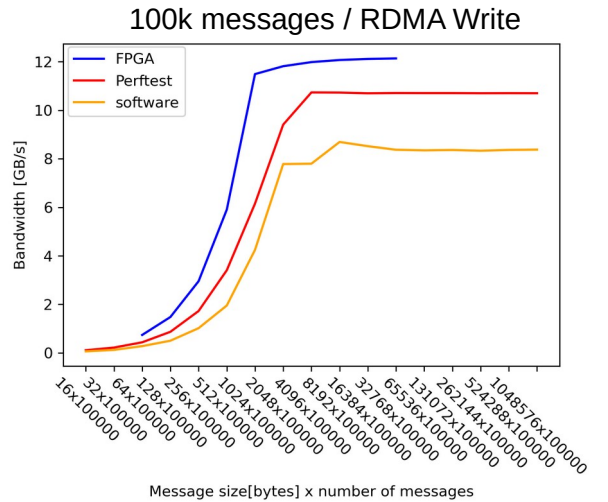




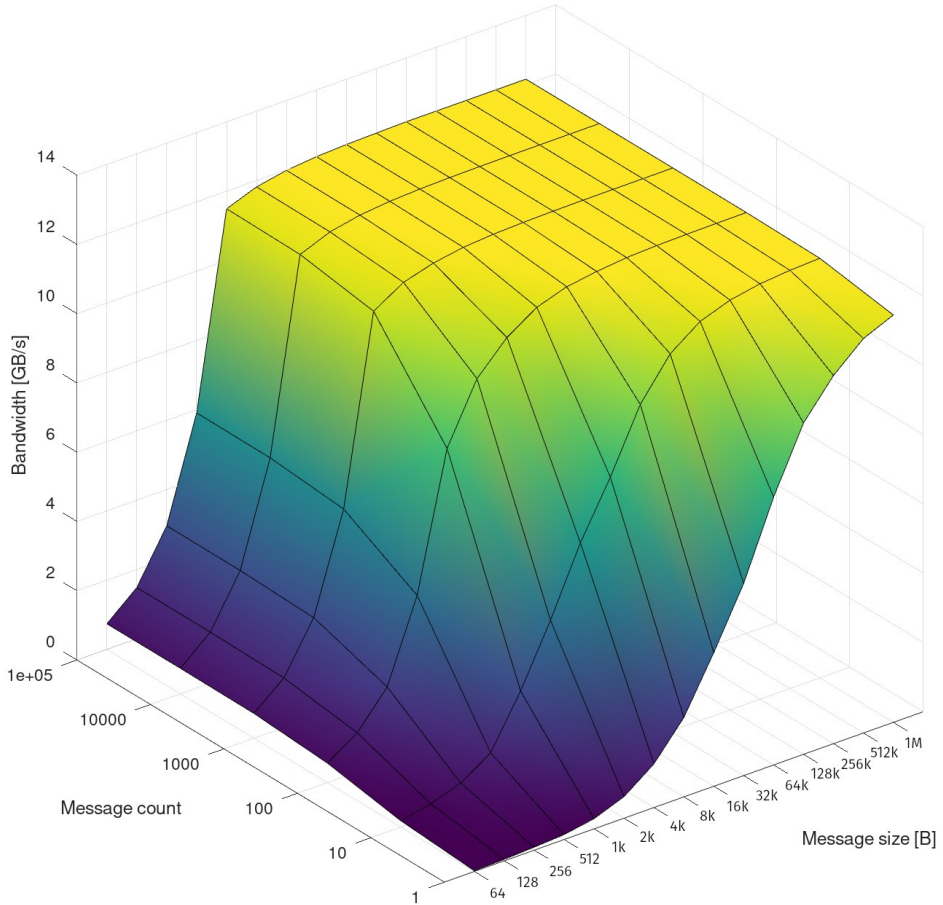


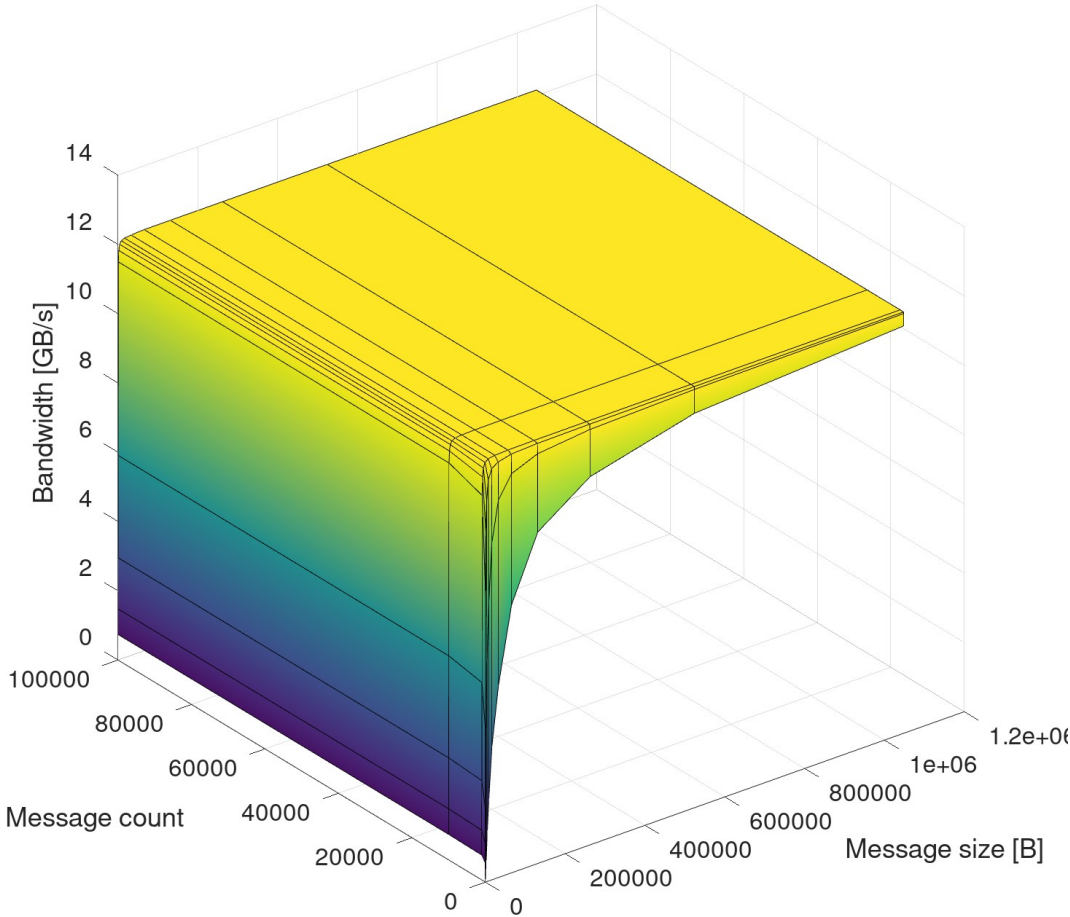






RDMA implementation performance (2)





Conclusions & Future Work

- New implemented functionality:
  - Full 3-way handshake RDMA connection establishment between the subscriber and the provider on top of *felix-star*
  - Modularized architecture that disconnects the provider implementation from the publisher implementation
  - Sending data using RDMA Write from the local memory of the FPGA
    - Using an implementation running on a Xilinx Alveo board
    - Using XRT-based application to interact with the Alveo board
- Tested the functionality and the performance of the implementation:
  - Performance of the FPGA implementation surpasses those of the previous implementation version, getting close to the theoretical bandwidth limit of the communication link
- Future work:
  - Complete the implementation of support for multiple subscribers and multiple providers
    - Test the functionality and performance of the implementation
  - Perform an analysis on what could be the best choice for message count and message size so that this information could be used in further developments
  - Given that RDMA Write simply writes in a remote memory region, memory management strategies for handling the proper use of that memory will be needed
  - Last, but not least, with the new FELIX FLX-182 prototype being planned to have a 100GbE port onboard, we are looking forward to getting this FPGA RDMA implementation running on a FELIX board

Backup Slides

Development setup

DELL Networking Z9100-ON switch

MM AOC – Multimode Active Optical Cable
DAC – Direct Attached Cable (Copper)

switch:
OS: Cumulus Linux 3.7.15
CPU: Intel Atom C2538 2.4 GHz
RAM: 8GB DDR3
ASIC: Broadcom Tomahawk BCM56960 – supports **RoCEv2**
Ports: 32x 100GbE QSFP28 and 2x 10 GbE SFP+
ECN (Explicit Congestion Notification) enabled on 100 GbE ports for lossless network operation

DELL PowerEdge T640

100 Gbps MM AOC

100 Gbps DAC

DELL PowerEdge R740XD

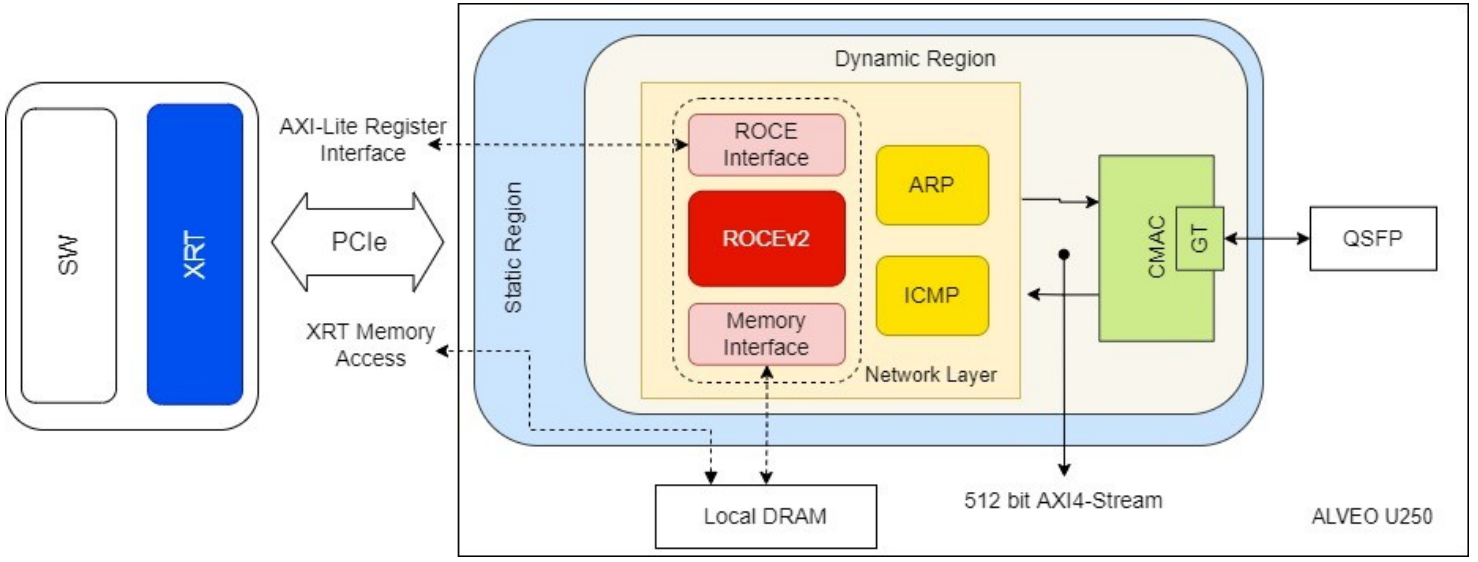
publisher & provider:

OS: CERN CentOS Linux release 7.9.2009 (Core)
CPUs: 2x Intel Xeon Silver 4215R 3.2G, 8C/16T, 11M Cache
RAM: 128GB DDR4, 3200MT/s
RDMA: NVIDIA Mellanox ConnectX-5 EX Dual Port 40/100GbE QSFP28
Firmware: 16.32.2004;
Driver: MLNX-OFED-LINUX 5.6-2.0.9.0
FPGA board: XILINX Alveo U250

subscriber:

OS: CERN CentOS Linux release 7.9.2009 (Core)
CPUs: 2x Intel Xeon Gold 6234 3.3G, 8C/16T, 24.75M Cache
RAM: 320GB DDR4, 3200MT/s
RDMA: NVIDIA Mellanox ConnectX-5 EX Dual Port 40/100GbE QSFP28
Firmware: 16.32.2004
Driver: MLNX-OFED-LINUX 5.6-1.0.3.1

Alveo Test System: Resource Usage



	CLB LUTs		CLB Registers		Block RAM Tile		URAM	
ROCEv2 IP	37224	2.16%	78435	2.27%	45	1.67%	0	0.00%
ROCE Infrastructure	22836	1.32%	17593	0.51%	16	0.60%	18	1.41%
Network Layer (ARP, ICMP,...)	15762	0.91%	31705	0.92%	3.5	0.13%	0	0.00%
CMAC	13876	0.80%	41244	1.19%	22	0.82%	0	0.00%
TOTAL	89698	5.20%	168977	4.89%	86.5	3.22%	18	1.41%
Alveo U250 Total	1726216		3456000		2688		1280	