SPADI Alliance

Signal processing and data acquisition infrastructure alliance

Recent developments of data link from FEE

Yuto Ichinohe (RIKEN Nishina Center)

Streaming Readout Workshop SRO-XIV — FRIB, MSU
2026.9.29

Outline

1. UDP based data-streamer and slow-control protocol (ANEMOI)

2. 100 Gbps RDMA module based on FEROCE

Background

SPADI alliance

- Signal Processing and Data Acquisition Infrastructure (SPADI) alliance working toward the development and standardization of streaming readout DAQ.
- In the SPADI alliance, the **Ethernet-based data communication** has been adopted for the data link of front-end electronics (FEE) to utilize Commercial Off-The-Shelf (COTS) components.

SPADI Alliance

Signal processing and data acquisition infrastructure alliance

SiTCP (silicon TCP)

A proprietary network processor for FPGAs widely used in particle and nuclear physics experiments in Japan. It provides Ethernet-based data transfer over GbE/10GbE using TCP.



Concept: UDP-based DAQ

TCP-based data transfer is reliable, but…

- Significant CPU resources required for data reception
- TX buffer for retransmission is required in FEE
- Backpressure from receiver

SiTCP specific shortcomings

- Data streaming to / from only a single host
- No explicit mechanism for simultaneous addressed-register control requests from multiple hosts
- Limited flexibility in MTU configuration
- No DHCP-based dynamic IP configuration

UDP

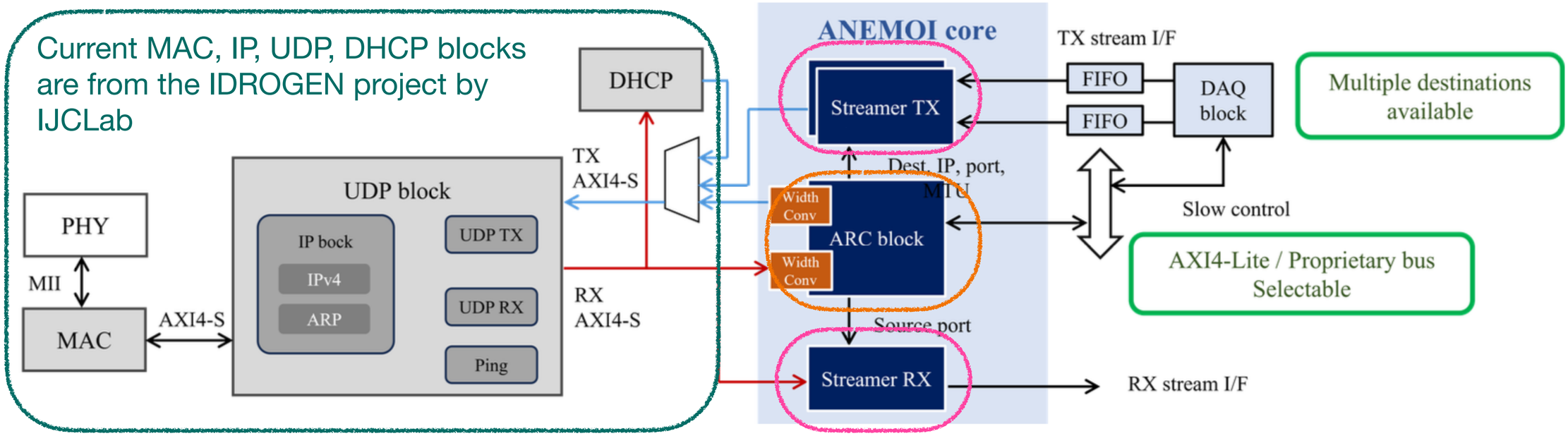
- Lightweight / high throughput
- No need for a TX buffer for the data retransmission
- Detaches FEEs from the backpressure from PCs
- Recent studies have explored reliable, lossless communication up to 100 Gbps

→ New UDP-based DAQ/slow control protocol

ANEMOI

ANEMOI (Adaptive NEtwork Messaging for Open Instruments)

ANEMOI moves the high-throughput data path to UDP and adds a more flexible streaming/control architecture.

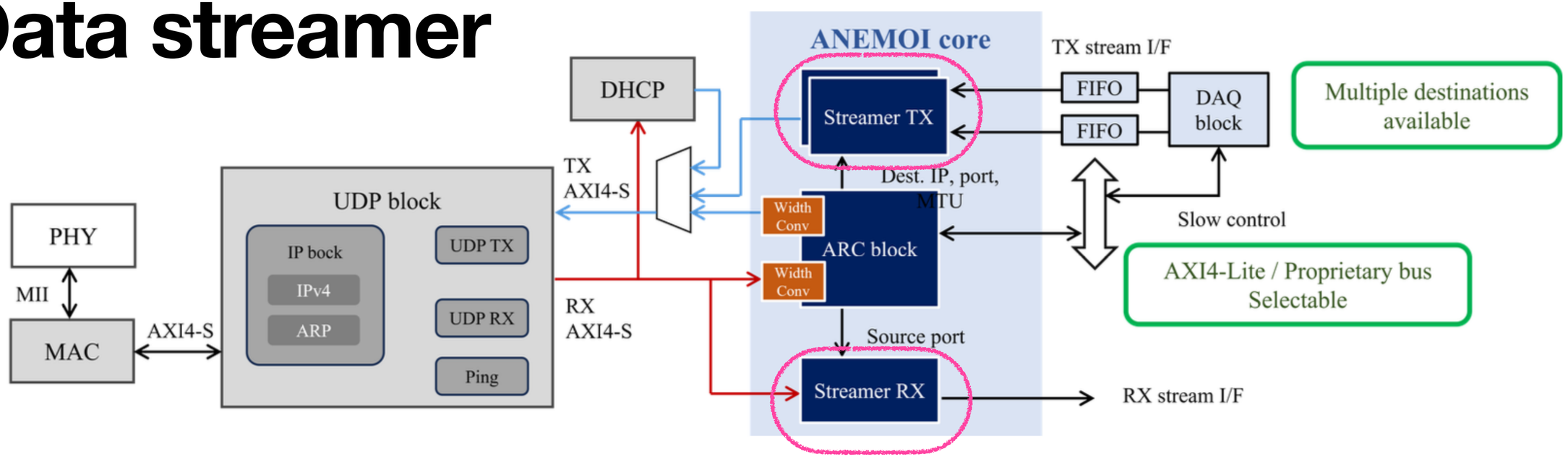


Developed in ANEMOI

- Data Streamer + Addressed Register Control (both based on UDP)
- The ANEMOI core can be adapted to any UDP block with the AXI4-Stream interface.

Low-level functionalities (MAC, IP, UDP, DHCP) are reused from IDROGEN project by IJCLab

Data streamer

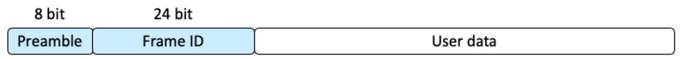


Streamer TX

- Stores data from user circuit into the internal FIFO and starts transmission when
  - Data amount stored in FIFO exceeds the preset packet length
  - A certain amount of time has passed after the first data is stored (time out)

→ The user circuit can stream data without considering the Ethernet framing

- Parameters (IP, port, MTU) are changeable through the ARC block
- Multiple destinations available by instantiating multiple streamer modules



Packet structure is still under discussion

Streamer RX

It contains an internal FIFO and buffers RX data destined for specific port numbers.

- The output interface is AXI4-Stream

Slow control: ANEMOI ARC (Addressed-Register Control)

ARC:
Receives a UDP request → Performs the requested bus access → Waits for busAck → Generates and sends the response packet.

- 32-bit data, 32-bit address
- Possible to control from multiple hosts

Broadcast responder
Responds to the limited broadcast and returns MAC address + IP address + arbitrary data (e.g., firmware ID)

Supports sequential / list mode bus access

Header (64-bit)
Address
Data 0
Data 1
Data 2

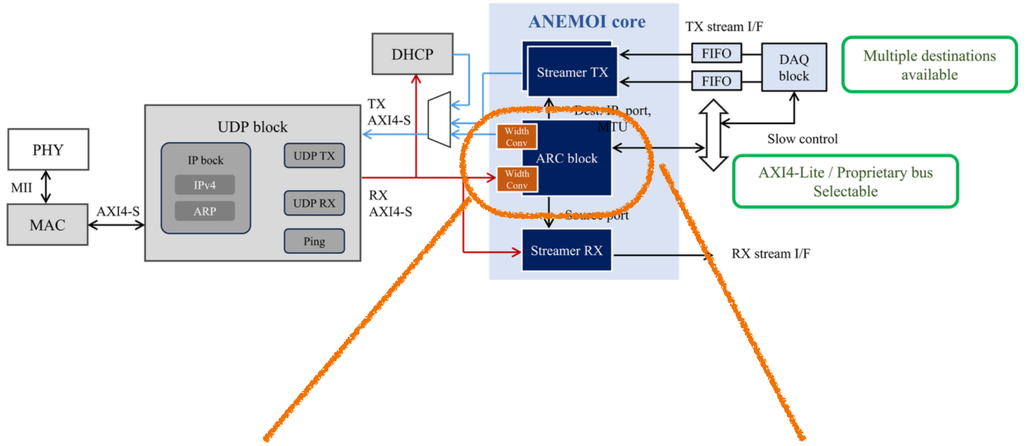
Sequential mode

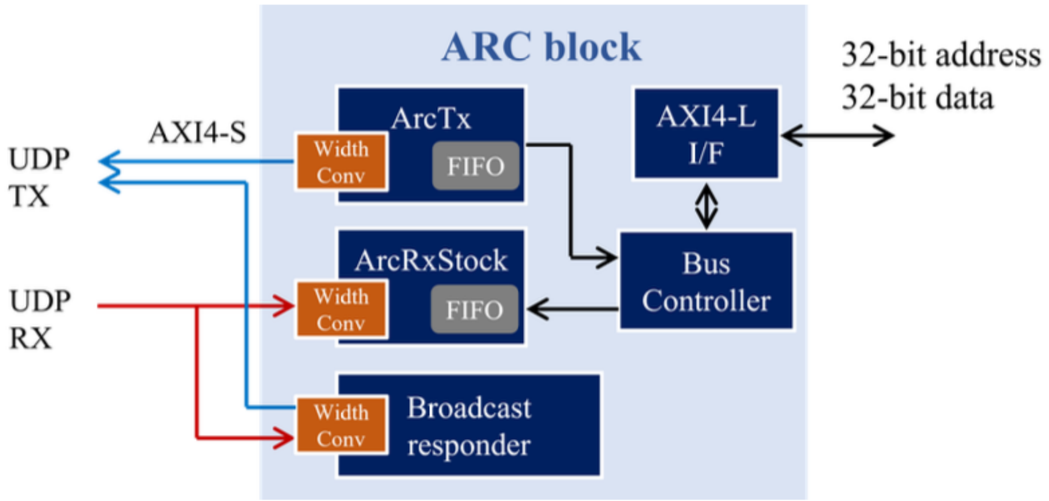
- Read / Write data at the same address
  - FIFO
- Read / Write with address auto increment
  - RAM sequential access

Header (64-bit)
Address 0
Data 0
Address 1
Data 1

List mode

- Read / Write using the address / data pair





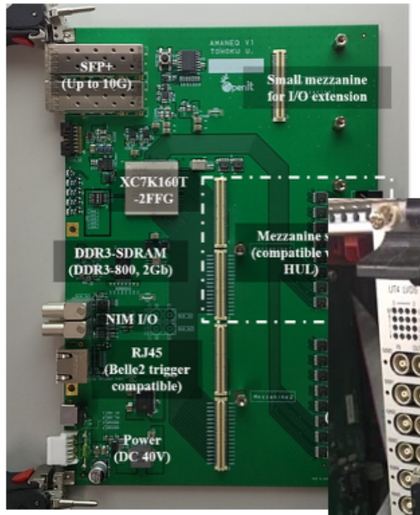
Implementation and performance

Implemented on multiple boards

Tested 1G/10G Ethernet speeds
- Directly connected to NIC (X710BM2-2SP, Intel)
- No packet losses at 1 Gbps
- Non-zero losses found at 10 Gbps

Jumbo frames improved throughput and packet loss

	Data gen. rate (Mbps)	Throughput @ PC (Mbps)	Packet loss (%)	MTU (Byte)
1G	957.7	957.7	0.00	1500
	992.9	992.9	0.00	9200
10G	9140.6	9111.3	0.13	1500
	9845.6	9845.4	0.01	9200



AMANEQ (XC7K16T-2)



UT4 (XCVU080)

CTS (Kria K26 (PL))





ELF (Efinix Trion T20Q100F3C3)
100 Mbps

Summary and future prospects

ANEMOI
- UDP-based protocol for FEEs, having data streamer and addressed register control functionalities
- Works with any UDP block that has an AXI4-Stream interface
- Tested for GbE and 10 GbE

DAQ test in real experiments

Support of 40GbE and 100GbE
- Ideally, users only need to modify the parameter (generic) to make ANEMOI operate in different link speed (1G, 10G, 40G, 100G, etc.)

Reliability
- Forward Error Correction using Reed-Solomon Coding
- Reliable UDP (re-transmission over UDP protocol)

Open-source release
- https://github.com/spadi-alliance/ANEMOI
- Nothing uploaded yet…

K Program

This work is supported by JST K-program, grant number JPMJKP24J2

Developers of ANEMOI: Ryotaro Honda (KEK), Hidetada Baba (RIKEN), Yun-Tsung Lai (KEK), Yuto Ichinohe (RIKEN)

RDMA for streaming readout

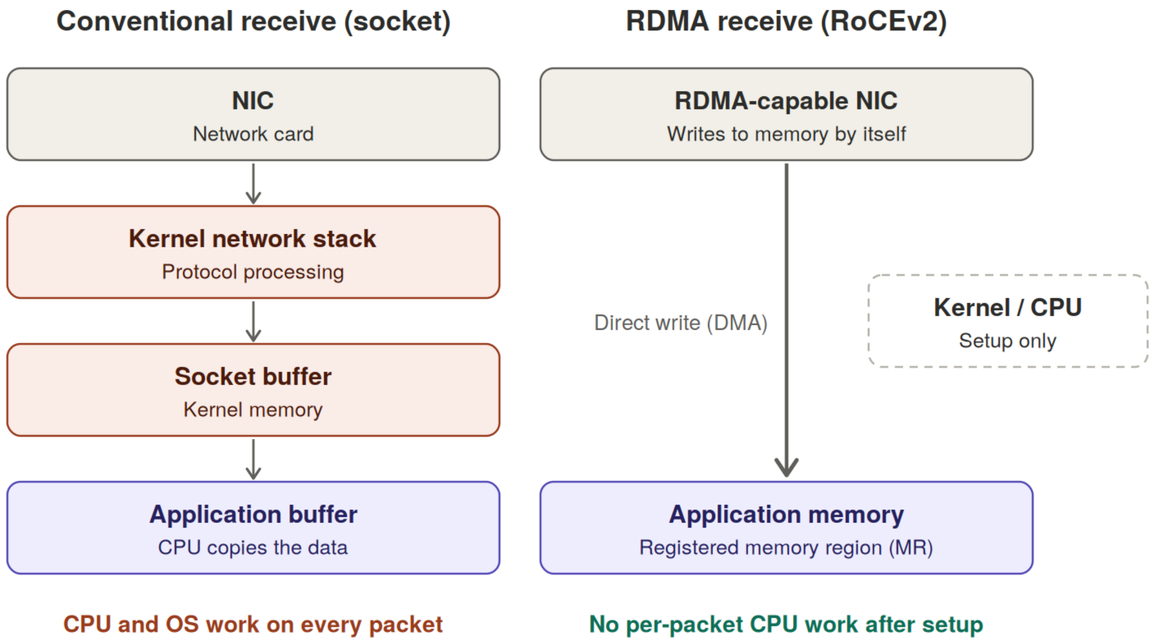
Streaming readout sends a continuous data stream to servers

Normal sockets

- With normal sockets, the kernel handles every packet, and the CPU copies the data
- CPU overheads can be a problem in high-throughput environments
  - Large experiments
  - Large data per FEE (e.g. waveforms)

RDMA (Remote Direct Memory Access)

- NIC writes the data directly into application memory
- The CPU is needed only for setup, and after that, is free for other work

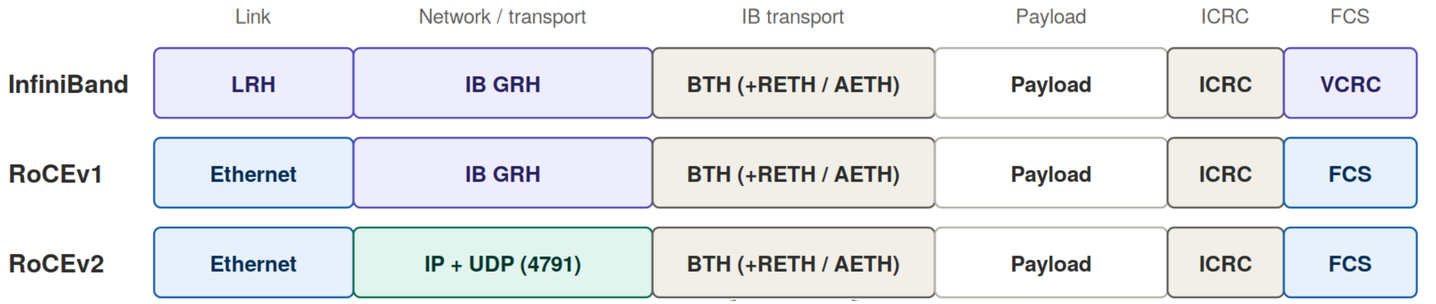


	Local memory	RDMA	TCP/IP
Bandwidth	472 Gbps	92 Gbps	66.3 Gbps (10 parallel)
Latency	0.2 us	1 us	700 us (ping)

Example performance comparison using two CX5 NICs

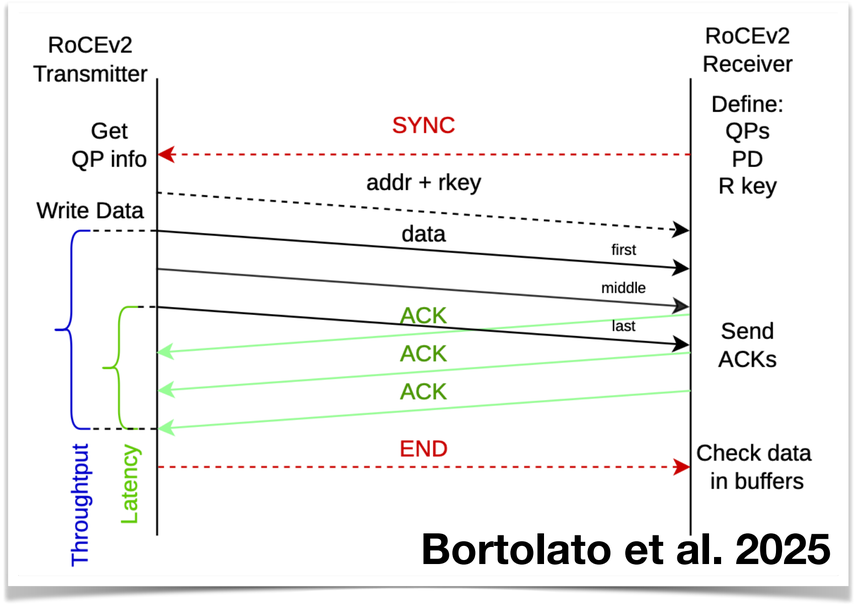
RoCEv2: RDMA over normal Ethernet

RoCE (RDMA over Converged Ethernet)
- InfiniBand: the original RDMA protocol
  - needs its own switches
- RoCEv2: InfiniBand protocol carried on UDP/IP
  - works with normal ethernet switches



RoCEv2 protocol
- Every packet has a Base Transport Header (BTH)
  - Queue Pair (QP) number, Packet Sequence Number (PSN), Opcode (RDMA SEND/WRITE etc.)

RDMA WRITE (direct write from FEE to server)
- The first packet of a WRITE also has an RDMA Extended Transport Header (RETH)
  - remote address, access key, length
- RX sends ACK. TX retransmits lost packets.
- Plain WRITE does not notify RX. RX must check memory itself.

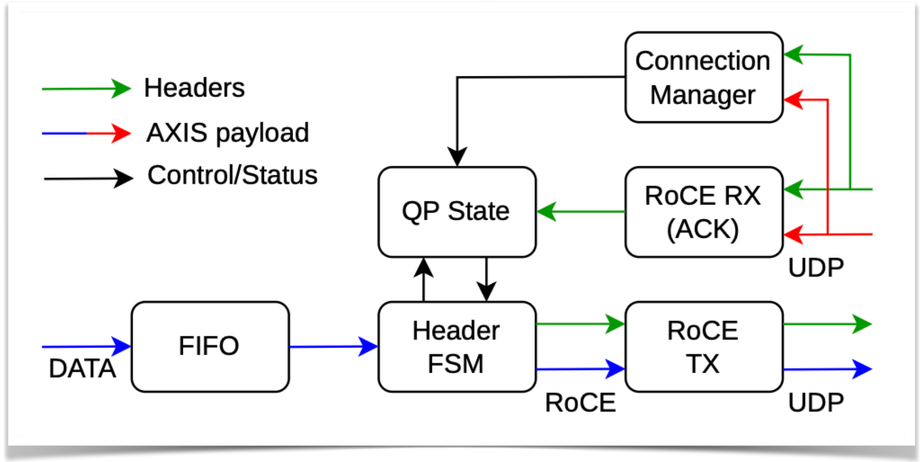


Starting point: FEROCE project

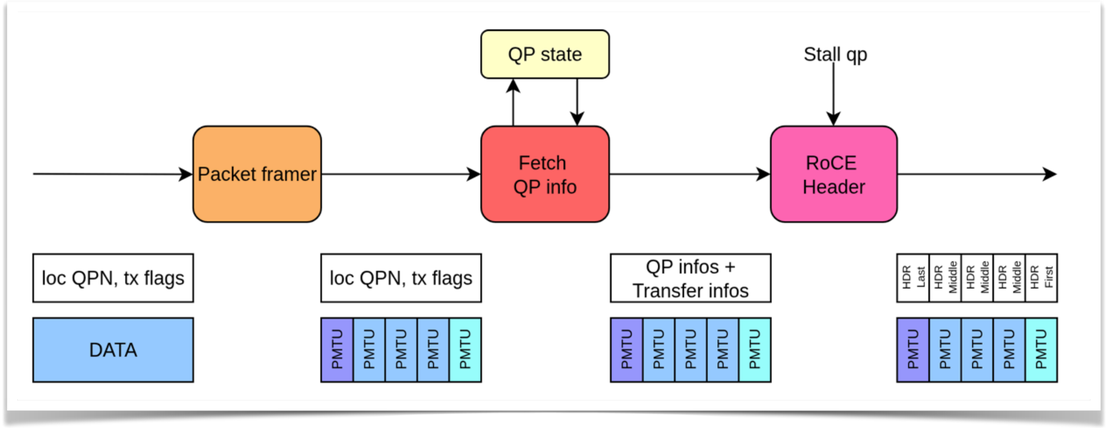
FEROCE project (Bortolato et al. 2025)
Open-source FPGA implementation of RoCEv2 TX module
https://github.com/Gabriele-bot/100G-verilog-RoCEv2-lite

- Supports RDMA WRITE/SEND with/without IMMEDIATE

- Three layers for 100 Gbps: RoCE core, 512-bit UDP/IP stack (based on Alex Forencich's verilog-ethernet), Xilinx CMAC
- The core splits data into packets, makes headers, and handles ACKs and retransmission
- The user gives a work request (where, how many bytes) with the data (AXI-Stream)



Bortolato et al. 2025



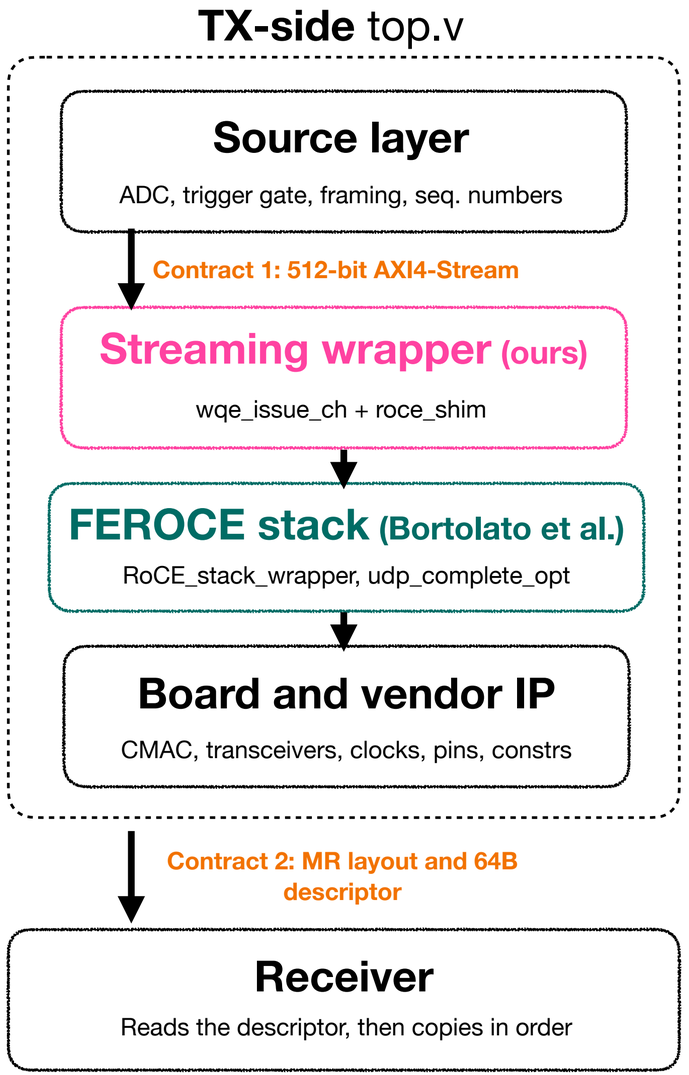
Streaming wrapper: layers

We added a thin wrapper on top of FEROCE that turns a continuous stream into RDMA WRITE requests

Four layers
- Arbitrary streaming source
- Streaming wrapper
- FEROCE stack
- Board and vendor IP

Only two contracts
- Input data: 512-bit AXI-Stream
- Receiver: agreed memory map and descriptor format

For a new board or data source, only the outer layers need to be changed



Streaming wrapper: interface

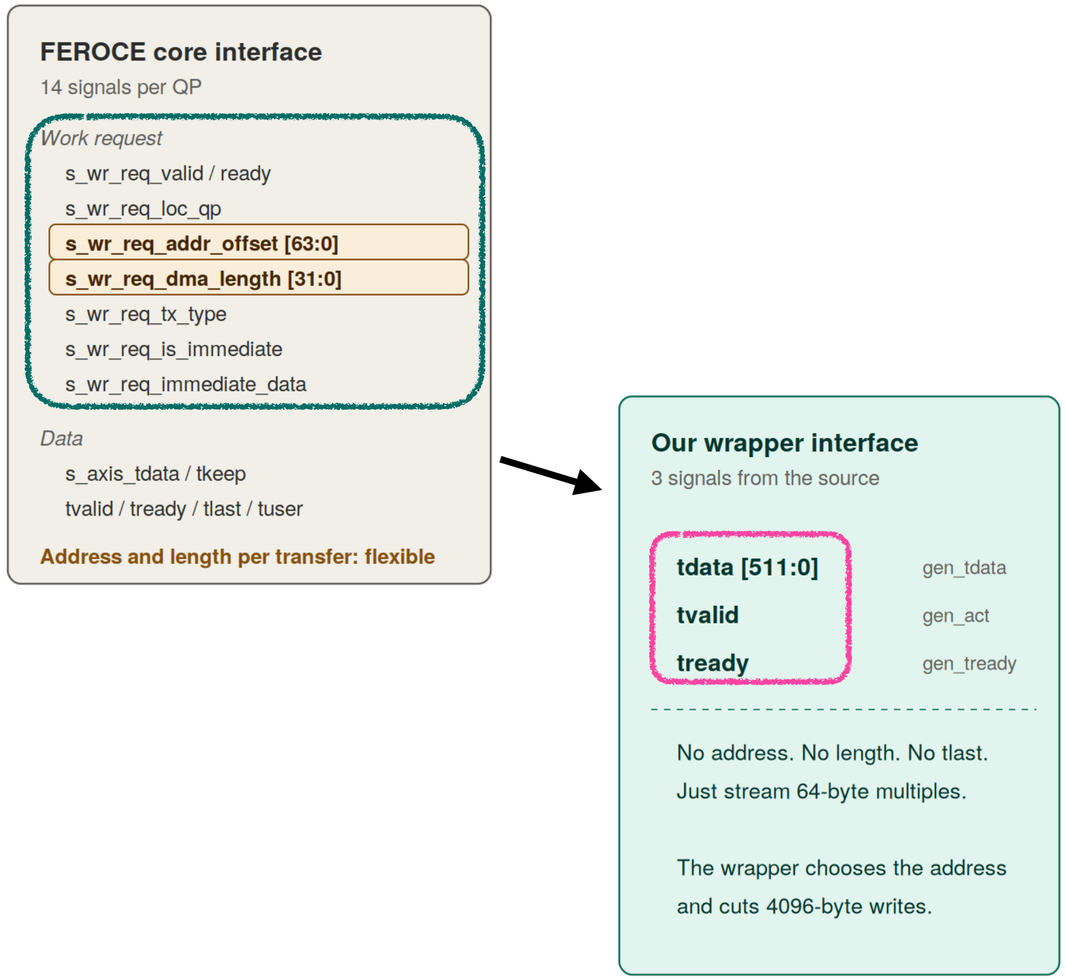
FEROCE core

- Requires work request (address, length) for every transfer
  - Accurate and flexible RDMA control

Our concept: we just create the stream and have the wrapper (+FEROCE) handle all the rest for us.

- Only three signals: tdata, tready, tvalid

- Buffers the stream and cuts it into 4096B RDMA WRITEs
- Automatically increments the write address around a ring buffer in the receiver's memory
- If data pauses, it pads the current write and records the amount of padding
- Every 16th write (tunable), and always when the stream pauses, it writes a small descriptor



Descriptors instead of WRITE IMMEDIATE

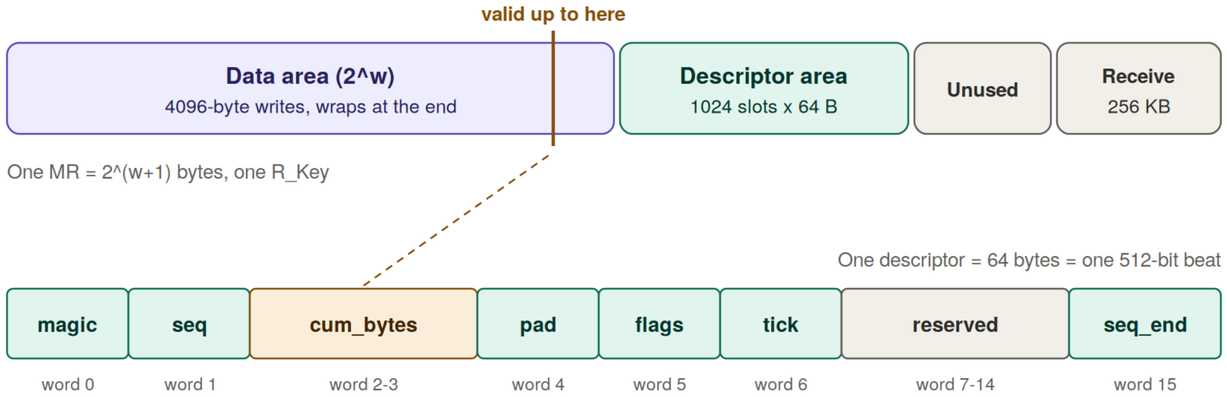
The receiver must know how far the data is valid

WRITE with IMMEDIATE

- RX must post a receive request for every message. If it falls behind, TX is stopped, or the connection fails.
- Only 4B: can’t deliver much information

Our concept: Descriptor in a separate memory area

- Sent on the same QP, after the data → arrival of the descriptor guarantees the arrival of the data
- 64B → contains sequence number, valid bytes, padding and time
- The data area stays clean: nothing is inserted into the data region
- A slow receiver program never stops the sender. If it falls too far behind, it detects an overrun, and the connection stays up.



The receiver reads data only up to cum_bytes of the newest valid descriptor.

Implementation test

Implemented on multiple boards

Alveo U50 (data-center accelerator card)
- Synthetic sources: counter pattern and pseudo-random waveforms following Poisson count rates
- 97.9 Gbps at 4096B PMTU

ZCU111 (RFSoC, 8 ADC channels, 4.096 GSPS, 12 bits)
- Synthetic sources
- Real ADC signal: 2-channel trigger/busy gate and arbiter
- 97.4 Gbps

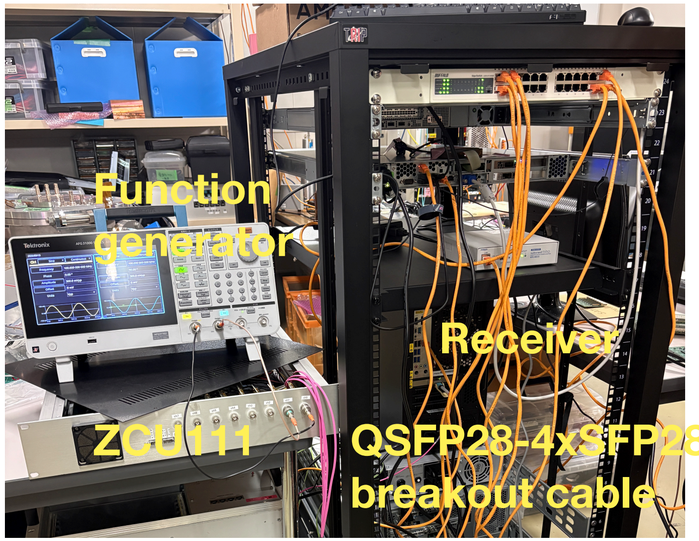
Same ceiling, no errors.
The wrapper code did not change. Only the board and source layers changed.

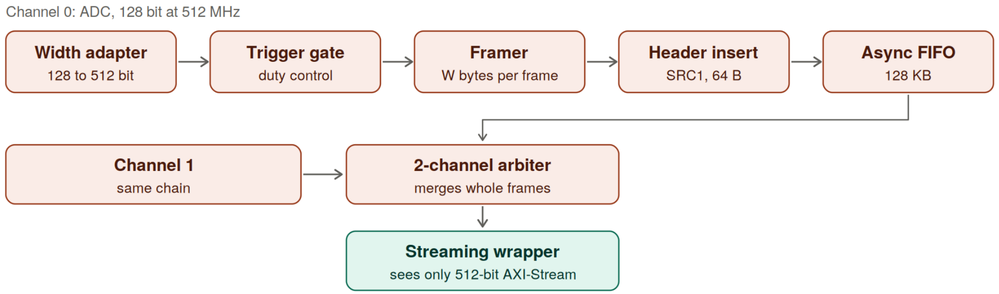
Resources
- Alveo: LUT 7.7% / FF 6.9% / BRAM 13.6% / URAM 11.1% (71 blocks)
- ZCU111: LUT 17.86% / FF 15.43% / BRAM 23.56% / URAM 88.75% (71 blocks) (includes the ADC source layer)



Alveo U50

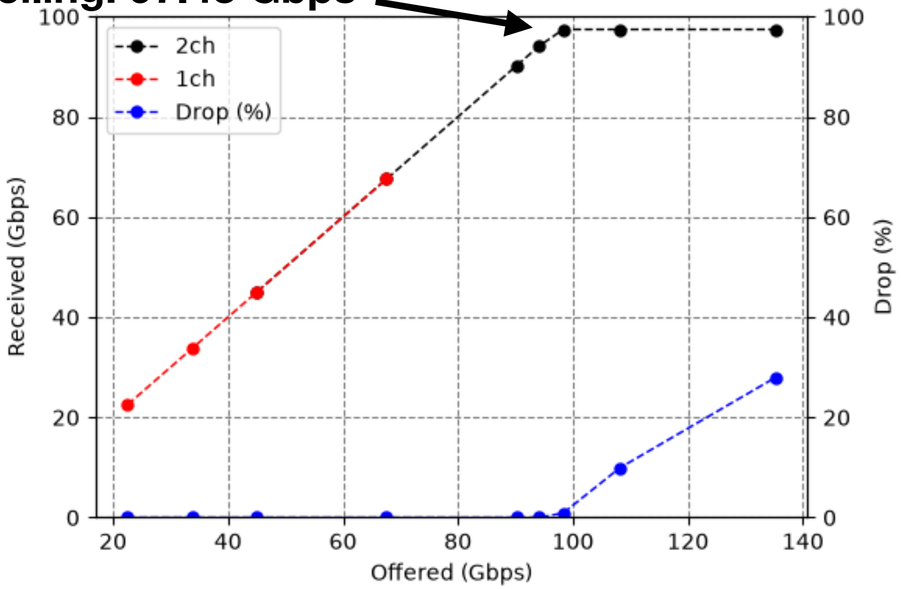
ZCU111





Performance

Ceiling: 97.43 Gbps

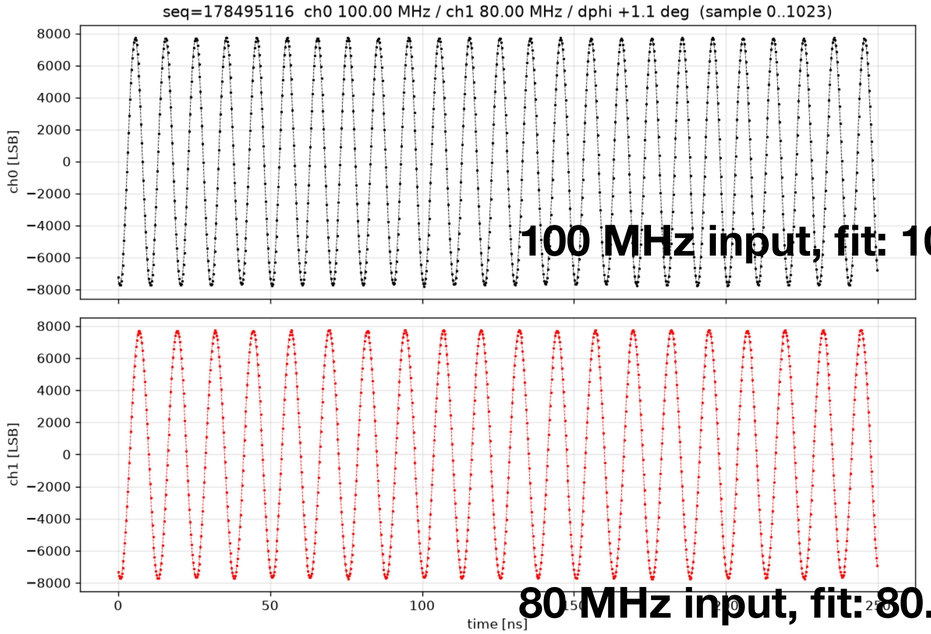


Throughput

- Zero loss up to 94.03 Gbps offered
- Ceiling: 97.43 Gbps → matches the RoCEv2 framing limit at the 100 Gbps line speed

Latency

- Wrapper input to wire: 0.66~2.03 us (estimate)
- Packet out to its ACK back: 2.4 us (measured)



100 MHz input, fit: 100.00 MHz

80 MHz input, fit: 80.00 MHz

Stability and integrity

- The receiver copied all the data (12.2 GB/s) during the runs.
- 40 runs of 10 sec at about 94 Gbps: 0 retransmissions
- 0 corrupted bytes in 1024 waveform frames (2 MB, compared byte by byte)
- Waveforms reproduced correctly

PMTU (offered: 90.11)

- 4096B: 90.11 Gbps, 0 loss (needs jumbo frames)
- 2048B: 90.12 Gbps, 0 loss (needs jumbo frames)
- 1024B: 89.63 Gbps ceiling (fits a standard 1500B MTU)

Summary and future prospects

Summary

- We built a streaming wrapper on top of FEROCE
- A source only needs a 512-bit AXI-Stream. The wrapper handles addresses, padding and descriptors.
- The same code runs on Alveo U50 and ZCU111
- The ceiling of 97.43 Gbps matches the RoCEv2 framing limit at the 100 Gbps line speed

Package the wrapper as a library

- We plan to contact Gabriele Bortolato's group

General-purpose digitizer module

- Still an idea, not projectized yet
- 100Gbps RDMA as a standard feature of the firmware?

We thank G. Bortolato and the FEROCE team for the open-source core.

G. Bortolato et al., FEROCE: Front-End RDMA Over Converged Ethernet, a lightweight RoCE endpoint, EPJ Web Conf. **337**, 01184 (2025)

Overall summary

ANEMOI
- UDP-based protocol for FEEs, having data streamer and addressed register control functionalities
- Works with any UDP block that has an AXI4-Stream interface
- Tested for GbE and 10 GbE

DAQ test in real experiments

Support of 40GbE and 100GbE
- Ideally, users only need to modify the parameter (generic) to make ANEMOI operate in different link speed (1G, 10G, 40G, 100G, etc.)

Reliability
- Forward Error Correction using Reed-Solomon Coding
- Reliable UDP (re-transmission over UDP protocol)

Open-source release
- https://github.com/spadi-alliance/ANEMOI
- Nothing uploaded yet…

100 Gbps RDMA
- We built a streaming wrapper on top of FEROCE
- A source only needs a 512-bit AXI-Stream. The wrapper handles addresses, padding and descriptors.
- The same code runs on Alveo U50 and ZCU111
- The ceiling of 97.43 Gbps matches the RoCEv2 framing limit at the 100 Gbps line speed

Package the wrapper as a library
- We plan to contact Gabriele Bortolato's group

General-purpose digitizer module
- Still an idea, not projectized yet
- 100Gbps RDMA as a standard feature of the firmware?

K Program

This work is supported by JST K-program, grant number JPMJKP24J2

We thank G. Bortolato and the FEROCE team for the open-source core.