Research Progress of RDMA

Sheng Dong
dongsheng@ihep.ac.cn
July 10, 2025

中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

Outline

- RDMA and ROCEv2
- Firmware Development
- Preliminary Result
- Summary and Plan

RDMA vs TCP/IP

- TCP/IP

  第一次拷贝（CPU执行：用户空间 → 内核空间）

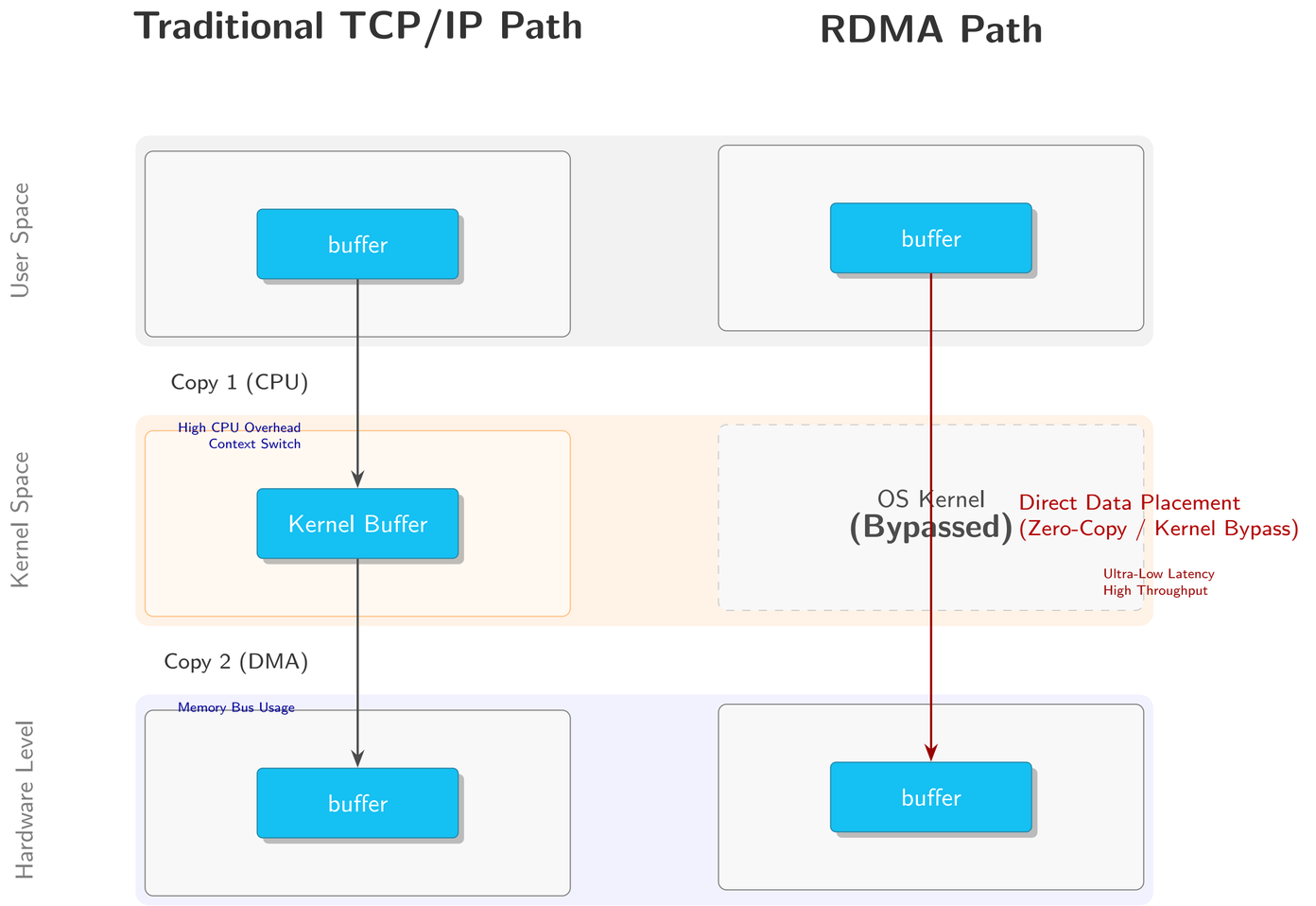
  - 使用CPU将你的数据从应用程序的缓冲区，完整地拷贝一份到内核空间的缓冲区

  第二次拷贝（DMA执行：内核空间 → 网卡缓冲区）

  - 网卡驱动程序会指示DMA（Direct Memory Access）控制器，将准备好的数据包从内核缓冲区，再次拷贝一份到网卡自己的板载物理缓冲区

- RDMA

  通过内核旁路技术，它允许应用程序直接指示DMA控制器，在用户空间缓冲区和网卡缓冲区之间进行数据传输，从而完全消除了上述两次CPU和内存密集型的拷贝

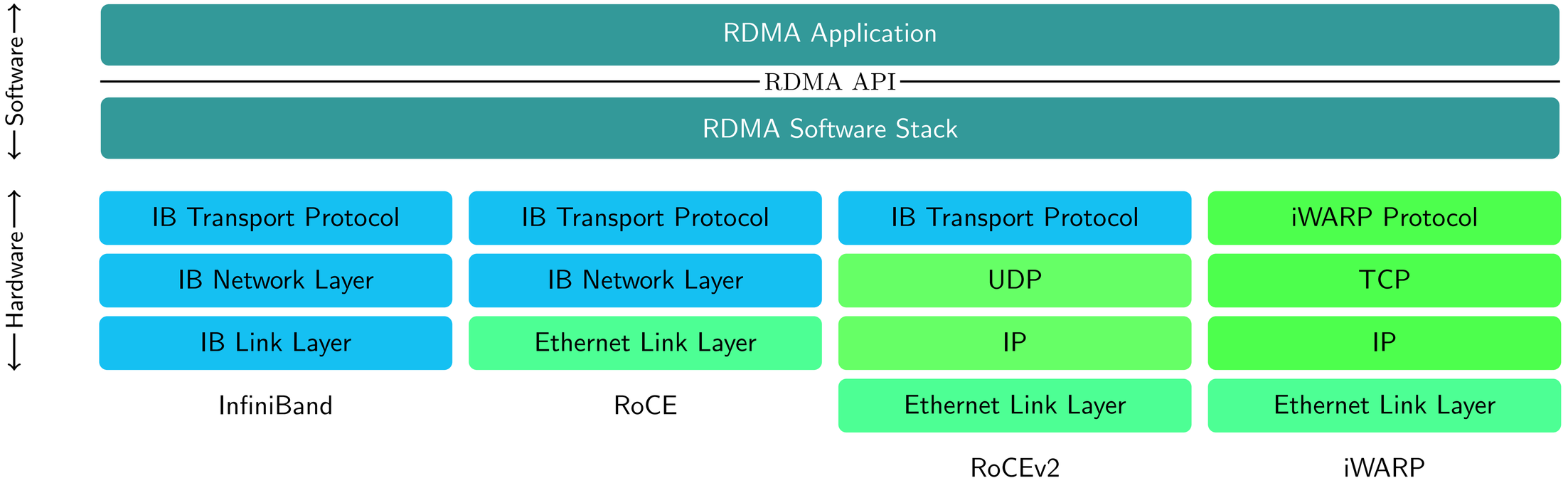


RDMA Stack

特性	InfiniBand	RoCEv1	RoCEv2	iWARP
底层传输	原生RDMA协议	以太网二层	UDP/IP (以太网三层)	TCP/IP
HEP主要优势	最低延迟、高吞吐量、成熟的CPU卸载	利用以太网硬件、较InfiniBand低成本	利用以太网硬件、可路由、兼容现有IP网络、成本效益	利用成熟TCP/IP、无需无损以太网、广域网适应性较好
HEP主要劣势/挑战	专用硬件、成本较高、广域网复杂性	依赖无损以太网 (L2)、不可路由	依赖无损以太网 (PFC配置复杂)、相比IB可能有略高延迟	TCP处理开销、相比IB/RoCE延迟可能较高
HEP主要应用/研究	事件组建 (CMS Run-2)、HPC集群、ALICE EPN Farm	早期探索	DAQ (ESRF RASHPA 、DUNE FPGA)、事件组建 (CMS Run-3 、LHCb评估)、SmartNIC集成	特定集群网络 (CERN案例)

- 基于复杂度、成本、性能以及主流研究方向的综合考虑，选择ROCEv2作为研究方向

RDMA Stack



- RoCE和RoCEv2本质都是IB协议的扩展和改进
- RoCE系列方案是为了兼容以太网基础设施，实现IB的高性能RDMA能力而产生的技术方案
- RoCEv2通过增加IP层，实现了跨网段通信与路由功能，使其在以太网中更易部署、更具灵活性，因此成为主流的RDMA方案

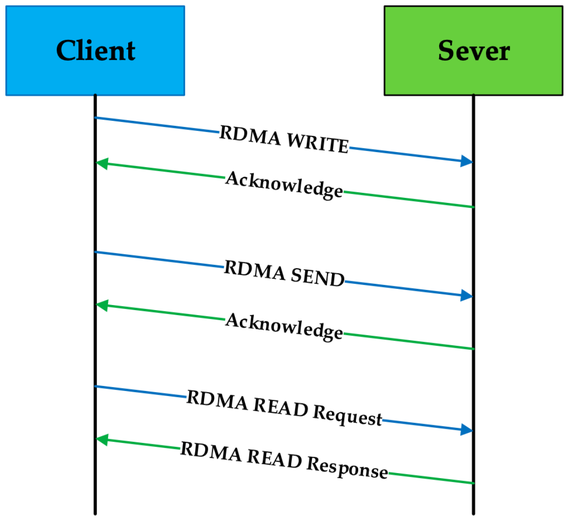
RoCEv2 Connection Mode

对比项	RC（可靠连接）	UD（不可靠数据报）
连接类型	点对点 QP 配对	任意目标，动态指定地址
传输操作支持	Send / Write / Read / Atomic 全支持	仅 Send / Receive
可靠性	✅ 有 ACK、重传、顺序控制	❌ 无保障
延迟	稍高（需握手）	更低（适合控制面）
应用场景	数据通道、存储读写、触发分发等	控制信息、广播、名称解析等

链路层：基于PFC/ECN（优先流控/显示拥塞通知）等协议实现无损以太网（不丢包）

协议层：

1. PSN（Packet Sequence Number） 保证包按序接收、检测丢包；
2. ACK/NACK 应答机制 用于确认成功或请求重传；
3. 超时重传：发送端未在规定时间内收到 ACK 会自动重发；
4. RNR（Receiver Not Ready）控制，避免接收方队列溢出；
5. ICRC 校验 确保数据包内容完整性。



RoCEv2 on FPGA

- ~~CMAC + ERNIC~~
- 基于两个开源项目学习和开发

RDMA exploration

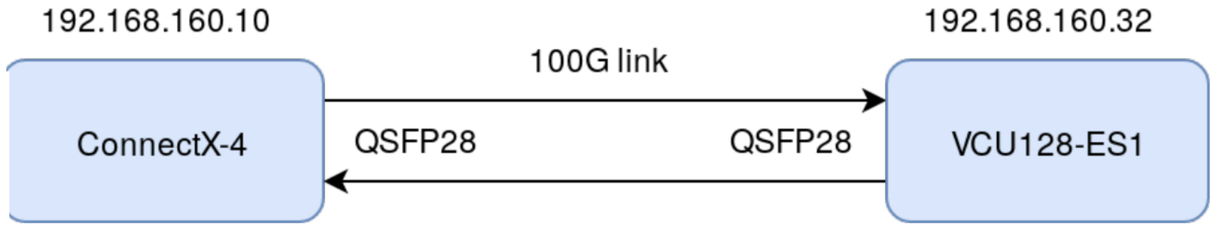
1 Introduction

This brief introduction covers the motivation of this work, providing a general overview on the project and describing the structure of this report.

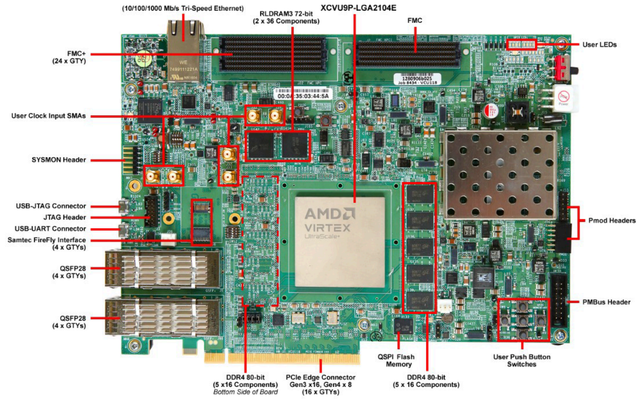
Currently the readout system for Felix Phase I has succesfully been developed. Now Phase II is in development and the desire for a higher speed readout system at the Atlas project is increasing steadily. The current data transfer protocols in use are supporting data rates up to about 10 Gbps. Combining multiple of these links in parallel results in huge data transfers but a more efficient approach would be to use less links which support higher data rates.

Nowadays Ethernet link supporting data rates of 100 Gbps and even 400 Gbps are standardized in data centers and all technology to make use of these data rates is available. Such data rates are very desirable to implement in the FELIX project. However to support such rates there has to be a technology used called RDMA (Remote DMA).

This document will describe how to implement this RDMA technology to establish a 100 Gbps link in between a sever (x86) and a FPGA (Xilinx VU37P)



Connectx-5



VCU118

VCU128 -» VCU118
ConnectX-4 -» ConnectX-5

徐畅正在此基础上研究工作
https://gitlab.cern.ch/nboukadi/rdma

RoCEv2 on FPGA

- ~~CMAC + ERNIC~~
- 基于两个开源项目学习和开发

Implementing RoCEv2 in Verilog for FPGA

Gabriele Bortolato[a,b,c], Matteo Migliorini[c], Andrea Triossi[a,b]

[a]INFN sez. Padova, [b]DFA Padova University, [c]CERN

Finanziato dall'Unione europea NextGenerationEU

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

CERN

UNIVERSITAS STUDII PADUANI MCCXXII

INFN Istituto Nazionale di Fisica Nucleare

VCU118 + ConnectX-5

TCP/IP -» RoCEv2

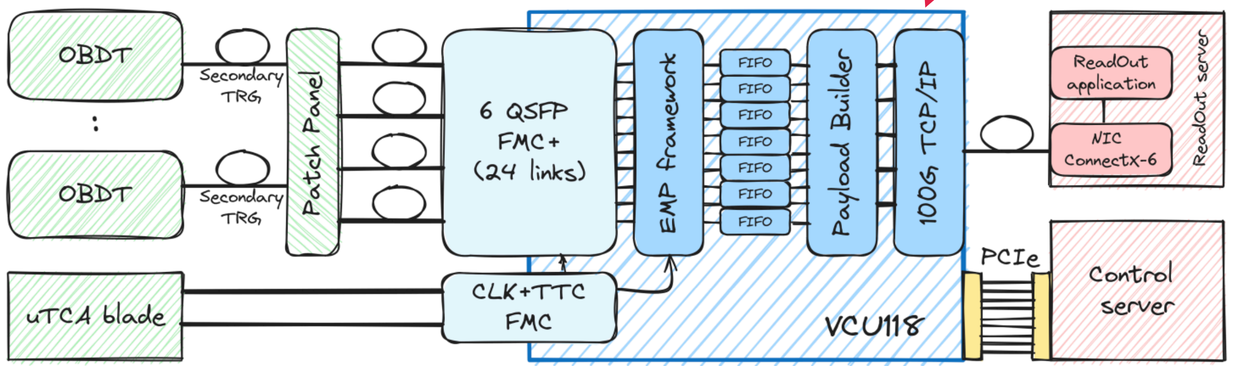


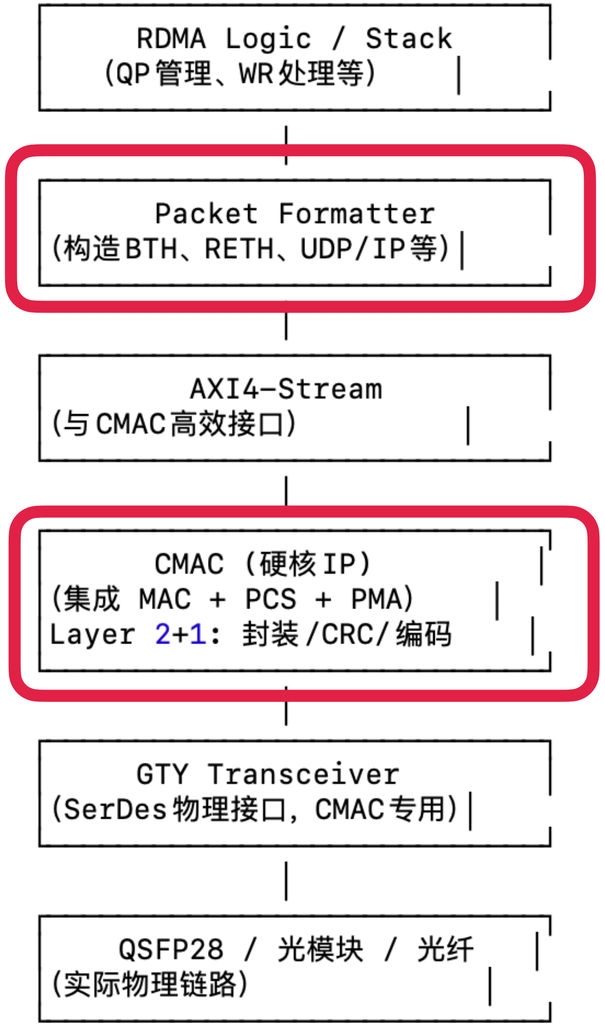
Figure 1. Schematic representation of the DT 40 MHz readout system. From left to right: the connection to the CMS TCDS and secondary lpGBT links from the OBDTs are received by a VCU118. A simplified illustration of its main components is shown in the middle box. The firmware is based on the CMS EMP framework, connected via a set of FIFOs and a payload builder process to a 100G TCP/IP module. Board and firmware are monitored and controlled via PCIe. A second server is used to receive and process the TCP/IP stream.

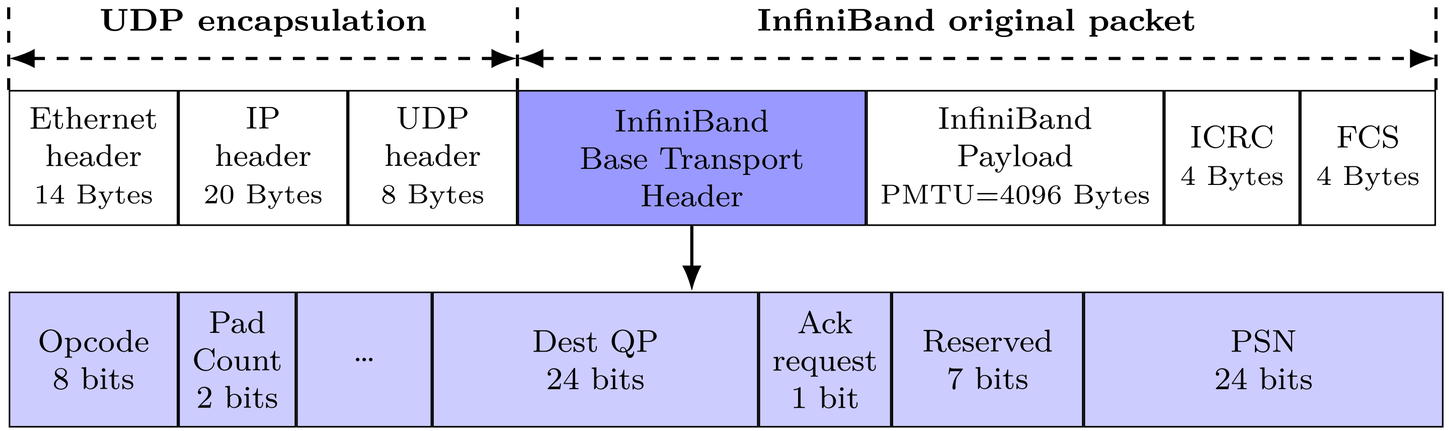
40 MHz triggerless readout of the CMS Drift Tube muon detector

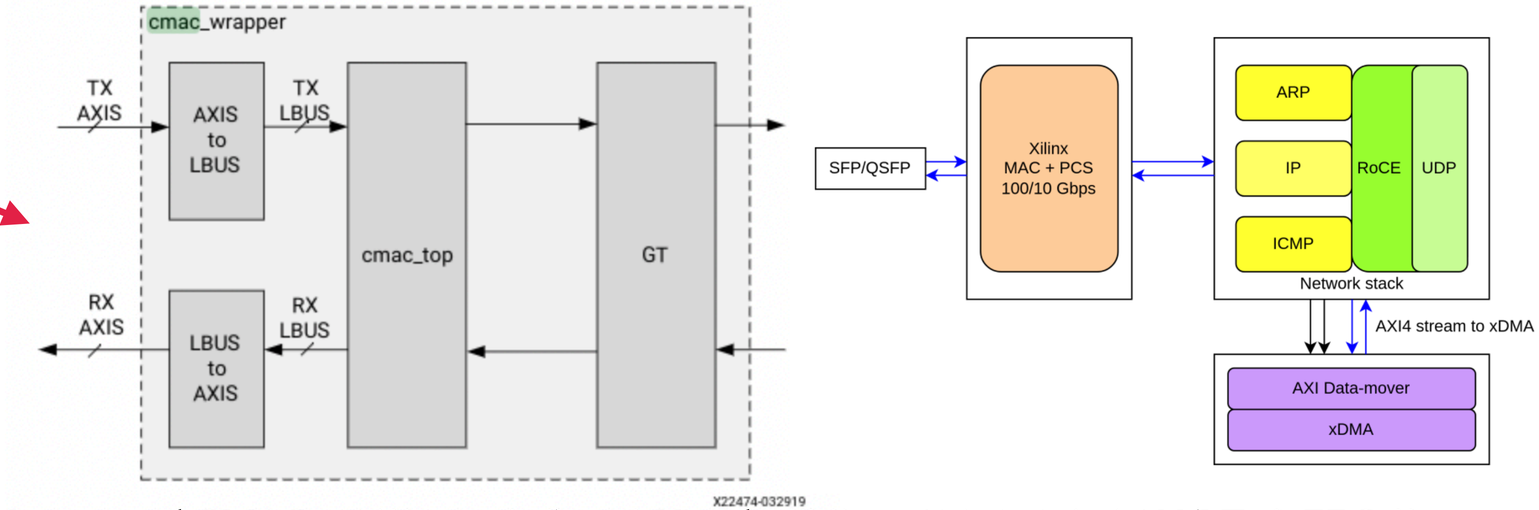
DOI 10.1088/1748-0221/19/02/C02050

https://github.com/Gabriele-bot/100G-verilog-RoCEv2-lite

Firmware Structure



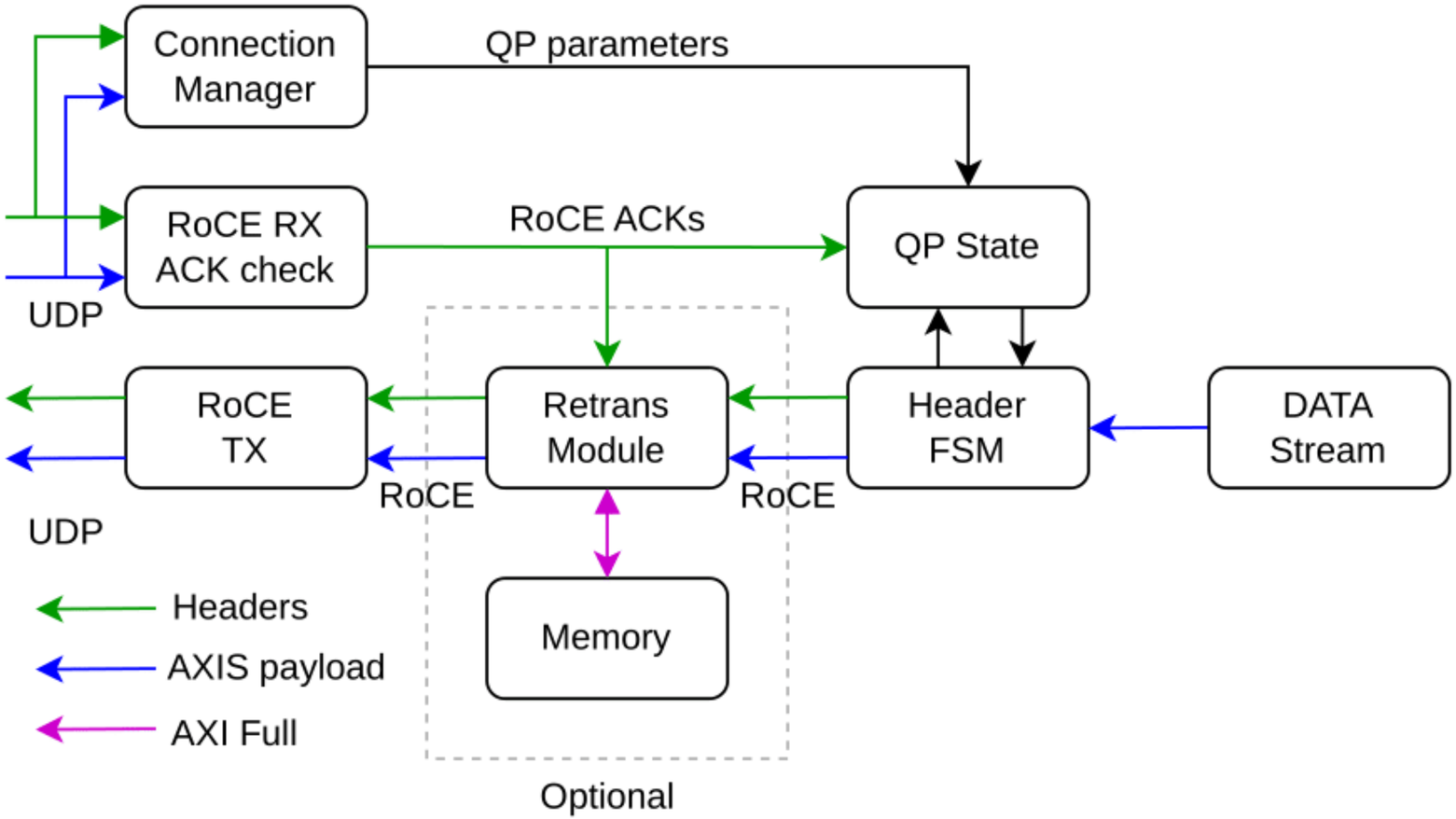




- Integrated CMAC（硬核）：免费、随 Vivado 搭载，适用于不需要 AN/LT 或 FEC 的应用；
- 附加功能（AN/LT、FEC）：需要额外购买对应许可证，如 RS-FEC 或 AN/LT，否则相关功能无法使用。

Firmware Design

- 仅 TX
- RoCEv2 端点的超级精简版本
- 仅支持 RC RDMA WRITE 和 RC SEND（带或不带 IMMEDIATE）
- RX 部分仅用于读取 ACK 和 NAK。



Connection Manager

交换 QP 参数，将 QP 从 RESET 状态切换 INIT→RTR→RTS状态，三种方式：

1. 手动带外 (OOB) 设置：通过 TCP、UDP 或任何其他通道

   最常见；交换 QP参数；各自配置好 QP 状态机

2. RDMA 连接管理器 (librdmacm / RDMA CM)

   基于 IP 层的 rdma_cm 库（librdmacm），实现自动 QP 参数协商。由内核/用户库自动完成参数交换和 QP 状态切换，无需手动操作 QP。

3. InfiniBand 连接管理器 (IB CM)

   只在极少数高阶 IB/RoCE环境下使用，比如自定义管理协议；实际应用很少，主要用于自研管理系统。

QP参数：

QP Number
Initial PSN - 初始包序列号
Base Address - RDMA Memory Region起始地址
Remote Key - Memory Region 远程密钥
GID - 全局标识符，IP Address （RoCEv2）

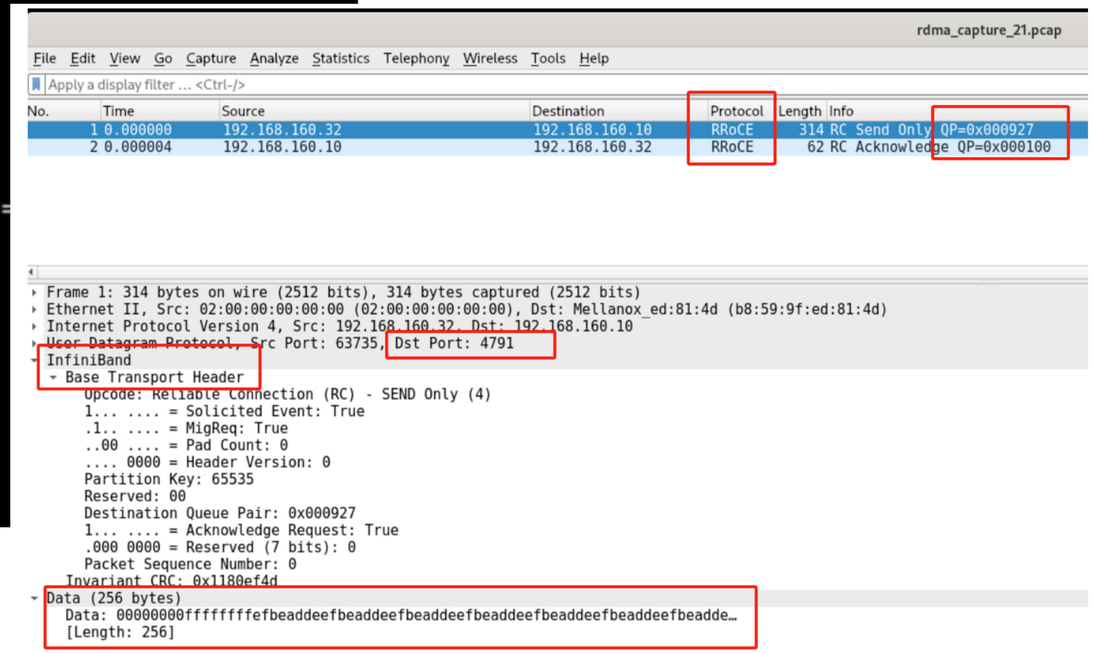
当前的方式OOB（UDP）：

1. 固件固定一个QPN (256)
2. 开发基于libibverbs的软件，建立QP同时切换Q状态至RTS，并通过UDP端口（17185）向固件的CM模块发送QP信息
3. 同样通过UDP，发送控制指令，使FPGA的QP状态转换至RTS
4. 自此QP连接建立，FPGA可向RNIC发送数据。

Capture Package

```
[dongs@hd09 rdma]$ ibdev2netdev
mlx5_0 port 1 ==> ens4f0np0 (Down)
mlx5_1 port 1 ==> ens4f1np1 (Up)
[dongs@hd09 rdma]$ sudo ibdump -d mlx5_1 -i 1
Initiating resources ...
searching for IB devices in host
Port active_mtu=4096
MR was registered with addr=0x192c910, lkey=0x1c0b, rkey=
 ------------------------------------------------
 Device                         : "mlx5_1"
 Physical port                  : 1
 Link layer                     : Ethernet
 Dump file                      : sniffer.pcap
 Sniffer WQEs (max burst size)  : 4096
 ------------------------------------------------

Failed to set port sniffer1: Invalid opcode
```



- 网卡驱动原生抓包工具（ibdump）
  - Nvidia Employee: We removed support for Offloaded Traffic Sniffer feature, but we can suggest using the docker-container solution to use RDMA devices, in order to capture and analyze RDMA packets using tcpdump.
- 用 tcpdump 命令行抓取 udp 4791 端口流量，写入 pcap 文件，然后用 wireshark 打开文件分析

Software

▸ **服务端（PC）建立 QP**

1. 打开 RDMA 设备并分配 Protection Domain (PD)
2. 分配 Completion Queue (CQ)
3. 创建 Queue Pair (QP)
4. 注册内存区域（Memory Region, MR）
5. 初始化 QP 状态：RESET → INIT，INIT → RTR，RTR → RTS

https://code.ihep.ac.cn/rdma/qp-server

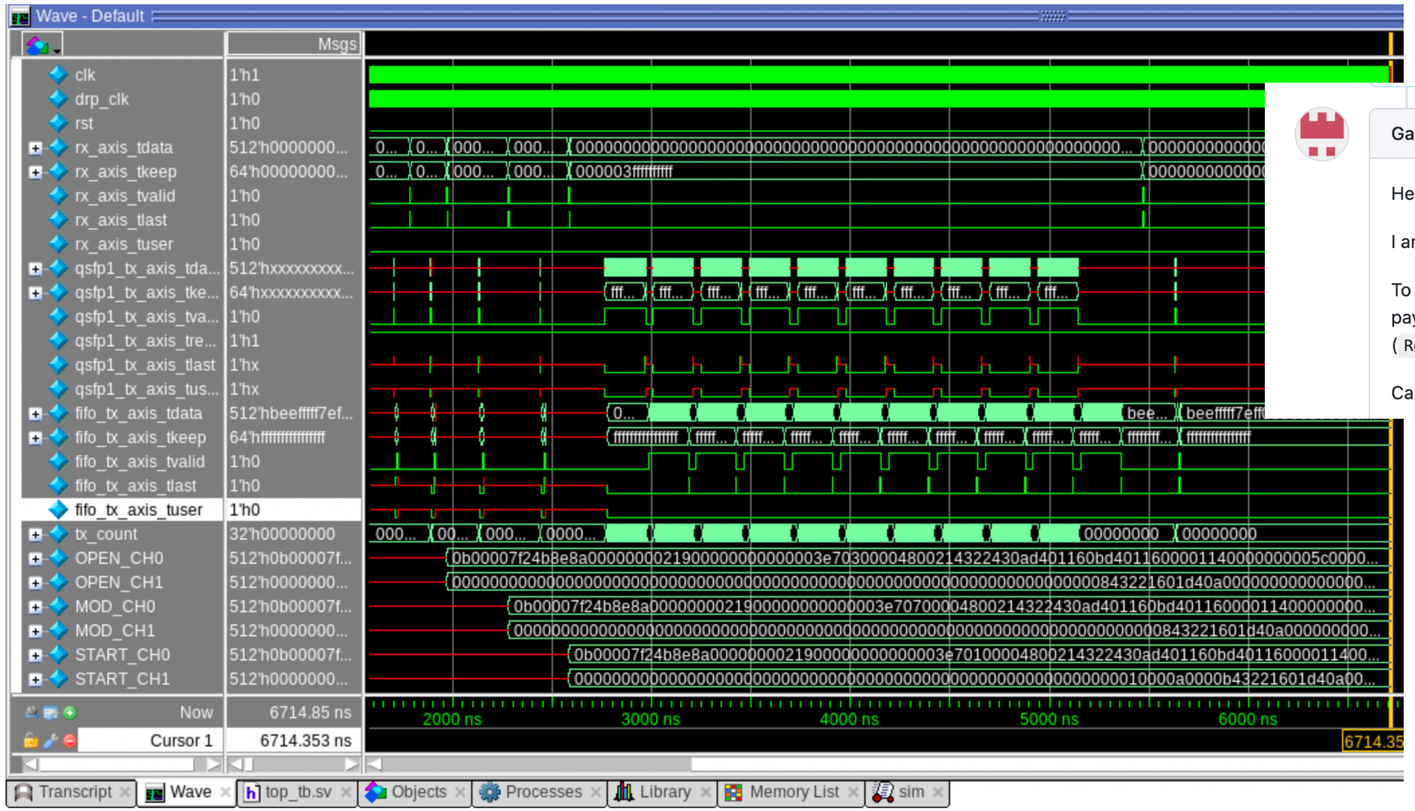
▸ **同步QP信息以及控制FPGA的QP状态**

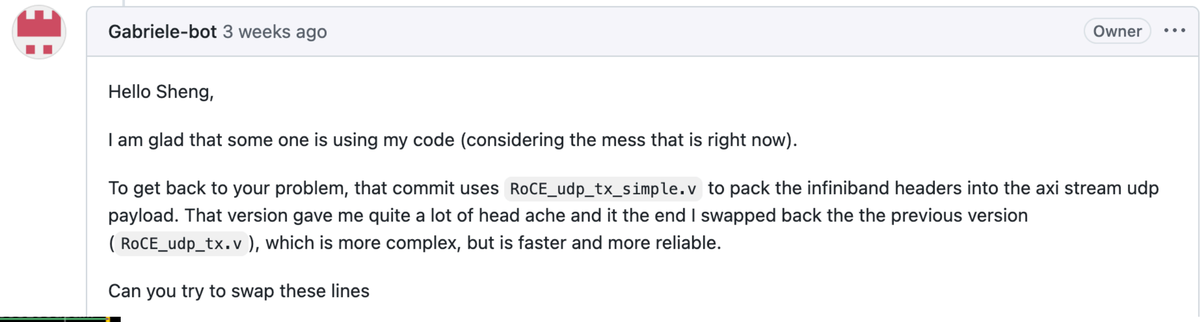
发送连接信息, 向非 4791 的 UDP 端口发送固定帧格式的 QP 连接信息 （QPN/PSN/MEM Base ADDR/MEM Key）

通过 Connection manager 使 FPGA 的 QP 也转为 RTS 状态 (Python Script, UDP socket)

DEBUG - Continuous Read/Write

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.160.32	192.168.160.10	RRoCE	74	RC Send Only Immediate QP=0x0009a6
2	0.000004	192.168.160.10	192.168.160.32	UDP	106	44226 → 17185 Len=64
3	0.000004	192.168.160.10	192.168.160.32	RRoCE	62	RC Acknowledge QP=0x000100
4	4.702509	192.168.160.10	192.168.160.32	ICMP	98	Echo (ping) request id=0x00f5, seq=1/256, ttl=64 (no response found!)
5	5.113446	Mellanox_ed:81:4d	02:00:00:00:00:00	ARP	60	Who has 192.168.160.32? Tell 192.168.160.10
6	5.753488	192.168.160.10	192.168.160.32	ICMP	98	Echo (ping) request id=0x00f5, seq=2/512, ttl=64 (no response found!)





Gabriele-bot 3 weeks ago — Owner

Hello Sheng,

I am glad that some one is using my code (considering the mess that is right now).

To get back to your problem, that commit uses `RoCE_udp_tx_simple.v` to pack the infiniband headers into the axi stream udp payload. That version gave me quite a lot of head ache and it the end I swapped back the the previous version (`RoCE_udp_tx.v`), which is more complex, but is faster and more reliable.

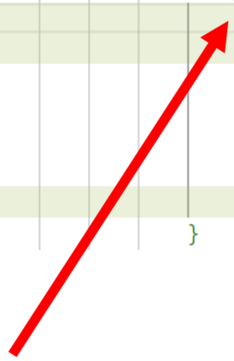
Can you try to swap these lines

- 通过仿真（模拟AXI总线数据，ping/arp/udp）定位问题，消息（Message）分包逻辑出现问题，状态机卡死
- 提交Issue，开发者回复解决
- 另外解决了Write出错问题（iCRC没有被正确的填充）
- 相比ILA调试，仿真发现以及解决固件问题的效率高的多！

Software Trick

- 吞吐率从 39Gbps 提升到 93.5Gbps

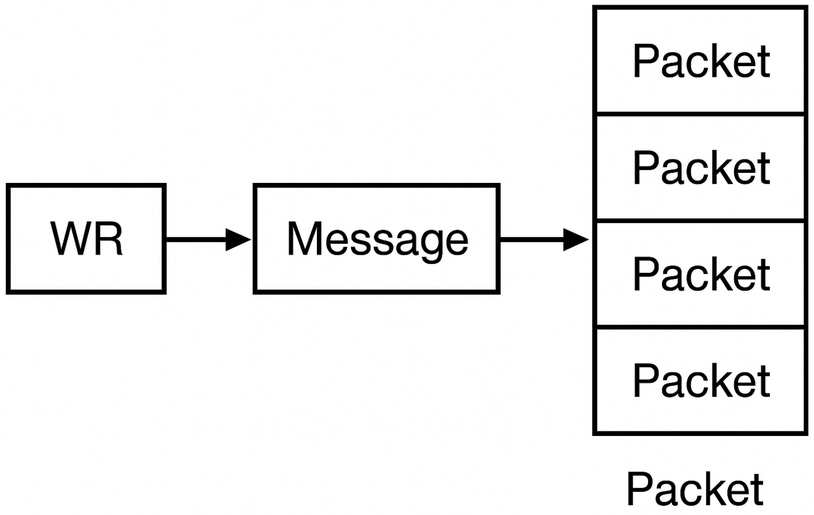
```
631             if (xfer_len > 0 && !m_write_immediately) {
632+                // std::vector<char>& buf = m_recent_received_data[m_recent_data_index];
633+                // buf.assign(slot.ptr, slot.ptr + xfer_len);
634                 if (m_recent_data_count < MAX_STORED_MSGS) {
635                     m_recent_data_count++;
636                 }
637                 size_t idx_print = m_recent_data_index;
638+                // m_recent_data_index = (m_recent_data_index + 1) % MAX_STORED_MSGS;
639             }
```

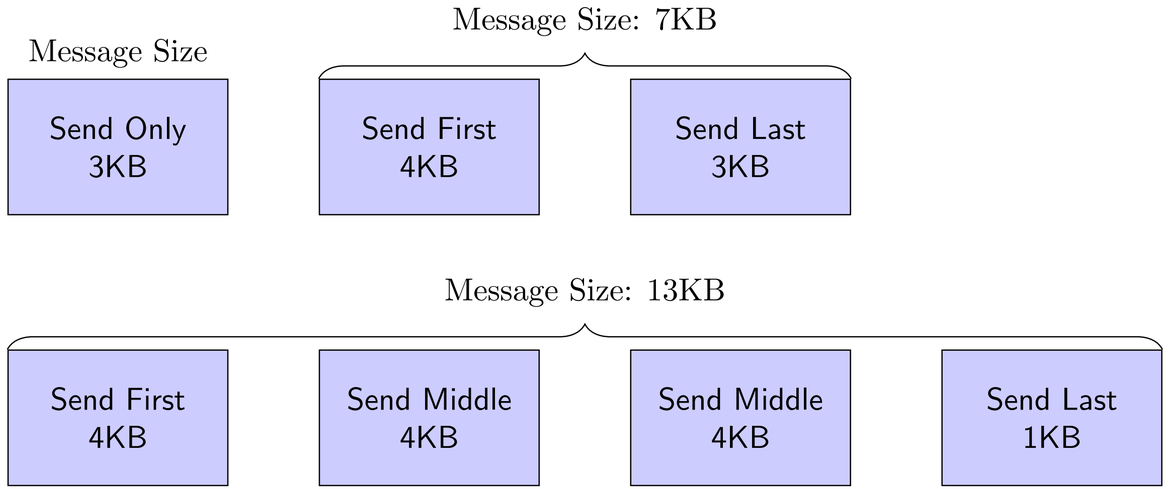


```
[dongs@hd09 build]$ numactl --cpunodebind=0 --membind=0 ./rdma_app --recv_op write --msg_size 2048000 --num_wrs 2000
RDMA Application (C++ Threaded & Parameterized) starting...
--- Effective Configuration ---
Device: rocep94s0f1, Port: 1
Local SGID Index: 3 (CRITICAL: User must verify this!)
Remote IP: 192.168.160.32, Remote QPN: 0x100 (256), Remote Initial PSN: 0
Local Initial SQ PSN: 0
Buffer Size: 10485760000 bytes, Num WRs: 2000, Message Size: 2048000 bytes
Path MTU: 4096 bytes
Receive operation: write
Stream data to file: no
Debug mode: off
-----------------------------
RdmaManager instance created.
  Device: rocep94s0f1, Port: 1, Target SGID Index: 3
  Remote Target: 192.168.160.32, Remote QPN: 0x100, Remote Initial PSN: 0
  Local Initial SQ PSN: 0
  Path MTU set to: 4096 bytes
  Receiving operation type: write
Using RDMA device: rocep94s0f1
Port 1: State: active, LinkLayer: Ethernet, Active MTU: 4096 bytes, Using path MTU: 4096 bytes, LID: 0x0, GID table length: 256, Link rate: 100 Gbps
  Using SGID at index 3: 0000:0000:0000:0000:0000:ffff:c0a8:a00a
PD allocated.
Main buffer (10485760000 bytes) allocated and initialized with 0x77.
Memory region (MR) registered:
  Buffer Address (main_buffer_ptr): 0x7f6824800000 (Decimal: 140085265694720)
  MR Buffer Address (main_mr->addr):  0x7f6824800000 (Decimal: 140085265694720)
  MR Length:                          10485760000 bytes
  MR LKey:                            0x5948
  MR RKey:                            0x5948 (22856)  <-- Remote will use this RKey
CQ created with 4000 entries.
QP created with system-assigned QPN: 0xa8b (2699)
QP state changed to INIT.
QP state changed to RTR.
QP state changed to RTS. Ready for RDMA operations!
2000 initial receive WRs posted.
Connection parameters written to 'rdma_params.json'.
CQ polling thread launched.
Main thread: CQ polling thread started.
Send Ctrl+C/kill signal to request shutdown.
[CQ Thread] Started in busy polling mode on CPU 0. Local QP 0xa8b
Data received: 78125 MB in 6.68475 s (11687.1 MB/s).
```

- 原因：接收软件代码中存在内存复制
- **最大吞吐：93.5 Gbps**

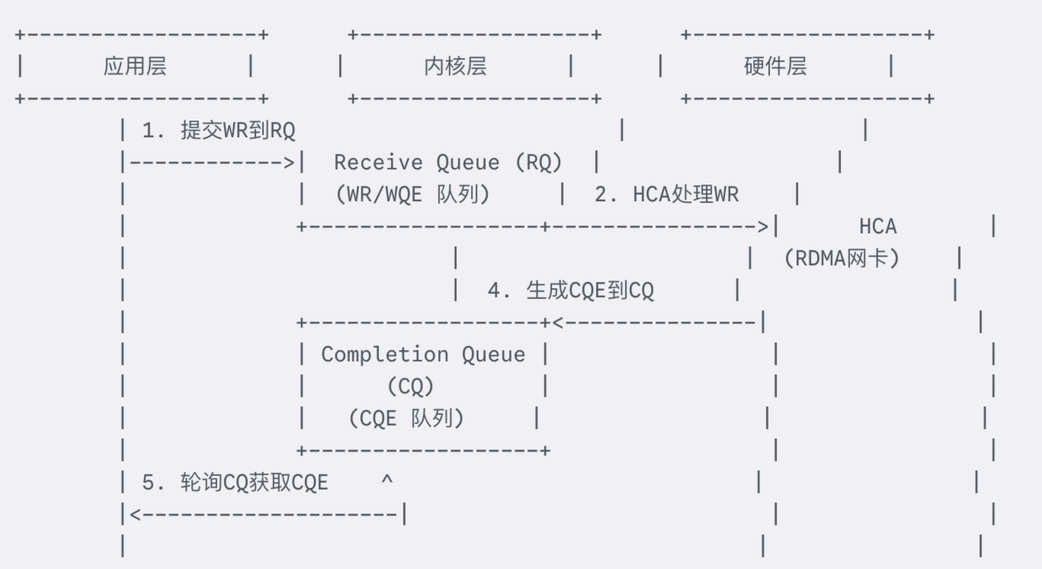
DEBUG - Throughput





- 一条 WR 代表一个 RDMA 操作，处理一个完整的 Message，这个 Message 是一段连续的数据区域，最终会被拆分成多个 Packet 发送
- 目前固件WR的数量固定为32个，接受端WR数量最多可设置为32768（由网卡决定）
- Message越小，传输效率越低，吞吐率越低；为保证吞吐率，接受端需要的WR的数目越多，但是与此同时同时，接受端需要的缓存buffer也越多（N * bufferSize）

Software Trick



- 提交工作请求 (Work Request – WR)
- 生成完成队列项 (Completion Queue Entry – CQE)
- 硬件处理WR
- CQE进入完成队列 (Completion Queue – CQ)
- **查询CQ (Polling CQ)**

- 忙轮询 (Busy–Polling) – 速度最快，但最耗费资源

  应用程序在一个紧凑的循环中，不停地、连续地调用轮询函数（如 ibv_poll_cq）来检查完成队列（CQ）中是否有新的完成队列项（CQE）

- 事件驱动/阻塞式轮询 (Event–Driven / Blocking) – 效率最高，但延迟较高

  应用程序不主动轮询，而是请求硬件在有新的CQE到达时通过一个“事件”来通知它。应用程序的线程会“睡眠”或阻塞，直到事件发生。

- 混合轮询 (Hybrid Polling) – 最佳实践的折中方案

  结合以上两种方式的优点。先进行一小段时间的忙轮询。如果在这段时间内没有等到CQE，就转为事件驱动的阻塞模式，等待下一次通知。

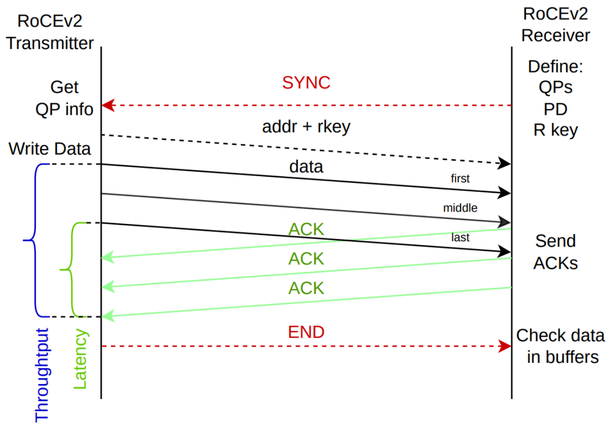
Software Trick

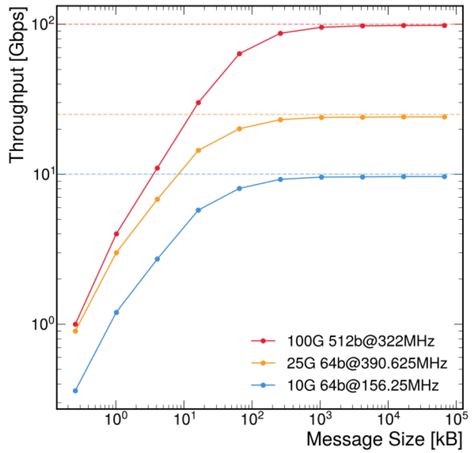
- NUMA（非一致内存访问架构）
  - 多路 CPU 系统（如双路 Xeon）具有多个 NUMA 节点；
  - 每个节点有本地内存和本地 CPU 核；
  - 跨节点访问内存会有更高延迟和更低带宽；
  - RDMA 通常对延迟/带宽非常敏感，因此控制亲和性至关重要。

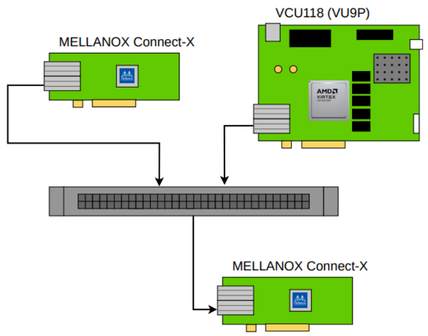
```
[[dongs@hd09 ~]$  cat /sys/class/infiniband/rocep94s0f1/device/numa_node
0
[[dongs@hd09 ~]$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
node 0 size: 94669 MB
node 0 free: 35579 MB
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47
node 1 size: 96715 MB
node 1 free: 44921 MB
node distances:
node     0    1
   0:   10   21
   1:   21   10
```

```
numactl --cpunodebind=0 --membind=0 ./rdma_app --recv_op write --msg_size 2048000 --num_wrs 16384
```

Preliminary Result

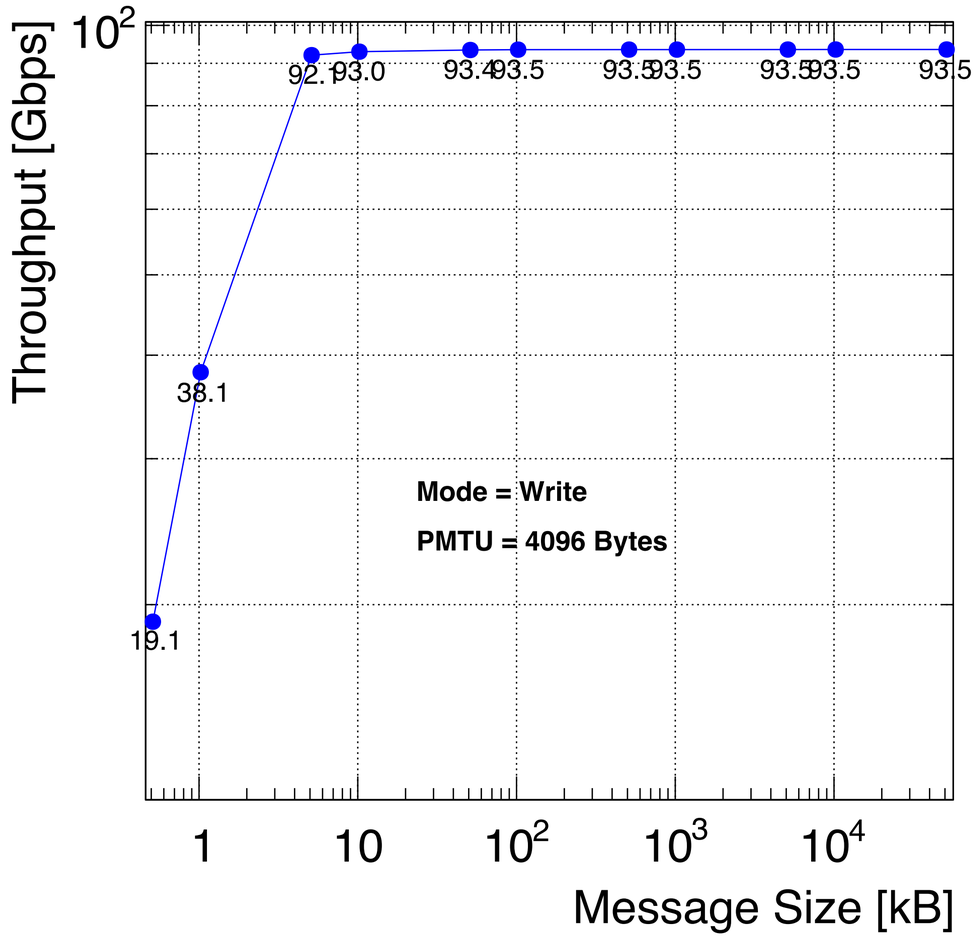






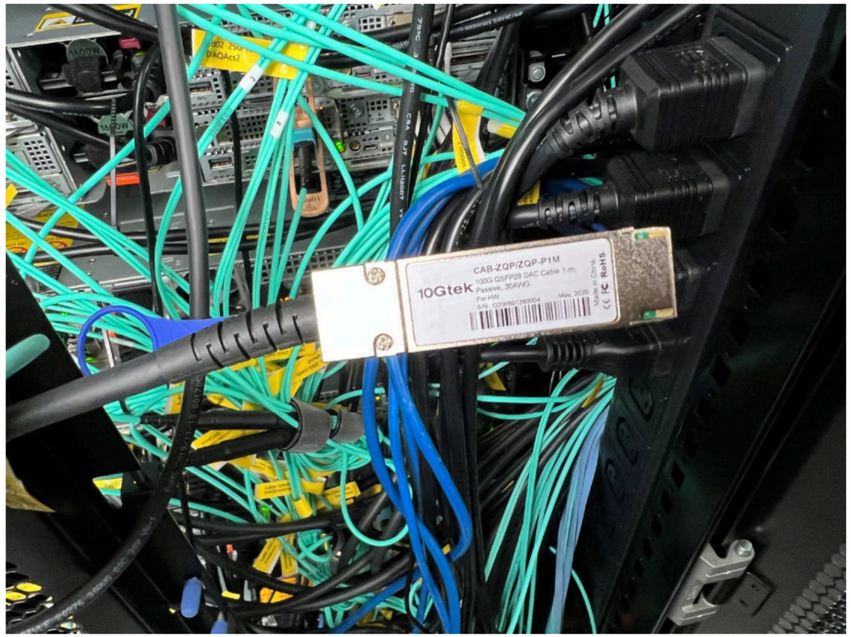
Speed	Msg.size [kB]	Latency [μs]	Tot. Throughput [Gbps]
10G	262	~15	9.63
25G	262	~20	24.09
100G	262	~25	93.1

Small issues related with the priorities configured on the switch. The Connect-X was sending on all 8 priority queues, while the VCU118 only on priority queue 0. This triggers a slight TX asymmetry (~62% Connect-X, ~38% VCU118

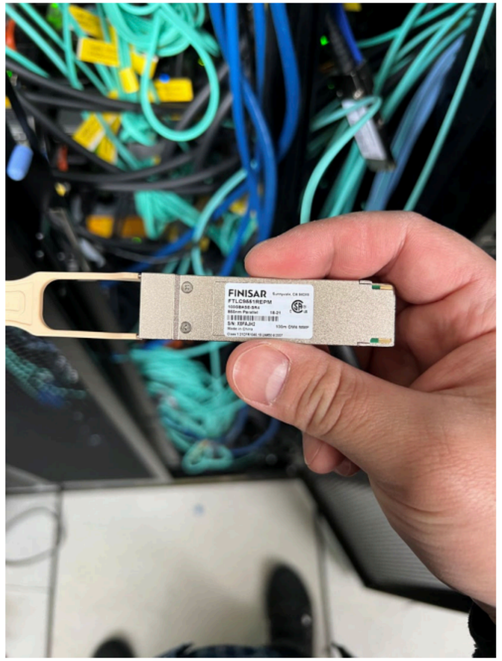


- Message大小5K时，吞吐率达到92.1Gbps，最大吞吐为93.5Gbps

Cables



- 无源铜缆（DAC）测试链接通过



- 光缆（收发器）测试链接不通过

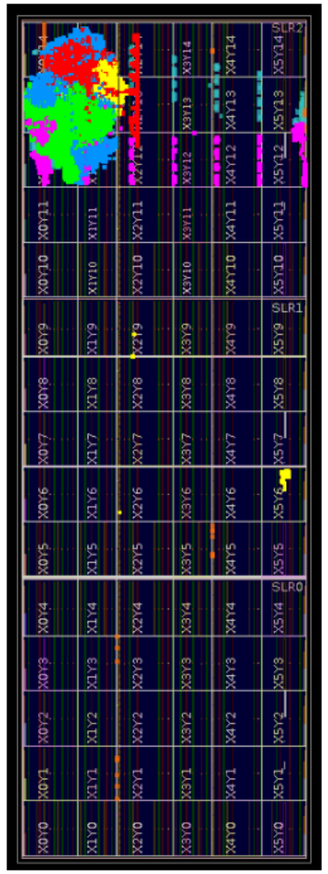
Summary and Outlook

- 基本复现了CMS一个开源项目的工作，解决了以下问题：
  - ✓连续传输逻辑
  - ✓Write出错
  - ✓接收端软件编写
  - ✓吞吐率
- 后续计划：
  - 基于DDR的重传功能实现
  - 多QP的实现
  - 优先级流量控制（PFC）
  - 后端计算/存储压力?

BACKUP

Resources

100G Floorplan

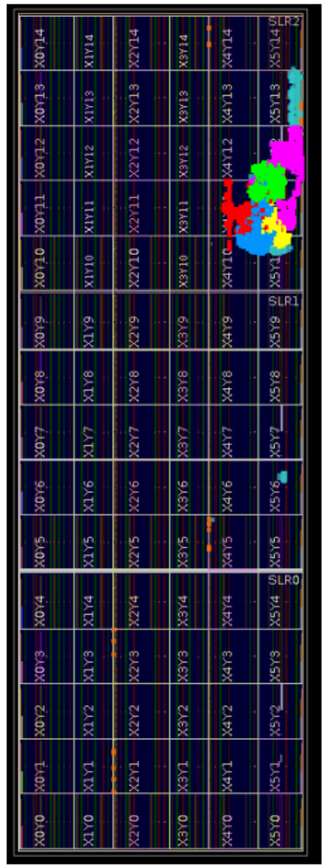


	Speed	LUT [k]	Regs [k]	BRAM	URAM
With Retrans	10G	16	19	13.5	8
	25G	16	19	11.5	18
	100G	37	52	36.5	64
Without Retrans	10G	13	16	11.5	0
	25G	13	16	11.5	0
	100G	30	43	24.5	0

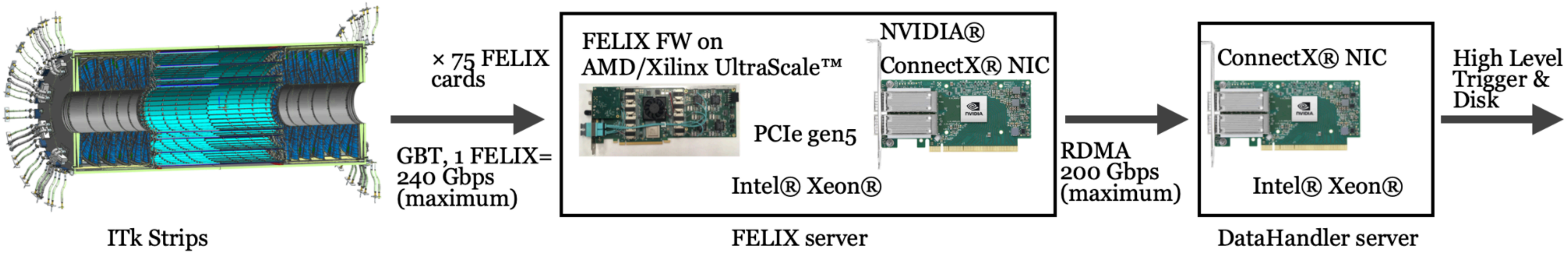
Example floorplan of the 100G implementation.
In violet the MAC and GTYs, in green the UDP/IP components, in blue the RoCEv2 TX and in red the retransmission module are shown.
Some ILAs are used during the debugging steps and monitoring.

Retransmission memory set to to buffer around 160 μs worth of data. So 2MB for 100G, 0.5MB for 25G and 0.26MB fo 10G

10G/25G Floorplan



RDMA in ATLAS



RDMA in HEP

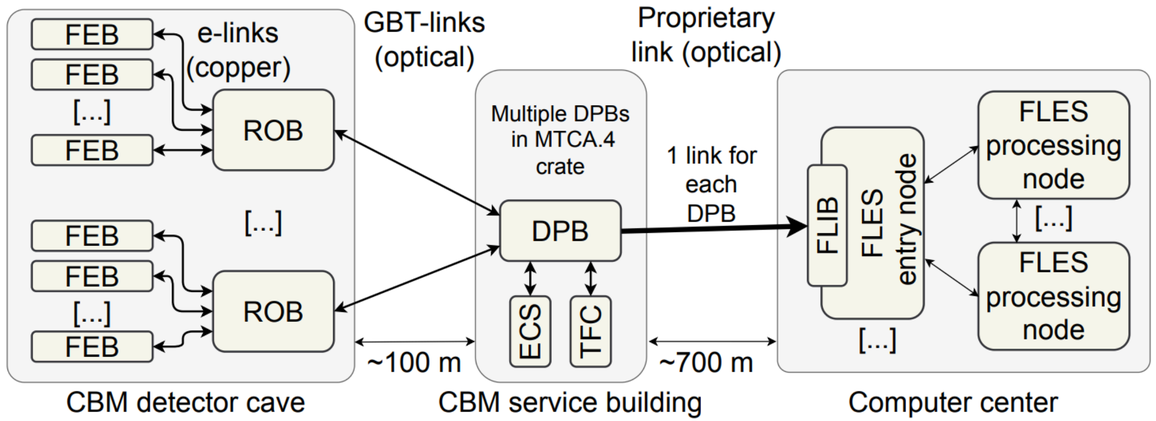


Figure 2. Block diagram of the first DPB-based prototype of the STS readout chain in the CBM experiment.

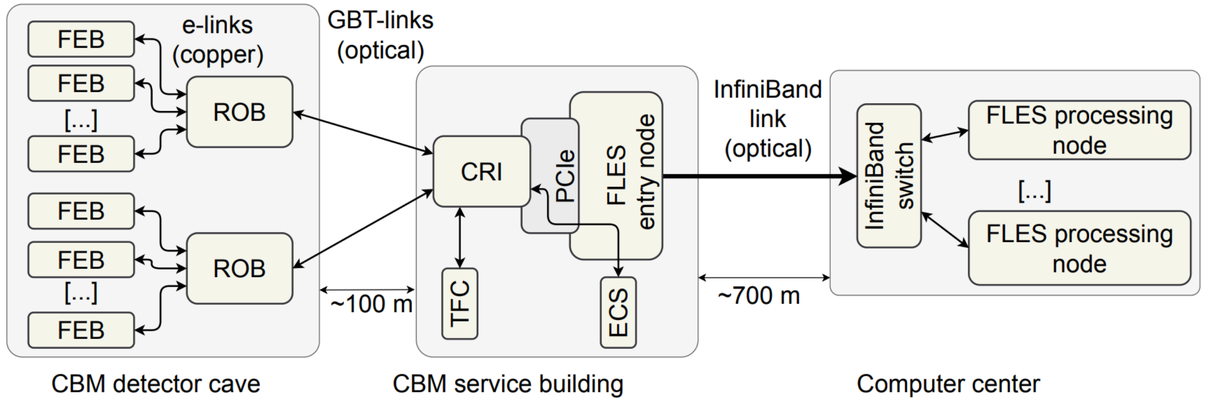


Figure 4. Block diagram of the second prototype of the STS readout chain for the CBM experiment.