Draft version November 21, 2024
Typeset using LaTeX default style in AASTeX631

A 400Gbit Ethernet core enabling High Data Rate Streaming from FPGAs to Servers and GPUs in Radio Astronomy

Wei Liu,[1] Mitchell C. Burnett,[2] Dan Werthimer,[1] and Jonathon Kocz[1]

[1] *Department of Astronomy*
University of California, Berkeley
425 Campbell Hall
Berkeley, CA, 94720-3411, USA
[2] *Department of Electrical And Computer Engineering*
Brigham Young University
Provo, UT, 84602, USA

ABSTRACT

The increased bandwidth coupled with the large numbers of antennas of several new radio telescope arrays has resulted in an exponential increase in the amount of data that needs to be recorded and processed. In many cases, it is necessary to process this data in real time, as the raw data volumes are too high to be recorded and stored. Due to the ability of graphics processing units (GPUs) to process data in parallel, GPUs are increasingly used for data-intensive tasks. In most radio astronomy digital instrumentation (e.g. correlators for spectral imaging, beamforming, pulsar, fast radio burst and SETI searching), the processing power of modern GPUs is limited by the input/output data rate, not by the GPU's computation ability. Techniques for streaming ultra-high-rate data to GPUs, such as those described in this paper, reduce the number of GPUs and servers needed, and make significant reductions in the cost, power consumption, size, and complexity of GPU based radio astronomy backends. In this research, we developed and tested several different techniques to stream data from network interface cards (NICs) to GPUs. We also developed an open-source UDP/IPv4 400GbE wrapper for the AMD/Xilinx IP demonstrating high-speed data stream transfer from a field programmable gate array (FPGA) to GPU.

Keywords: Astronomical techniques, Astronomical instrumentation, Astronomy data acquisition

1. INTRODUCTION

Modern radio telescopes, such as the Square Kilometre Array (SKA)Schilizzi et al. (2007); Dewdney et al. (2009), the Atacama Large Millimeter/submillimeter Array (ALMA)Brown et al. (2004), Deep Synoptic Array(DSA)Hallinan et al. (2019); Connor et al. (2022) and MeerKATJonas & MeerKAT Team (2016) generate data at unprecedented rates, reaching tens to hundreds of terabytes per second. This data needs to be transported from data acquisition systems to real time processing units. There are different ways to process this data. Traditionally, an application specific integrated circuit based system, such as the Wideband Interferometric Digital ARchitecture(WIDAR)Carlson (2000) correlator, would be used. Alternatives include FPGA and more recently FPGA/GPU based hybrid systems.

In radio astronomy, it is typically desirable to process as much frequency bandwidth as possible. High-speed ADCs are commonly used to process bandwidths of tens to hundreds of GHzJiang et al. (2020). To receive these data streams, FPGAs with high-speed transceivers are utilized to receive, channelize and then subsequently transfer them to data processing unitsAlvear et al. (2016); Liu et al. (2021). If these units are housed in a server (as is the case for CPU or GPU based processing), this is typically done via one or more NICs. Between the NICs and the FPGAs, high-speed Ethernet switch is also a critical part for the data transportation.

GPUs are renowned for their capacity for parallel processing, making them ideal for the computationally demanding jobs needed for astronomical data analysis. For instance, Fast Fourier Transforms (FFTs) are utilized in GPUs to

transform time-domain signals into a frequency-domain representationAdámek et al. (2021). The performance of GPU-based FFTs is typically much better than CPU-based FFTsAyala et al. (2022). Correlators need to receive and process data from several stations. The most common correlator implementation is known as the "FX correlator", which first channelizes the band using a polyphase filter bank, and then cross-multiplies each antennas channelized data with every other antenna's data. The computations can be done in parallel, which is ideal for GPUsClark et al. (2013). In particular, Tensor-cores on the latest GPUs are special-purpose, matrix-matrix multiplication units that operate on limited-precision input data, which are good for correlator implementationsRomein (2021). Various other common elements of a data processing pipeline for radio telescopes (such an interference mitigation, calibration algorithms, and transient detection via beamforming and de-dispersion) can also be performed on GPUs Akeret et al. (2017); Agarwal et al. (2020); Yu et al. (2024).

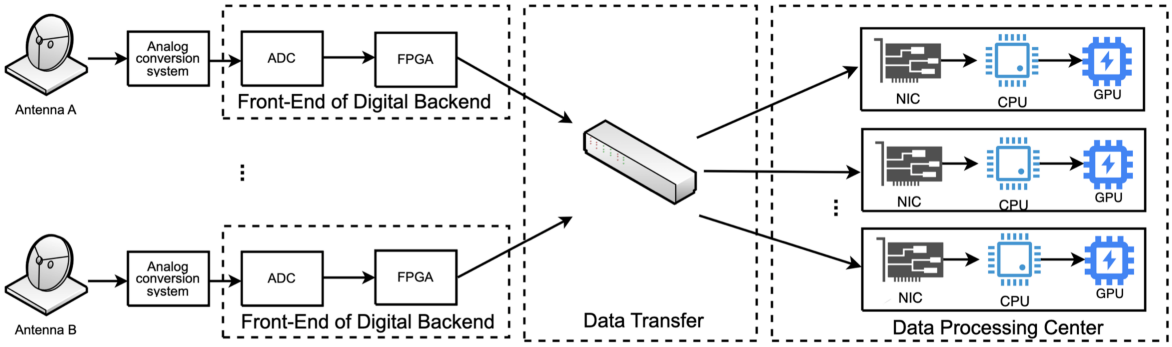


Figure 1. Diagram of a typical radio astronomy system showing the flow of data from telescopes to data processing centers. NICs, FPGAs, and GPUs are required to manage high-speed data streams.

Modern GPUs are powerful enough for most radio astronomy data processing applications. However, moving data between different system components, such as FPGAs to NICs to GPUs is often a limiting step in the data processing pipeline. Utilizing high-throughput interconnects such as Peripheral Component Interconnect Express (PCIe) and NVIDIA's "NVLink"Li et al. (2019) along with remote direct memory access (RDMA) technologyKalia et al. (2016) can mitigate these issues, however, it is still difficult to design, develop and implement a high-performance data transportation system. Additionally, while there are existing solutions(such as GPUDirect) for transferring large amounts of data between two GPU servers, the initial ingest (from e.g. an FPGA based system) is typically done via UDP packets, and is not necessarily able to take advantage of these techniques.

The Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) aims to reduce this hurdle through the development of open source, general purpose hardware, libraries, tools, and reference instrument designsHickish et al. (2016). These tools enable researchers to implement the high-performance instrumentation needed to achieve their science goals in a cost effective and timely manner.

The adoption of 400GbE in radio astronomy offers significant improvements in data transmission and processing capabilities. As radio telescopes collect vast amounts of data, the need for faster, more efficient data handling becomes critical. 400GbE provides the necessary bandwidth to accommodate these large data rates, enabling real-time processing and analysis which are essential for timely discoveries and insights.

This paper outlines our research, beginning with the evaluation of RDMA techniques for bandwidth performance between NICs and GPUs. We then present the development of a 400GbE core with an FPGA for high-speed data transfer and an RDMA-based software framework for server-side data reception. This core will be integrated into the CASPER library, facilitating the rapid development of high-performance instrumentation by other researchers.

In Section 2, the core hardware components used for this research are listed, and estimates the capability of the hardware established; The RDMA techniques for transporting high-speed data streams from NICs to GPUs are demonstrated in Section 3, which includes moving data through the host memory (Section 3.2), and using GPUDirect technology (Section 3.3, 3.4, 3.5); The implemented 400GbE core and associated test results for data transport from FPGA to NIC and then to GPUs are given in Section 4; A summary of all results and outlines for future research directions are provided in Section 5.

2. HARDWARE REQUIREMENTS FOR 400GBE

2.1. *System Description*

To transfer data from an FPGA to a GPU at 400 Gbps, it is important that all components in the data path are capable of operating at such speeds. This includes the FPGA, connecting cables, and all elements of the receiving server system. As RDMA is typically used to transfer high-speed data, RDMA technology should be supported by the NIC as well. A 400GbE NIC with a 16 lane PCIe 5.0 interface is required to support a theoretical maximum of 512 Gb/s between the NIC and other devices (DRAM controller, CPU or GPUs) on the PCIe 5.0 bus. A PCIe 5.0 motherboard and a PCIe 5.0 CPU are also necessary for the high data rate transportation. As DRAM is often used to transfer data to GPUs, having more DRAM controllers in the CPU will enhance performance. The NIC used for the tests described in this paper is the NVIDIA MCX75310AAS-NEAT[1], which supports PCIe 5.0 and 4x100 Gbps/lane; the CPU is Intel Xeon Silver 4410T[2], which contains 80 lanes of PCIe 5.0 and eight DDR5 controllers; the motherboard is a Gigabyte MS03-CE0[3], which has seven PCIe 5.0 slots.

To test packet transfers from the FPGA, we need to be able to implement a 400GbE core on the FPGA board, which requires high-speed transceivers at a minimum. Because the NIC supports 100Gbps/lane, the FPGA transceivers should also support 100Gbps/lane. We choose the VPK180[4] FPGA board based on AMD Versal Premium System on chip (SoC). Inside the Versal SoC, there are tens of GTM transceivers, which supports up to 112 Gbps when working in half-density mode.

The majority of GPUs currently on the market support up to PCIe 4.0, capable of transfers of at least 200 Gbps. Therefore, we used two PCIe 4.0 GPUs for these tests. GPUs can be broadly classified into two categories: gaming GPUs and scientific, or compute, GPUs. In Section 3 we compare the effectiveness of various data-moving techniques when comparing these two types of GPUs. The GPUs we tested are the NVIDIA RTX A6000[5] and RTX 4070[6].

Either copper cables or fiber optic links powered by transceivers are used for Ethernet interconnect. Inexpensive passive copper cables can be used for shorter distances (less than 3 meters for 400GbE), while fiber optic links, which can extend several kilometers, are needed for greater distances. In our work, we used copper cables from Fiberstore, which support the 400GBASE-CR4 interface standard. Table 1 summarizes the hardware used in this work.

2.2. *PCIe Bandwidth*

PCIe is a high-speed bus interface standard for connecting the peripheral hardware components of a computer system. It serves as the critical technology enabling communication between the motherboard and peripherals, such as GPUs and NICs. The achievable bandwidth for the different PCIe versions is shown in Table 2. The above rates are unidirectional. Full-duplex bidirectional rates are twice the table values.

As the PCIe bus is the primary mechanism for data transfer used by majority of devices, including the NICs and GPUs, it is important to measure the PCIe bus bandwidth in order to estimate the maximum performance that is achievable. Table 1 indicates that the system we built for testing has a bottleneck on GPUs, which have a PCIe 4.0 interface, whereas other components support PCIe 5.0. To verify we achieve the PCIe 4.0 bandwidth, we used the

[1] https://docs.nvidia.com/networking/display/connectx7vpi/specifications

[2] https://www.intel.com/content/www/us/en/products/sku/232388/intel-xeon-silver-4410t-processor-26-25m-cache-2-70-ghz/specifications.html

[3] https://www.gigabyte.com/Enterprise/Server-Motherboard/MS03-CE0-rev-1x-3x

[4] https://www.xilinx.com/products/boards-and-kits/vpk180.html

[5] https://www.nvidia.com/en-us/design-visualization/rtx-a6000/

[6] https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4070-family/

Table 1. Core Hardware for 400GbE Evaluation

Hardware	Part Number	Specification
CPU	Intel Xeon Silver 4410T	80 lanes of PCIe 5.0; 8 memory channels of DDR 5
NIC	MCX75310AAS-NEAT	PCIe 5.0; 4 x 100Gbps/lane
Motherboard	MS03-CE0	7 x PCIe 5.0 slots; 8-Channel DDR5 slots
FPGA Board	VPK180	Transceivers support up to 112Gbps
GPUs	RTX A6000/ RTX 4070	PCIe 4.0 scientific/gaming GPUs
Copper cables	OSFP400-PC015	Support 400GBASE-CR4; OSFP-RHS cable for the NIC to NIC test
Copper cables	OSFPFL-400G-QDDPC01	Support 400GBASE-CR4; OSFP-RHS to QSFP-DD cable for the FPGA to NIC test

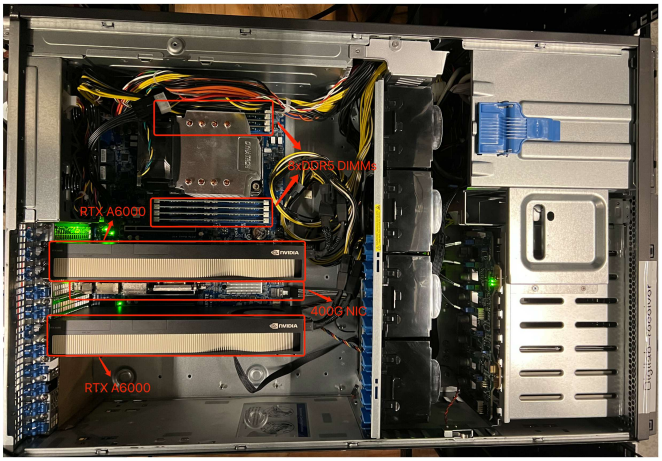


Figure 2. The server we setup for the 400G test, which includes 2 x RTX A6000(RTX 4070) GPUs, a 400G NIC, 8 x DDR5 DIMMs, a PCIe5.0 mother board and a PCIe5.0 CPU.

PCIe Version	**Release Year**	**Transfer Rate (GT/s per lane)**	**Bandwidth (GB/s per lane)**	**Unidirectional Bandwidth (x16 configuration, GB/s)**
1.0	2003	2.5	0.250	4.0
2.0	2007	5.0	0.500	8.0
3.0	2010	8.0	0.985	15.75
4.0	2017	16.0	1.969	31.5
5.0	2019	32.0	3.938	63.0
6.0	2022	64.0	7.877	126.0

Table 2. PCIe Versions and Their Associated Data Transfer Rates

code[7] provided by NVIDIA to measure GPU bandwidth. The bandwidth performance for the RTX A6000 and RTX 4070 is shown in Figure 3(a) and Figure 3(b), respectively. Despite the substantial transfer rate achieved, this falls significantly short of the theoretical maximum transfer of 256 Gbps. The rates achieved do, however, match those

[7] https://github.com/liuweiseu/bandwidthtest/blob/master/bandwidthtest.cu

achieved by NVIDIA with this code [8]. As such, while it may be possible to achieve higher rates with, for example, other GPU models or drivers, we take this to be the limit of our current configuration for data capture and throughput testing purposes.

```
Device: NVIDIA RTX A6000
Transfer size (MB): 512

Pageable transfers
  Host to Device bandwidth (GB/s): 8.726530
  Device to Host bandwidth (GB/s): 19.170784

Pinned transfers
  Host to Device bandwidth (GB/s): 24.979626
  Device to Host bandwidth (GB/s): 26.353208
```

(a) RTX A6000 bandwidth performance

```
Device: NVIDIA GeForce RTX 4070
Transfer size (MB): 512

Pageable transfers
  Host to Device bandwidth (GB/s): 8.172738
  Device to Host bandwidth (GB/s): 15.021167

Pinned transfers
  Host to Device bandwidth (GB/s): 25.108865
  Device to Host bandwidth (GB/s): 26.303710
```

(b) RTX 4070 bandwidth performance

Figure 3. Measured bandwidth performance of the RTX A6000 and RTX 4070 GPUs over a PCIe 4.0 interface. The results indicate a maximum achievable bandwidth of ∼200 Gbps.

2.3. *Memory Bandwidth*

High-speed data transfer also depends on memory bandwidth, which in this case must have a minimum capacity of 400 Gbps in order to accommodate simultaneous 400 Gbps data streams. With eight DDR5 DIMMs operating at 4800 MHz and an eight DDR5 controller built into the CPU, the memory bandwidth is sufficient. The memory bandwidth is measured using the Intel Performance Counter Monitor (Intel PCM) tool[9] and stress-ng[10][11], verifying that the 8-channel DDR5 memory bandwidth met the requirement for 400 Gbps data transfer (Figure 4).

3. TECHNIQUES FOR DATA TRANSPORTATION FROM NIC TO GPU

Transmission Control Protocol/Internet Protocol (TCP/IP) and RDMA are crucial concepts for high-speed data transportation. Both techniques are used for data transfer, but they operate in fundamentally different ways and have distinct performance characteristics. This section discusses the differences between these approaches and showcases the performance of RDMA techniques using the test setup shown in Figure 5.

3.1. *Remote Direct Memory Access*

TCP/IP, the fundamental set of communication protocols for the Internet and most local networks, consists of IP, TCP and UDP (User Datagram Protocol). Because of no buffering requirement, UDP is simpler to implement and allows for faster overall transmission speeds. In most radio astronomy applications, high-speed data transmission is in unidirectional, from FPGAs to a data processing center, through a dedicated network. Consequently, UDP is commonly implemented.

While it is simple to receive UDP streams using a Linux kernel UDP socket, it requires multiple buffer copies and heavy CPU usage, limiting its throughput. RDMA bypasses the host CPU and kernel, providing a high-throughput and low-latency network communication. Two outstanding features of RDMA are implemented in comparison to TCPIP: zero-copy and bypass kernel. The difference between UDP and RDMA is shown in Figure 6. For a 400GbE network, RDMA technology is essential for optimal performance.

[8] https://github.com/NVIDIA/nvbandwidth

[9] https://github.com/intel/pcm

[10] https://github.com/ColinIanKing/stress-ng

[11] For best performance, please hand compile the benchmarks, since out of the box they wouldn't use the most efficient AVX.

```
|---------------------------------------|
|--            Socket  0              --|
|---------------------------------------|
|--     Memory Channel Monitoring     --|
|---------------------------------------|
|-- Mem Ch  0: Reads (MB/s):  8115.05 --|
|--            Writes(MB/s):  8098.09 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  2: Reads (MB/s):  8115.27 --|
|--            Writes(MB/s):  8098.03 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  3: Reads (MB/s):  8115.09 --|
|--            Writes(MB/s):  8098.06 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  5: Reads (MB/s):  8115.20 --|
|--            Writes(MB/s):  8098.10 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  6: Reads (MB/s):  8115.18 --|
|--            Writes(MB/s):  8098.13 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  8: Reads (MB/s):  8115.93 --|
|--            Writes(MB/s):  8098.44 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  9: Reads (MB/s):  8115.16 --|
|--            Writes(MB/s):  8098.07 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch 11: Reads (MB/s):  8116.04 --|
|--            Writes(MB/s):  8098.39 --|
|--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --|
|-- NODE 0 Mem Read (MB/s) : 64922.91 --|
|-- NODE 0 Mem Write(MB/s) : 64785.32 --|
|-- NODE 0 PMM Read (MB/s):     0.00 --|
|-- NODE 0 PMM Write(MB/s):     0.00 --|
|-- NODE 0 Memory (MB/s):   129708.23 --|
|---------------------------------------|
|---------------------------------------||---------------------------------------|
|--           System DRAM Read Throughput(MB/s):      64922.91              --|
|--          System DRAM Write Throughput(MB/s):      64785.32              --|
|--            System PMM Read Throughput(MB/s):          0.00              --|
|--           System PMM Write Throughput(MB/s):          0.00              --|
|--                System Read Throughput(MB/s):      64922.91              --|
|--               System Write Throughput(MB/s):      64785.32              --|
|--              System Memory Throughput(MB/s):     129708.23              --|
|---------------------------------------||---------------------------------------|
```

Figure 4. Memory bandwidth measurement of an 8-channel DDR5 memory configuration using the Intel Performance Counter Monitor (PCM) tool and stress-ng. The total memory bandwidth achieved is approximately ∼1013 Gbps, meeting the requirements for 400 Gbps data transfer.

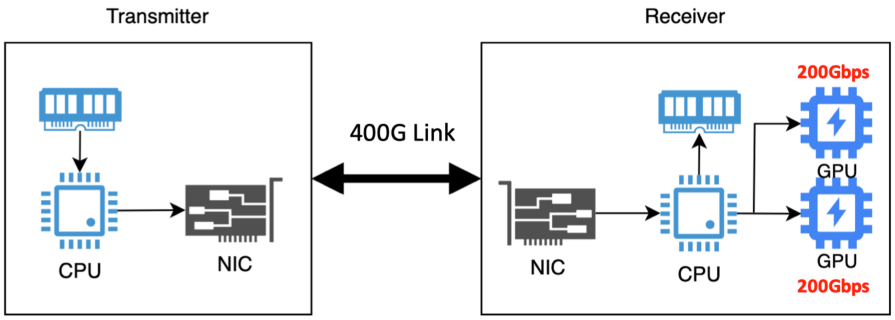


Figure 5. Setup of the 400G test servers used in the experiment. The servers are equipped with 400G NICs and GPUs to evaluate data transfer performance using various techniques.

Ibverbs (InfiniBand Verbs) is a low-level programming interface specifically made for InfiniBand/RDMA over Converged Ethernet(RoCE) networks that uses RDMA technology for high-performance network communication. Developers are able to design applications that demand exceptionally low latency and high throughput because to its direct access to the RDMA hardware. The OpenFabrics Enterprise Distribution (OFED) includes ibverbs as part of a broader suite of RDMA-supporting technologies.

For the 400GbE core test, we select RDMA with ibverbs based on performance, compatibility, complexity, and system upgradeability. The 400GbE core generates UDP packets. On the receiver side, the Queue Pair type created

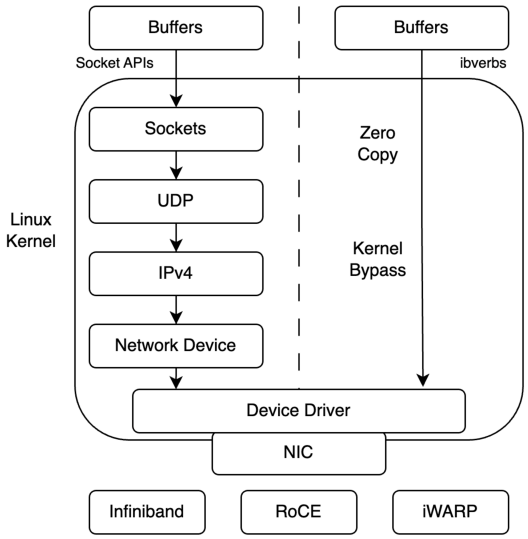


Figure 6. Comparison of data transfer processes using UDP and RDMA. The figure highlights the benefits of RDMA, including zero-copy and kernel bypass, which result in higher throughput and lower latency.

by ibverbs is RawEth, which captures UDP packets via kernel bypass and zero-copy.

3.2. *NIC to DRAM to GPU*

Data transfers from NIC to GPU via DRAM is a general method compatible with both gaming and scientific GPUs. For 400GbE applications, a full-duplex memory bandwidth of at least 800 Gbps is required. Our servers meet this memory bandwidth requirement, as shown in Section 2.3. Figure 7 illustrates the data transfer path from NIC to GPU through DRAM.

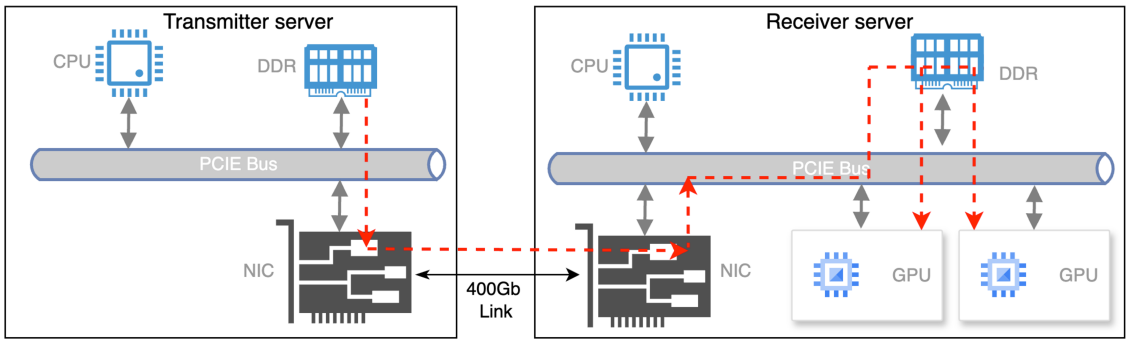


Figure 7. Data transfer path from NIC to GPUs via DRAM. This method utilizes the available memory bandwidth to handle high-speed data streams. Red dashed lines show the direction of data transfer.

In order to estimate the efficiency of data transfer from a NIC to GPUs via DRAM, a packet generator is developed on the transmitting server that sends packets using a 400GbE link. The maximum data rate of the packet generator is ~380 Gbps. As this exceeds the capture code capability, the speed of the generator is artificially limited to ensure we can the maximum data rate without packet loss. Each packet payload contains a packet sequence counter that was incremented by one. By examining the packet count sequence on the receiving end, we determine if there is any packet loss.

The open Source HASHPIPE framework[12][13]MacMahon et al. (2018) is used on the receiving server to capture packets and verify packet sequence values. Two threads, the network thread and gpu thread, are created in HASHPIPE for receiving packets and moving packets to the GPUs. A shared memory buffer of status values is provided by HASHPIPE for real-time status monitoring. The HASHPIPE framework used in the test is shown in Figure 8.

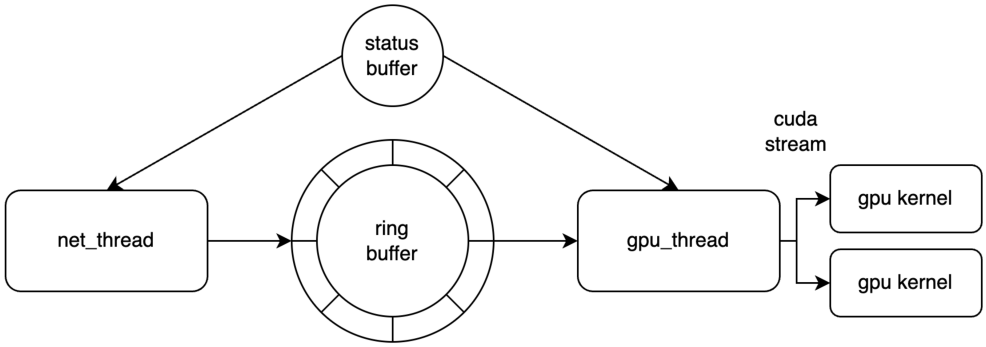
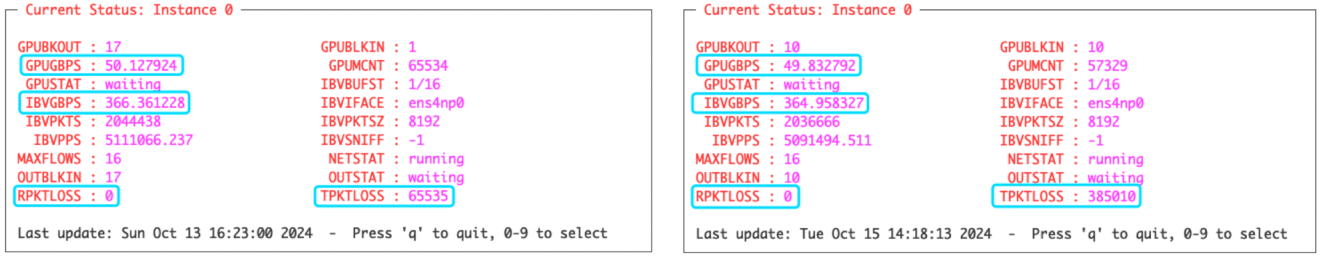


Figure 8. Architecture of the HASHPIPE framework used in the data transfer tests. The framework provides separate network and GPU threads for efficient packet handling, as well as provides real-time thread status monitoring.

The ibverbs APIs are used in the network thread and allow for receiving packets at a high data rate. Once the network thread receive packets from the 400G NIC, they transport the data to shared ring buffers in the host RAM. Data from the ring buffer were transferred to two GPUs using two CUDA streams in the gpu thread. To determine the data transfer rate, we measure the amount of time it takes to fill the ring buffer. Additionally, we verify the counter sequence values in received packets checking for packet loss. The calculated data rate and packet loss are stored in real time using the status buffer, shown in Figure 9(a)(RTX A6000) and Figure 9(b)(RTX 4070).



(a) RTX A6000 bandwidth performance

(b) RTX 4070 bandwidth performance

Figure 9. Measured bandwidth performance of the RTX A6000 and RTX 4070 GPUs during the NIC to GPUs via DRAM test. Results show that the data transfer rate without packet loss is about 360 Gbps. RPKTLOSS shows the real-time packet loss, which is always zero after the initialization; TPKTLOSS shows the total number of packet loss, which is non-zero during the initialization, but remains constant after initialization.

3.3. *Direct Data Transport from NIC to a Single GPU*

GPUs and NICs, both PCIe devices, benefit from PCIe's high bandwidth and low latency when RDMA is used. In order to directly transfer data from NICs to GPUs, specific NICs and GPUs are needed: The NICs and GPUs should both be able to support RDMA. As we use NVIDIA NICs as well NVIDIA GPUs in this test, GPUDirect technology[14] can be used. GPUDirect allows GPUs to communicate directly with other system parts, bypassing the CPU and system memory, reducing latency and increasing throughput. Since both Tesla and Quadro GPUs support GPUDirect,

[12] https://github.com/david-macmahon/hashpipe

[13] https://github.com/liuweiseu/hashpipe-ibverbs-demo.git

[14] https://docs.nvidia.com/cuda/gpudirect-rdma/

we utilized the RTX A6000 for the GPUDirect test. Figure10 shows the direct data transfer path from NIC to GPU using GPUDirect. As the PCIe limited IO bandwidth on the GPU is 200Gbps, we initially tested GPUDirect to one GPU with a 200G link.

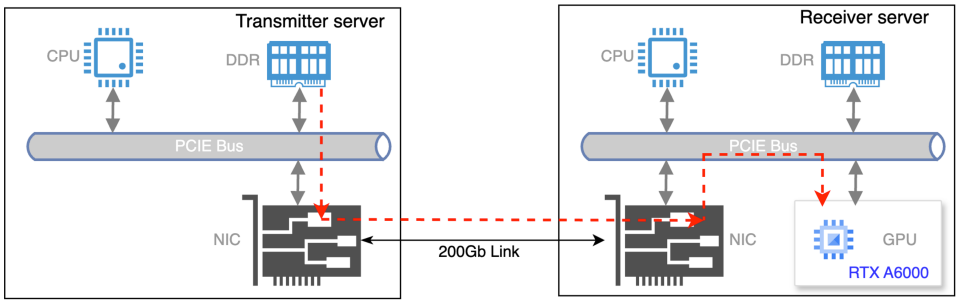


Figure 10. Direct data transfer path from NIC to GPU using GPUDirect technology. RDMA transactions reduces latency by bypassing the CPU and system memory. The link between the two servers is 200G. Red dashed lines shows the direction of data transfer.)

Packets on the transmitter server are generated using the ibverbs APIs and sent out in bursts to gauge bandwidth performance. Each burst transfers a total of 12.5 GB data, with each packet sizes to 8192 bytes and a total of 1638400 packets. The receiving server also uses ibverbs to capture and report packet counts. The ibverbs routine (ibv_poll_cq[15]) reports how many packets are received successfully, so the packet loss can be checked by comparing the number of sent packets and the number of received packets. The RTX A6000 GPU supports PCIe 4.0, limiting throughput to 200Gbps. We achieved a maximum data rate of 178Gbps[16] without packet loss, as shown in Figure11.

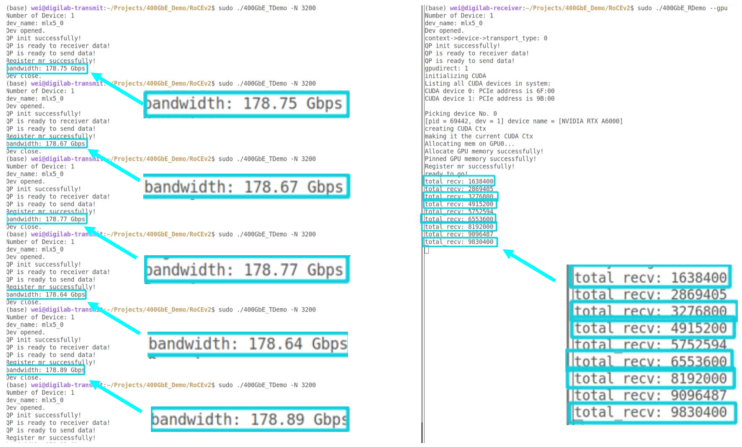


Figure 11. Measured bandwidth results of data transfers using GPUDirect to a single GPU on a 200 Gbps link. The maximum data rate achieved without packet loss is 178 Gbps. The packet loss is checked By comparing the number of sent packets and received packets. In each test, 1638400 packets are sent and 1638400 packets are received.

3.4. *Direct data transport from NIC to two GPUs*

As the 400G NIC is capable of supporting two 200Gbps links, and each GPU supports 200Gbps IO bandwidth, the capability of streaming to two GPUs simultaneously using GPUDirect was also tested. Figure 12 shows the direct data transfer path from NIC to two GPUs using GPUDirect.

[15] https://www.rdmamojo.com/2013/02/15/ibv_poll_cq/

[16] https://github.com/liuweiseu/400GbE_Demo.git

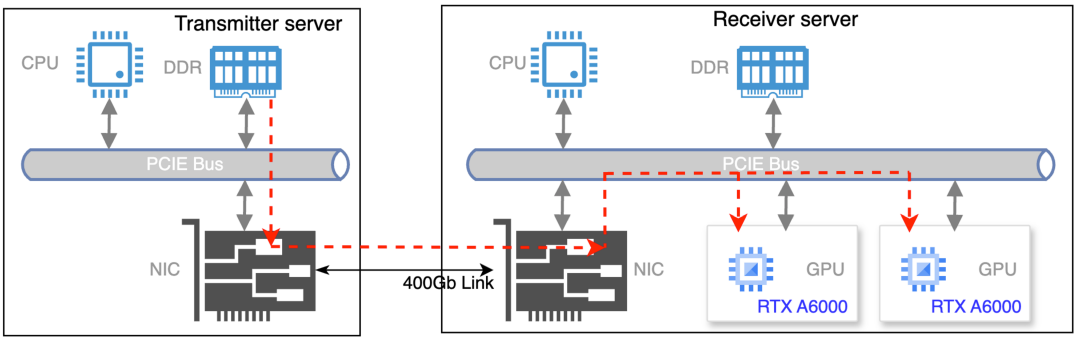


Figure 12. Direct data transfer path from NIC to two GPUs using GPUDirect technology. The link between the two servers is 400G link. Red dashed lines show the direction of data transfer.

On the transmitter server side, the same code based on ibverbs API mentioned in section 3.3 was used. In this case, however, the packets contain different source and destination port numbers. On the receiver server side, two receiving queue pairs with different flow are set for steering the two packet streams to the two GPUs. The resulting throughput (∼180 Gbps)is shown in Figure 13.

GPUDirect to GPU0

```
Start to recv...
The number of ib devices is 1.
Open IB device successfully.
Create IB resources successfully.
Init IB resources successfully.
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
total_recv: 1111412    Bandwidth: 72.837 Gbps
total_recv: 2508005    Bandwidth: 91.527 Gbps
total_recv: 3904828    Bandwidth: 91.542 Gbps
total_recv: 5300936    Bandwidth: 91.495 Gbps
total_recv: 6698201    Bandwidth: 91.571 Gbps
total_recv: 8094774    Bandwidth: 91.526 Gbps
total_recv: 9490545    Bandwidth: 91.473 Gbps
total_recv: 10886989   Bandwidth: 91.517 Gbps
total_recv: 12283616   Bandwidth: 91.529 Gbps
total_recv: 13679394   Bandwidth: 91.474 Gbps
total_recv: 15074994   Bandwidth: 91.462 Gbps
```

GPUDirect to GPU1

```
Start to recv...
The number of ib devices is 1.
Open IB device successfully.
Create IB resources successfully.
Init IB resources successfully.
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
total_recv: 780366     Bandwidth: 51.142 Gbps
total_recv: 2176995    Bandwidth: 91.529 Gbps
total_recv: 3573712    Bandwidth: 91.535 Gbps
total_recv: 4969825    Bandwidth: 91.496 Gbps
total_recv: 6366914    Bandwidth: 91.560 Gbps
total_recv: 7763874    Bandwidth: 91.551 Gbps
total_recv: 9159603    Bandwidth: 91.470 Gbps
total_recv: 10555971   Bandwidth: 91.512 Gbps
total_recv: 11952608   Bandwidth: 91.530 Gbps
total_recv: 13348517   Bandwidth: 91.482 Gbps
total_recv: 14744203   Bandwidth: 91.468 Gbps
```

Figure 13. Measured bandwidth results of data transfers to two GPUs using GPUDirect on a 400Gbps link. The maximum data rate achieved is ∼90Gbps × 2.

3.5. *Direct Data Transport from two NICs to two GPUs*

While GPUDirect is based entirely on the PCIe bus similarly to RDMA, the performance was more limited, particularly when working with multiple GPUs. Besides measuring the performance of GPUDirect from one NIC to two GPUs, the performance of two pairs of NICs to GPUs with GPUDirect was also tested. The data transfer path is shown in Figure 14, and the result (~276 Gbps) is shown in Figure 15. The improvement in this rate compared to the single 400Gbps NIC suggests that the limitations in data rate transfer of the single NIC to two GPUs are due to the GPUDirect configuration and PCIe lanes available in the case of a single NIC.

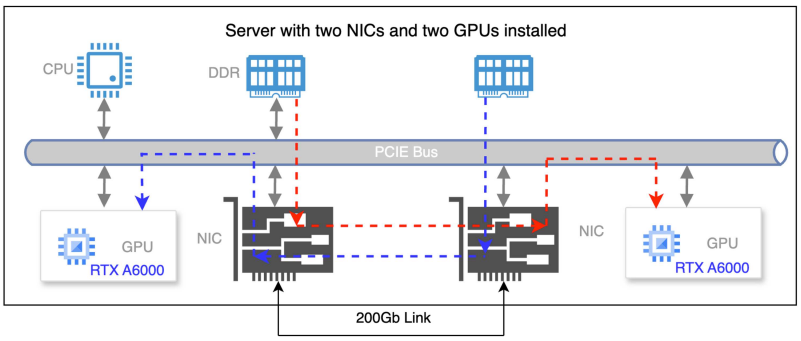


Figure 14. Direct data transfer path from two NIC to two GPUs on one server. Both of the two NICs send data to each other, and use GPUDirect to move the received data to two GPUs. The link is 200G. Red and blue dashed lines shows the direction of data transfer.

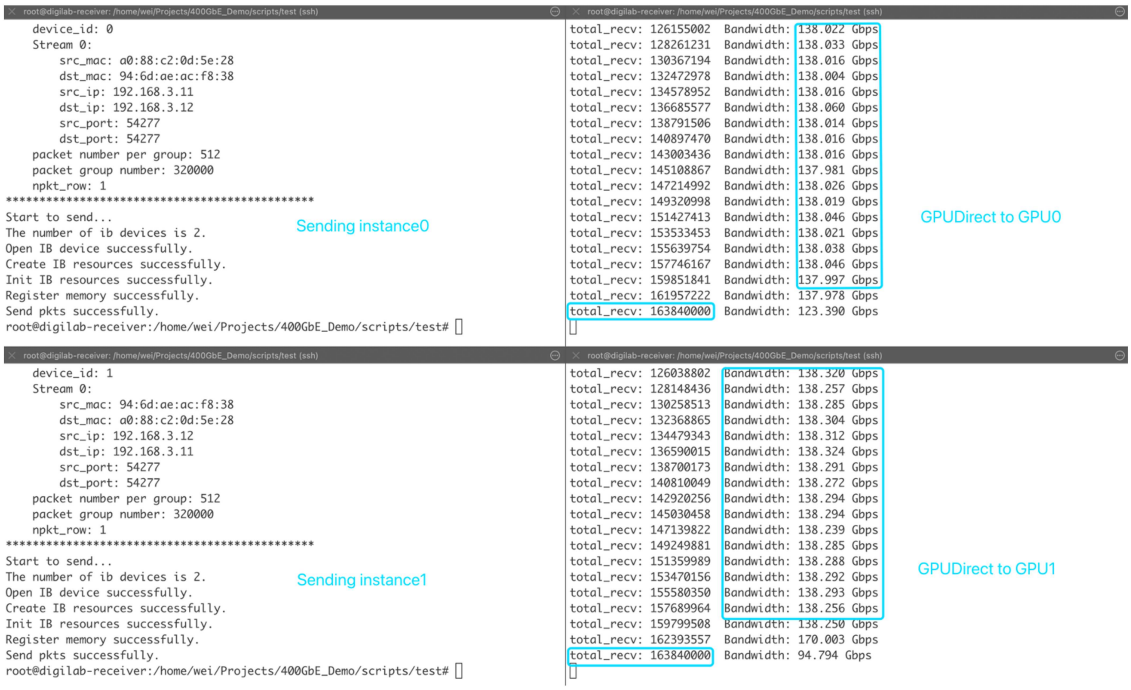


Figure 15. Measured bandwidth results of data transfers from two NICs to two GPUs using GPUDirect on a 200Gbps(× 2) link. The maximum data rate achieved is ~138Gbps × 2.

4. 400GBE FPGA CORE

The open source 400GbE FPGA core comprises two primary modules: the UDP packet generation core and the MAC/PHY core. The UDP packet generation core creates the UDP data packet and inserts the user-defined MAC address, IP address, port numbers into the packets. The MAC/PHY core, compliant with the IEEE 802.3ck-2022 400Gbps Ethernet standard, interfaces the FPGA with other devices over copper or optical transceiver modules. The AXI4 bus is used to transport control signals and statistical data. Figure 16 illustrates the core modules.

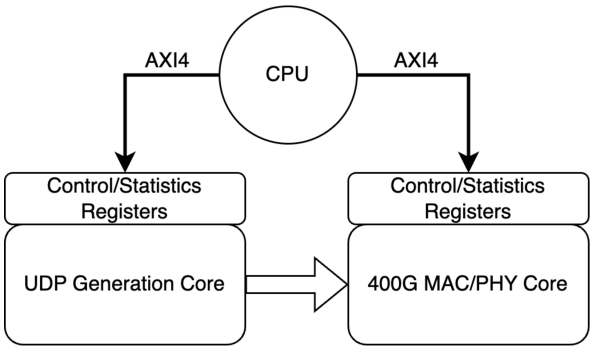


Figure 16. Block diagram of the 400GbE FPGA core shows the UDP packet generation and MAC/PHY core modules.

4.1. *400GbE Core Implementation*

4.1.1. *UDP packet generation core*

The 400GbE UDP generation core, modified from the CASPER 100G core[17], consists of two submodules: a streaming data module, and a CPU data module.

The streaming data module handles high-speed data transmission and reception to and from the 400G MAC/PHY core. It obtains MAC, IP, and port information via the AXI4 bus to produce packets at a high data rate, stored in registers or the ARP cache. For the TX data path, the module retrieves data from registers or an internal ARP cache, creates TX packets, and sends them to a TX ring buffer. For the RX data path, received packets are stored in an RX ring buffer and filtered by an RX filter to remove any unexpected packets with incorrect MAC, IP, or port numbers. The block diagram of the streaming data module is shown in Figure 17.

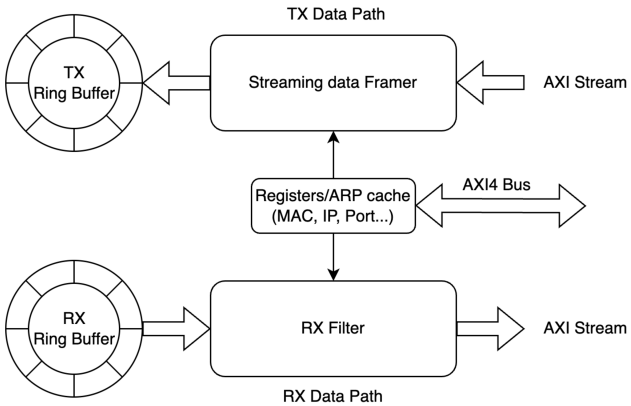


Figure 17. Block diagram of the streaming data module within the 400GbE FPGA core. The module handles high-speed data transmission and reception, including packet generation and filtering.

[17] https://github.com/casper-astro/kutleng_skarab2_bsp_firmware

The CPU data module sends and receives data to and from a microcontroller implemented in the FPGA. It shares registers, ARP cache, and the MAC/PHY core with the streaming module. An arbitration module determines which data to forward to the MAC/PHY core.

4.1.2. *MAC/PHY Core*

In the 400GbE FPGA core, the MAC/PHY core includes the Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA), Physical Medium Dependent (PMD) layer, and support for varying numbers of lanes for various 400GbE protocols.

We employ AMD's (Xilinx) GTM transceivers and 600G channelized multi-rate ethernet subsystem (DCMAC) core. The DCMAC is a high-performance, flexible, Ethernet-integrated hard IP that targets various networking applications. The 400GbE, 200GbE, and 100GbE combinations that the core supports allow for a maximum data rate of 600Gbps. For 400GbE, 200GbE, and 100GbE, it implements all of the 400G PCS operations, including: encoder, scrambler, alignment marker insertion, and forward error correction (FEC). Additionally, it partially performs the PMA function, freeing up 8 data lanes (two PCS lanes are consolidated into one) for GTM transceivers to connect to. For our open source 400GbE core design, the DCMAC core is a good choice because the hard IP on Versal SoC is supported by AMD with a free license.

With capabilities for up to 56Gbps per lane, the Versal SoC's GTM transceiver is the highest performing AMD transceiver. Working in full-density mode or half-density mode, it carries out the remaining PMA functions. For 400GbE applications, the interface is 400GAUI-8 in full-density mode since each GTM transceiver operates independently; in half-density mode, two GTM transceivers cooperate to support 112Gbps per lane. In this scenario, the interface is 400GAUI-4. Since the NIC in our application supports 400GBASE-CR4, in order to obtain 106 Gbps/lane $\times$ 4 lanes, we must configure the GTM in half density mode. We use two GTM quads to achieve the 400Gbps data throughput, because each GTM quad contains four GTM transceivers. The MAC/PHY core, based on DCMAC and GTM transceiver, is shown in Figure 18. The complete FPGA implementation of the 400GbE architecture is shown in Figure 19.

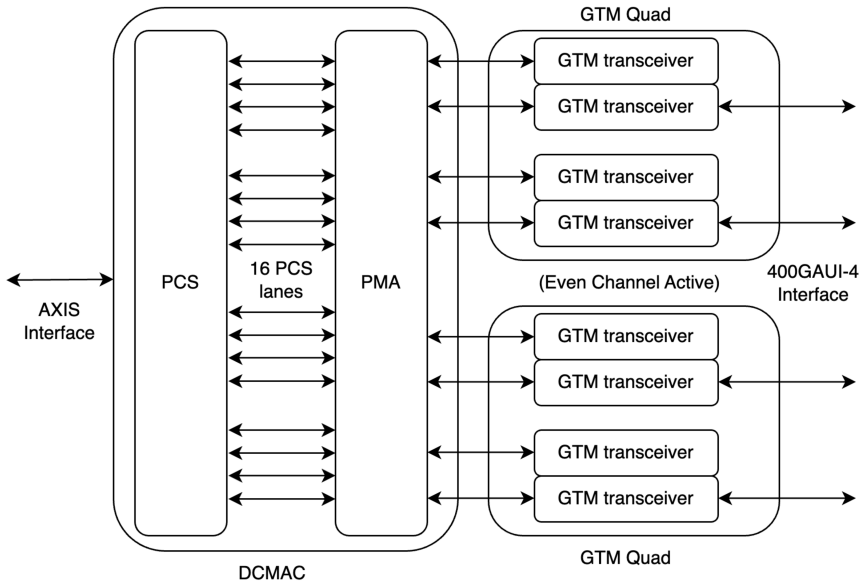


Figure 18. Integrated DCMAC core and GTM transceivers in the 400GbE MAC/PHY core. The components work together to support high-speed Ethernet communication.

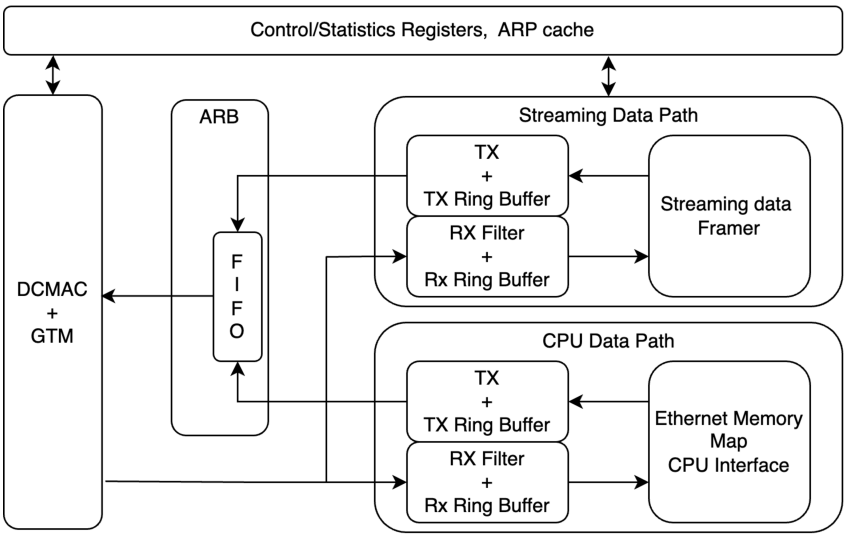


Figure 19. Complete architecture of the 400GbE FPGA core, illustrating the integration of various modules to achieve 400Gbps data transfer rates.

4.2. *FPGA to NIC to GPU through DRAM result*

To evaluate bandwidth performance of the 400GbE FPGA core, we developed a packet generation module that is connected to the 400GbE core. Each packet contains a 16-bit counter for packet loss detection, and the packet rate can be adjusted to obtain the data rate without packet loss. Using the same HASHPIPE framework-based code as described in Section 3.2, we monitor the packet loss status and real-time bandwidth measurement. Figure 20 illustrates the test setup, and Figure 21 shows the actual configuration.

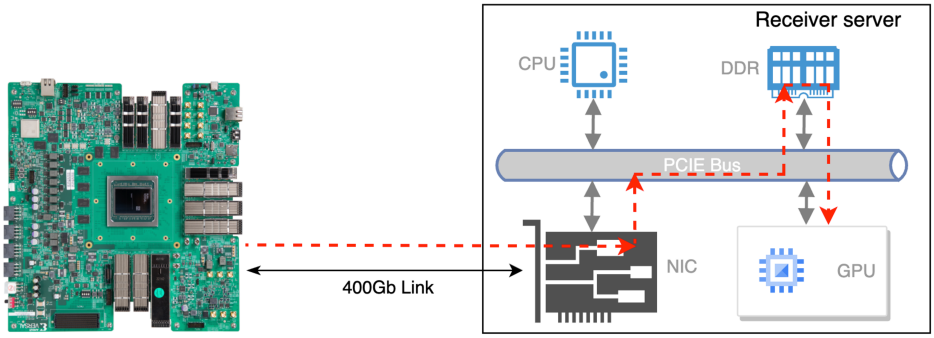


Figure 20. Diagram showing the setup for the FPGA to NIC to GPU via DRAM data transfer test. The setup includes the FPGA board, 400G NIC, and GPUs to evaluate data transfer performance.

Before further testing, we confirmed the 400G link between the FPGA and the server with a 400G NIC. Once configured with the 400GbE firmware, the server indicated an active state at 400G with four active lanes. Figure 22 shows the receiver NIC state.

The GPUs used in this test are two RTX A6000 and two RTX 4070. The clock rate on the FPGA side for generating packets and clocking the 400G Ethernet core is 390.625MHz, and the bus width of the DCMAC core is 1024 bits, so the total data rate from the DCMAC core is up to 400Gbps. After adjusting the speed of generating packets, the code running on the server side can capture the packets at up to ∼362 Gbps without packet loss. We observed no packet loss with data rates up to 362 Gbps. This result is identical as we obtained in Section 3.2. Figure 23 shows the results of the FPGA to NIC to two GPUs test.

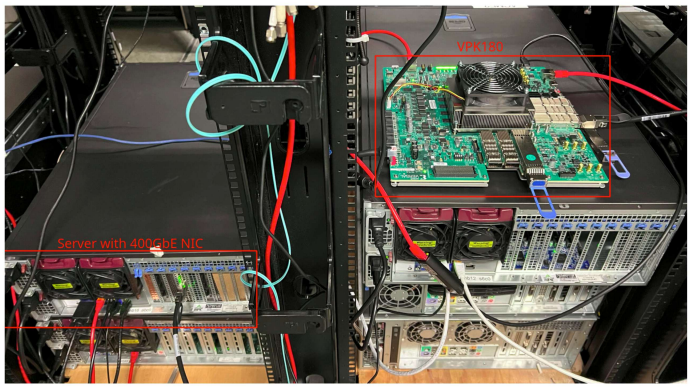
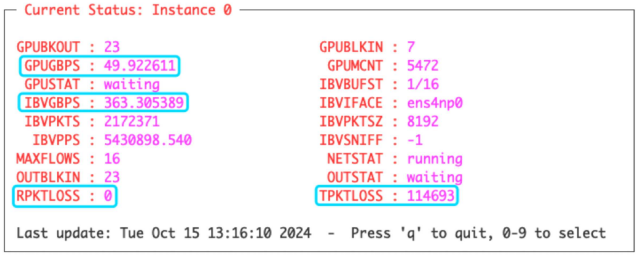


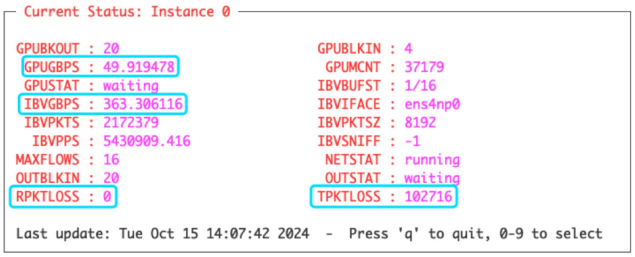
Figure 21. The actual setup used for the FPGA to NIC to GPU test. The setup demonstrates the hardware configuration and connections used in the experiment.

```
~$ sudo mlxlink -d mlx5_0 -e -c

Operational Info
----------------
State                              : Active
Physical state                     : ETH_AN_FSM_ENABLE
Speed                              : 400G
Width                              : 4x
FEC                                : Standard_RS-FEC - (544,514)
Loopback Mode                      : No Loopback
Auto Negotiation                   : FORCE - 400G_4X

Supported Info
--------------
Enabled Link Speed (Ext.)          : 0x00010000 (400G_4X)
Supported Cable Speed (Ext.)       : 0x00013ffe (400G_4X,200G_2X,200G_4X,100G_1
X,100G_2X,100G_4X,50G_1X,50G_2X,40G,25G,10G,5G,2.5G,1G)

Troubleshooting Info
--------------------
Status Opcode                      : 0
Group Opcode                       : N/A
Recommendation                     : No issue was observed

Tool Information
----------------
Firmware Version                   : 28.40.1000
amBER Version                      : 2.8
MFT Version                        : mft 4.27.0-83
```

Figure 22. Status indication of the FPGA linked to the 400G NIC at a 400 Gbps data rate. The figure confirms the active state and link speed of the connection.



(a) RTX A6000 FPGA to GPU bandwidth performance



(b) RTX 4070 FPGA to GPU bandwidth performance

Figure 23. Test results of the FPGA to NIC to two GPUs data transfer, showing a data rate of 362Gbps without packet loss. RPKTLOSS shows the real-time packet loss, which is always zero after the initialization; TPKTLOSS shows the total number of packet loss, which is non-zero during the initialization, but remains constant after initialization. The figure illustrates the effectiveness of the 400GbE FPGA core in high-speed data transfer.

4.3. *FPGA to GPU Test Result with GPUDirect*

As the GPUs we are using in the tests have PCIe 4.0 interface with the speed limitation to ~200Gbps, what we expected is moving data into one GPU at ~200Gbps. To send packets from FPGA to two GPUs with GPUDirect technology, packets are sent out from FPGA with different source and dest port numbers, and these two kinds of packets are sent out one by one. On the server side, two queue pairs with differet steer flows are created for directing the two kinds of packets into two GPUs. Figure 24 shows the test about FPGA to two GPUs with GPUDirect.

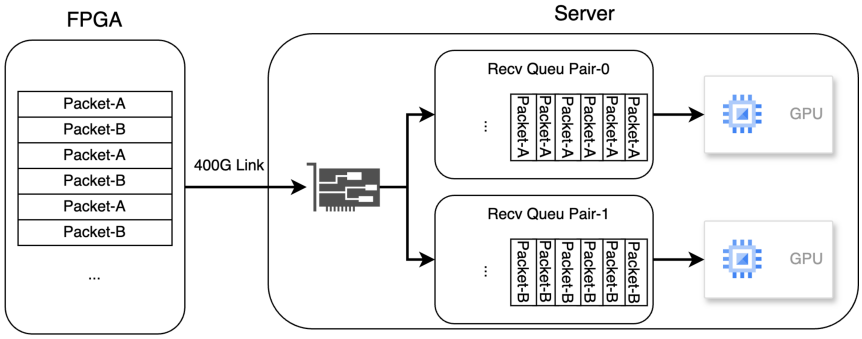


Figure 24. FPGA and server are connected with 400G link. FPGA sends out two kinds of packets with different source and dest port numbers: Packet-A and Packet-B. The data stream are steered to two queue pairs with different steer flow, and then sent to two GPUs directly.

If only one receiving queue pair is enabled, half of the data will be sent to one GPU. The bandwidth is about ~180Gbps, which is identical to the result mentioned in section3.3; If both of the two receiving queue pairs are enable, the one data stream will be split to two data streams, and the data will be sent to two GPUs at the same time directly. The bandwidth of each data stream is ~90Gbps, and the total bandwidth is ~180Gbps. The results about this test are shown in Figure 25(a) and figure25(b).

```
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
total_recv: 530154     Bandwidth: 34.744 Gbps
total_recv: 3057615    Bandwidth: 165.639 Gbps
total_recv: 5584760    Bandwidth: 165.618 Gbps
total_recv: 8111901    Bandwidth: 165.619 Gbps
total_recv: 10871648   Bandwidth: 180.862 Gbps
total_recv: 13647187   Bandwidth: 181.897 Gbps
total_recv: 16422726   Bandwidth: 181.897 Gbps
total_recv: 19198224   Bandwidth: 181.895 Gbps
total_recv: 21973792   Bandwidth: 181.899 Gbps
total_recv: 24749349   Bandwidth: 181.898 Gbps
total_recv: 27524851   Bandwidth: 181.895 Gbps
total_recv: 30300400   Bandwidth: 181.898 Gbps
```

```
root@digilab-receiver: /home/wei/Projects/400GbE_Demo/scripts/Recv (ssh)
    dst_ip: 192.168.3.12
    src_port: 54278
    dst_port: 54278
    use_gpu: 1, gpu_id: 1
    nsge: 1
    disable_recv: 1
**********************************************
Start to recv...
The number of ib devices is 3.
Open IB device successfully.
Create IB resources successfully.
Init IB resources successfully.
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
```

(a) GPUDirect from FPGA to GPU with one queue pair enabled

```
The number of ib devices is 3.
Open IB device successfully.
Create IB resources successfully.
Init IB resources successfully.
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
total_recv: 814971     Bandwidth: 53.410 Gbps
total_recv: 2211083    Bandwidth: 91.496 Gbps
total_recv: 3607634    Bandwidth: 91.524 Gbps
total_recv: 5004039    Bandwidth: 91.515 Gbps
total_recv: 6400649    Bandwidth: 91.528 Gbps
total_recv: 7796209    Bandwidth: 91.459 Gbps
total_recv: 9192310    Bandwidth: 91.495 Gbps
total_recv: 10588388   Bandwidth: 91.493 Gbps
```

```
root@digilab-receiver: /home/wei/Projects/400GbE_Demo/scripts/Recv (ssh)
Open IB device successfully.
Create IB resources successfully.
Init IB resources successfully.
Allocate memory on GPU.
Allocate GPU memory successfully!
Pinned GPU memory successfully!
Register memory successfully.
Create flow successfully.
total_recv: 131553     Bandwidth:  8.621 Gbps
total_recv: 1527580    Bandwidth: 91.490 Gbps
total_recv: 2924165    Bandwidth: 91.527 Gbps
total_recv: 4320544    Bandwidth: 91.513 Gbps
total_recv: 5716966    Bandwidth: 91.516 Gbps
total_recv: 7113164    Bandwidth: 91.501 Gbps
total_recv: 8508744    Bandwidth: 91.461 Gbps
total_recv: 9905153    Bandwidth: 91.515 Gbps
total_recv: 11300900   Bandwidth: 91.472 Gbps
```

(b) GPUDirect from FPGA to GPU with two queue pairs enabled

Figure 25. GPUDirect from FPGA to NIC results. If one queue pair is enabled on the receiver side, the bandwidth is about ~180Gbps; if both of the two queue pairs are enabled, bandwidth for each data stream is ~90Gbps, and total is about ~180Gbps.

5. SUMMARY AND FUTURE WORK

This study examines the bandwidth performance of two RDMA techniques: moving data through DRAM and GPUDirect, when transferring data from NIC to GPUs. Moving data to GPUs through DRAM can achieve approximately 180 Gbps to a single GPU without packet loss and about 360 Gbps to two GPUs. With GPUDirect, we can transfer ∼180Gbps to a single GPU, and ∼90Gbps × 2 to two GPUs simultaneously. Using two NICs and two GPUs with GPUDirect can achieve the bandwidth of ∼138Gbps × 2. When using GPUDirect to two GPUs from a single NIC, the data rate was half of that expected. In the case of the dual NICs to dual GPUs the data rate improved, but was still below that achieved when transferring the data via DRAM. We plan to conduct research on further high-performance computing methods in the future, including Data Plane Development Kit (DPDK) and Holoscan in order to compare the performance of these approaches to our current results.

In addition to the server to server tests, a 400GbE FPGA core was developed. As many radio telescopes using a networked architecture, this allowed the testing of a typical radio astronomy scenario where simple UDP packets were transmitted to a server. Using this core, data rates of 362 Gbps were transferred without experiencing any packet loss between the FPGA and two GPUs. Testing with GPUDirect, the 400GbE FPGA core can transfer data at about 180 Gbps to one GPU. This core will be added to the CASPER library, and the test results demonstrate the potential of RDMA techniques and the FPGA 400GbE core for high data rate applications.

6. ACKNOWLEDGEMENT

The authors would like to thank John Romein (ASTRON) for his advice on 400G NIC settings, Andre Renard (UofT)) for his advice on memory bandwidth improvement, Adam Thompson and Cliff Burdick (NVIDIA) for their assistance with the NIC to NIC tests, and Leo Karnan (AMD) for his advice on GTM transceiver settings. The authors also thank to AMD's University Program for their donation of the VPK180 FGPA board. The research was funded by NSF grants 2009537 and 2307781.

REFERENCES

Adámek, K., Novotnỳ, J., Thiyagalingam, J., & Armour, W. 2021, IEEE Access, 9, 18167

Agarwal, D., Lorimer, D., Surnis, M., et al. 2020, Monthly Notices of the Royal Astronomical Society, 497, 352

Akeret, J., Chang, C., Lucchi, A., & Refregier, A. 2017, Astronomy and computing, 18, 35

Alvear, A., Finger, R., Fuentes, R., et al. 2016, in Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy VIII, Vol. 9914, SPIE, 332–345

Ayala, A., Tomov, S., Luszczek, P., et al. 2022, Univ. Tennessee at Knoxville, Knoxville, TN, USA, Tech. Rep. ICL-UT-22–02

Brown, R. L., Wild, W., & Cunningham, C. 2004, Advances in Space Research, 34, 555

Carlson, B. 2000, Implementation, and Signal Processing, NRC-EVLA Memo, 1

Clark, M. A., Plante, P. L., & Greenhill, L. J. 2013, The International journal of high performance computing applications, 27, 178

Connor, L., Bouman, K. L., Ravi, V., & Hallinan, G. 2022, Monthly Notices of the Royal Astronomical Society, 514, 2614

Dewdney, P. E., Hall, P. J., Schilizzi, R. T., & Lazio, T. J. L. 2009, Proceedings of the IEEE, 97, 1482

Hallinan, G., Ravi, V., Weinreb, S., et al. 2019, arXiv preprint arXiv:1907.07648

Hickish, J., Abdurashidova, Z., Ali, Z., et al. 2016, Journal of Astronomical Instrumentation, 5, 1641001

Jiang, H., Yu, C.-Y., Chen, M., & Liu, M. 2020, Publications of the Astronomical Society of the Pacific, 132, 085001

Jonas, J., & MeerKAT Team. 2016, in MeerKAT Science: On the Pathway to the SKA, 1, doi: 10.22323/1.277.0001

Kalia, A., Kaminsky, M., & Andersen, D. G. 2016, in 2016 USENIX Annual Technical Conference (USENIX ATC 16), 437–450

Li, A., Song, S. L., Chen, J., et al. 2019, IEEE Transactions on Parallel and Distributed Systems, 31, 94

Liu, W., Meng, Q., Wang, C., et al. 2021, Journal of Instrumentation, 16, P08047

MacMahon, D. H., Price, D. C., Lebofsky, M., et al. 2018, Publications of the Astronomical Society of the Pacific, 130, 044502

Romein, J. W. 2021, Astronomy & Astrophysics, 656, A52

Schilizzi, R., Alexander, P., Cordes, J., et al. 2007, SKA Memorandum, 100

Yu, Z., Deng, F., Sun, S., et al. 2024, Research in Astronomy and Astrophysics