A RDMA Interface for Ultra-Fast Ultrasound Data-Streaming over an Optical Link

Andrea Cossettini*, Konstantin Taranov†, Christian Vogt‡, Michele Magno‡, Torsten Hoefler†, Luca Benini*§

*Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland
†Scalable Parallel Computing Laboratory, ETH Zürich, Zürich, Switzerland
‡Center for Project Based Learning, ETH Zürich, Zürich, Switzerland
§DEI, University of Bologna, Bologna, Italy

***Abstract*—Digital ultrasound (US) probes integrate the analog-to-digital conversion directly on the probe and can be conveniently connected to commodity devices. Existing digital probes are however limited to a relatively small number of channels, do not guarantee access to the raw US data, or cannot operate at very high frame rates (e.g., due to exhaustion of computing and storage units on the receiving device). In this work, we present an open, compact, power-efficient, 192-channels digital US data acquisition system capable of streaming US data at transfer rates greater than 80 Gbps towards a host PC for ultra-high frame rate imaging (in the multi-kHz range). Our US probe is equipped with two power-efficient Field Programmable Gate Arrays (FPGAs) and is interfaced to the host PC with two optical-link 100G Ethernet connections. The high-speed performance is enabled by implementing a Remote Direct Memory Access (RDMA) communication protocol between the probe and the controlling PC, that utilizes a high-performance Non-Volatile Memory Express (NVMe) interface to store the streamed data. To the best of our knowledge, thanks to the achieved datarates, this is the first high-channel-count compact digital US platform capable of raw data streaming at frame rates of 20 kHz (for imaging at 3.5 cm depths), without the need for sparse sampling, consuming less than 40 W.**

***Index Terms*—ultrafast, ultrasound, FPGA, RDMA, NVMe**

I. Introduction

US is one of the most used medical imaging modalities (it is low cost, portable, without ionizing radiation, and with high temporal resolution and real-time capabilities). *Digital* US systems include analog-to-digital converters (ADCs) directly on the probe (instead of sending the received signals to a backend system using heavy and expensive analog cables), enabling to directly interface it to commodity devices performing the complete processing in software (*software-defined* systems) [1]. Digital probes operating at frame rates of several kHz (*Ultrafast US*, UFUS) [2] challenge the data interfaces. For example, a 32-channel probe, sampling at 40 MSPS with 12 bits resolution, generates data with a throughput of 15.4 Gbps, which is beyond the capabilities of standard links (e.g., 5 Gbps for USB 3.0). Consequently, most of the existing digital probes are limited to a very small number of channels, do not operate at high frame rates, or perform on-board data reduction to transmit only selected information [3]. However, providing access to raw data (*open* US platforms) is critical for the development of efficient learning algorithms. Besides, good temporal resolutions are necessary to monitor fast biological processes, and a large channel count is needed to improve image quality [4].

Recently, optical-link-based connectivity appeared as a viable solution to empower digital probes with high data throughput capabilities; however, first attempts to translate UFUS into compact, low-power, open digital probes are still severely limited in the number of channels [5] or the achieved frame rates [6]. This paper presents a digital, open, FPGA-based, 192-channels UFUS platform capable of overcoming all the aforementioned limitations by implementing a RDMA interface to a host PC over two optical links. We employ RDMA-capable network controllers (RNICs), which allow applications to receive messages without intermediate buffering or operating system involvement, reducing the Central Processing Unit (CPU) load of applications and guaranteeing low-latency communication. To store the collected data, we propose to utilize novel NVMe-capable storage devices that empower developers to access the storage device directly, bypassing the POSIX abstraction. Besides, data access to NVMe-capable devices is offloaded to Direct Memory Access (DMA) engines, requiring zero CPU cycles to access the data.

The main contributions of this paper are the following: (1) first-time implementation of a RDMA communication system in an FPGA-based digital UFUS probe; (2) achieved data rates of 90.4 Gbps per optical link with synthetic data, demonstrating the capability of the network system to exploit the full bandwidth of the link; (3) achieved data rates of 42 Gbps per optical link (84 Gbps total) with US data, demonstrating the capability of the 192-channels system to sustain the streaming of ADCs operated in continuous mode; (4) reached frame rates $>$ 20 kHz for a 192-channels, compact, fully-digital, open, power-efficient, optical-link-based UFUS probe, surpassing by almost 40$\times$ the performance of systems with similar channel count, power budget, and form factor [6]; (5) demonstrated NVMe-based storage of UFUS streamed data at 12 Gbps. These achievements are enabled by using commodity components, whose performance progresses yearly [7], as driven by the server and high-performance computing industry. Our unique approach can continue to leverage such improvements, thereby defining a roadmap for open, digital, UFUS systems.

II. Background

A. Optical-link based US platform

Fig. 1 shows the US probe used in this work [6]. The probe is based on twelve 16-channels high-speed US

pulsers (STHV1600, STMicroelectronics), six 32-channels analog front-end (AFE) modules (AFE58JD32, Texas Instruments), and two Xilinx Kintex Ultrascale+ FPGAs (XCKU5P). Each FPGA is interfaced to an optical link (FireFly, Samtec) for 100G Ethernet data transmission. A host PC is equipped with a dual-slot RNIC (ConnectX-5, Mellanox), interfaced to the motherboard (X99-E WS, Asus) via Peripheral Component Interconnect Express (PCIe) 3.0. If the ADCs of the AFEs are operated continuously, such 192-channels system can generate a throughput of 92.16 Gbps (15.36 Gbps per AFE).

Fig. 2 shows the high-level signal flow implemented in the FPGA of the probe [6]. The communication between the different modules is organized as AXI-Stream (AXIS) interfaces [8] with bit widths between 128 bit and 512 bit. Due to the increasing data volume throughout the signal chain (e.g., due to the addition of Ethernet/IP headers and physical layer coding), three different clock domains are used to allow for increasing data throughput while minimizing timing requirements and power consumption. Configuration and high-level system control (acquisition trigger, IP-address handling, AFE configuration, etc.) is achieved with a Xilinx microblaze soft-processor running within the signal processing clock domain.

B. RDMA and NVMe interfaces

RDMA is a mechanism allowing the network controller to access the main memory of the host PC. Memory accesses are performed using a dedicated DMA hardware without any CPU intervention. This is essential to achieve the targeted sustained data throughput, as the CPU would be completely swamped if it had to be involved. Applications make use of offloaded RDMA communication by directly sending asynchronous work requests to an RNIC, bypassing the operating system. As UFUS generates too much data to be buffered for re-transmission, we focus on unreliable connections, considering a lightweight unreliable protocol called UD, which is similar to an UDP socket, but supports zero-copy packet delivery. Among the possible architectures, RDMA over Converged Ethernet (RoCE) v2 [9] performs better than other protocols [10] and is implemented by encapsulating the InfiniBand [9] transport protocol into the conventional UDP protocol. The integrity of RDMA packets is protected by an Invariant Cyclic Redundancy Check (ICRC) checksum, encapsulated into the UDP payload, and by the Frame Check Sequence (FCS) checksum of the Ethernet link.

NVMe protocol allows applications to access storage directly via PCIe. NVMe protocol is similar to RDMA, with the communicating endpoints being the host CPU and the storage device. Like RDMA, applications make use of NVMe by directly posting asynchronous work requests to a storage device that uses specialized DMA hardware to directly access memory. As a result, NVMe delivers high-bandwidth, low-latency storage access with near-to-zero CPU cost.

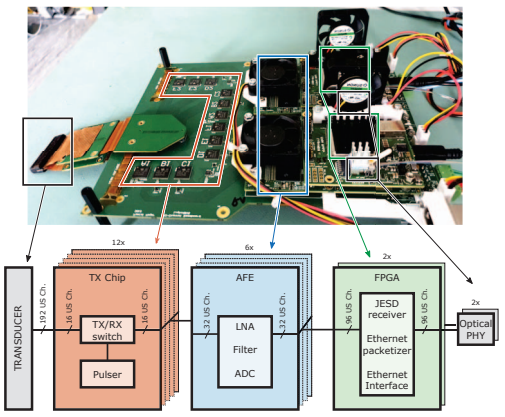

Fig. 1. Photo (top) and block diagram (bottom) of the main system components of the UFUS probe. A 192-channels linear array transducer is connected to twelve 16-channels high-speed US pulsers (STHV1600, STMicroelectronics) integrating transmit/receive (TX/RX) switches. Six 32-channels analog front-end (AFE) modules (AFE58JD32, Texas Instruments) amplify and sample the received signals (12-bit, 40 MSPS), sending digitized data via sixteen JESD204B serial outputs to two Xilinx Kintex Ultrascale+ FPGAs (XCKU5P) (three AFEs are connected to each FPGA, i.e., 96 US channels per FPGA) which aggregate the data for transmission to a host PC. Each FPGA is interfaced to an optical link (FireFly, Samtec) for 100G Ethernet data transmission.

III. System Design

A. FPGA: RDMA implementation

The RoCEv2 protocol requires to calculate an ICRC checksum. ICRC is a 32 bit Cyclic Redundancy Check (CRC) checksum of the invariant fields of the packet, based on the Ethernet polynomial [9]. To reduce clock speed and keep high throughput, our communication protocol utilizes a word length of 64 byte, which is processed in parallel during one clock cycle. Thus, the CRC checksum should also be calculated in a parallel fashion, and we do it following the approach of [11]. Fig. 3 shows the proposed FPGA RDMA module, which is structured in three main parts: (1) the input FIFO, containing one complete payload (max. 4096 byte); (2) the frame generator, which adds the required headers to the payload and a placeholder ICRC; (3) the ICRC of the transmitted RoCEv2 packet. A Xilinx CMAC Ethernet IP acts as a sink to the data stream. The data path between these main parts is structured as AXI-Stream (AXIS) bus with words of 64-byte length. The parallel processing of 64 bytes reduces the required clock speed to 160 MHz while allowing >100 Gbps

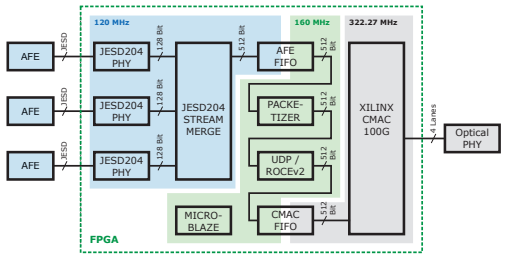

Fig. 2. Signal flow in the FPGA utilizing AXI-Stream links between the individual modules. AFEs data are recorded with a standard JESD204B interface (16 output channels, each containing data of two US transducers), de-interleaved, combined and stored in a First-In-First-Out (FIFO) memory (AFE-FIFO). The AFE-FIFO also decouples the clock domains of the AFE interfaces (120 MHz) and the signal processing (160 MHz) domain. Data from the AFE FIFO are then assembled into complete recording sequences (shots) and offered to the next processing block by the packetizer. Data of the shots are then split into multiple Ethernet packets by the User Datagram Protocol (UDP) packetizer. Transfer to a Xilinx CMAC IP-Core is managed via another FIFO memory (CMAC-FIFO), which is used as a bridge between the signal processing clock domain and the transmission clock domain (322.27 MHz).

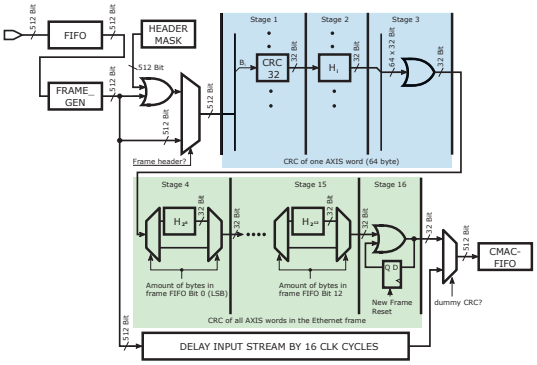

Fig. 3. Architecture of the RoCEv2 packet formation and ICRC calculation. Arbitrary packet lengths can be calculated correctly by either skipping or multiplying different H-matrices. The implemented pipeline works with a total of 16 stages, and it allows a maximum packet size of $2^{13} - 1 = 8191$ bytes.

throughput. The frame generator is triggered when the payload of an Ethernet packet gets in the packet FIFO. Then, it inserts the Ethernet, IP, UDP, and RoCEv2 headers into the 64-byte AXIS bus. Next, the payload from the FIFO is added to the data stream. Finally, a dummy ICRC is inserted as placeholder in the data stream once the FIFO is empty. This data stream is then processed in a pipelined fashion by the ICRC calculator (Fig. 3), which calculates the ICRC of each packet on all 64 byte of the AXIS bus in parallel, and finally inserts the ICRC at the correct position in the packet, replacing the dummy ICRC placeholder. In parallel to the ICRC calculation, the data stream is delayed by the same amount as the pipeline stages of the ICRC calculation. This enables the replacement of the dummy ICRC in the continuous data stream.

B. Host: RDMA stack and NVMe

Our system is equipped with a Samsung SSD 980 PRO 2TB, an NVMe-capable SSD that is attached to the M.2 PCIe slot. We implement direct access to the storage device with Storage Performance Development Kit [12]. Our RDMA receiver application is implemented with the RDMA core user library [13] and we use UD connections of RoCEv2 protocol. UD messages can consist of only one packet (maximum message size of 4096 bytes), and 40 bytes of each receive region are used to store information about the sender. Our approach consists in submitting multiple receive work requests of 4096 bytes for each packet of one shot and using null memory regions (as described in Fig. 4) to ensure that each shot is received in a contiguous buffer as it would be sent as one large message.

After receiving all parts of a shot, we submit one NVMe write request to write the shot to the storage device. NVMe

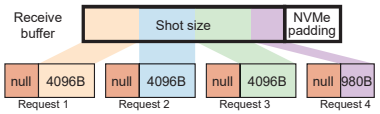

Fig. 4. The memory of a receive buffer is split into receive requests. As 40 bytes are used to store the information about the sender, we use null memory regions to discard the receiver information at the RNIC, and scatter-gather elements to use two memory regions in one receive request: each receive request consists of 40 bytes of the null region and 4096 bytes of the desired contiguous buffer. Thus, each shot is received in a contiguous buffer as if it were sent as one large message. The receive buffer is padded for aligned NVMe writes.

devices are addressed in blocks of 512 bytes, so we make each receive buffer to be the multiple of 512 bytes (see Fig. 4). After we receive a completion of the write request, we can reuse the buffer for receiving shots: we split the buffer in 4096 byte requests (with the null memory trick) and submit them to the RNIC. Note that we do not need to wait for a completion of the NVMe write request to submit the next request. NVMe supports submitting multiple write requests at the same time. To ensure steady writes of shots to the disk, we keep track of the load of the NVMe device by monitoring the number of pending requests. If the device is overloaded, we discard the received shot and immediately reuse the buffer for the network.

IV. Evaluation

We performed latency experiments to estimate the time required by the FPGA to react to a trigger event generated by the host and to send the data. To trigger the measurement, the host PC sends a special 64-byte UDP packet. The probe, in response, emits the acoustic wave according to the TX strategy, receives and samples the back-scattered acoustic echoes, and then sends the data to the PC. In this experiment, we measured the round trip time (RTT) between the emission of a trigger command from the PC and the reception (on the host) of the data, for a varying number of samples (N_s). The measured RTT of the single trigger command is approx. 5 µs. In contrast, the RTT for very small packet sizes is much higher (at least 560 µs), and it is determined by the time required by the FPGA to process the received command. Both protocols show an increased latency for increased N_s due to the increased number of packets to be transmitted. For UDP, this latency increase is $3\times$ larger compared to RDMA: for progressively larger packet sizes, the bottleneck of the UDP becomes more evident.

To assess the actual bandwidth and data transfer rate during real experiments, we operated the system in an external trigger mode, employing an external benchtop waveform generator to generate a periodic trigger at programmable frequencies f_{ext}, triggering measurements at any given shot frequency. Fig. 5 shows the achieved frame rate (FR) and bandwidth (BW) as a function of f_{ext} when the UDP protocol is used for the data transmission, for varying N_s. UDP faces a bandwidth bottleneck around 22-25 Gbps, which in turn results in a saturation of the maximum achieved frame rate. A smaller N_s yields a larger FR due to the shorter data transmission time for each measurement. Furthermore, for increasing f_{ext}, the number of missed packets presents a large variability. These results further demonstrate that UDP is not capable of sustaining the required bandwidth for the application.

Fig. 6 shows the corresponding results for the RDMA case. A bandwidth >40 Gbps was achieved for the $N_s = 4000$ case (progressively reduced for smaller N_s due to the overheads of smaller sets of data). Remembering that the maximum achievable throughput of 96 AFE channels (connected to one FPGA) is ≈46 Gbps, the RDMA implementation nearly reaches this limit. The corresponding frame rate is nearly 10 kHz at $N_s = 4000$ and approaches 20 kHz for $N_s = 2000$. When f_{ext} is further increased, the bandwidth and frame rate get halved

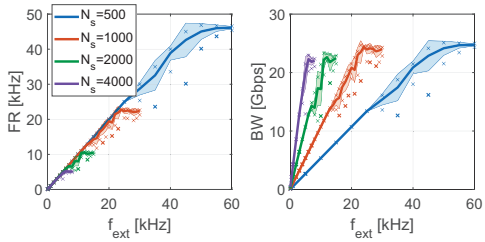

Fig. 5. UDP measured frame rate (FR, left) and bandwidth (BW, right) on each optical link as a function of f_{ext}. Results are reported for different choices of the number of samples (N_s) per shot. Thick colored lines represent the mean values, shaded areas represent the standard deviation, cross symbols represent the maximum/minimum values for each measurement.

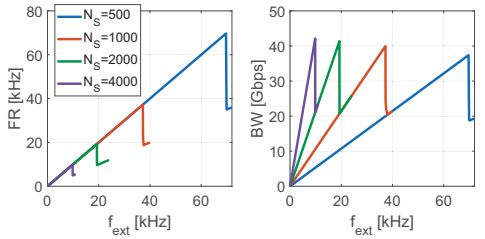

Fig. 6. Same as Fig. 5, for the RDMA case. Deviations from the mean are smaller than 0.1%, proving the more stable performance of the RDMA solution.

(with a too-high f_{ext}, one trigger event every two is missed, as the FPGA is still in the previous transmission phase).

Table I summarizes the maximum achieved frame rate and datarate. The performance of similar solutions with comparable channel-count [6] is surpassed by 39×. The maximum achieved cumulative bandwidth (two optical links) is 84.26 Gbps.

To test what is the actual maximum bandwidth of the FPGA-PC interconnect (overcoming the bandwidth limit set by the AFEs), we also implemented a synthetic data generation directly on the FPGA, achieving a bandwidth of 90.4 Gbps (not shown). This result demonstrates that even higher datarates are within reach if faster AFEs or more channels are used.

Fig. 7 shows the measured bandwidth and dropped shots for writing US data on the NVMe device at different f_{ext}. The NVMe supported 12 Gbps writing speed; however, fluctuations in the performance of NVMe yields drops of shots even at lower write bandwidths (e.g., f_{ext}=5 kHz). As expected, transmitting more data by means of increased f_{ext} results in a progressively larger number of dropped shots. We drop shots to maintain the overall progress of the application and to prevent overflow of NVMe work queues. The NVMe represents the current system bandwidth bottleneck, and its reduced write speed compared to the declared value (5,100 MB/s) is due to the motherboard, which does not support PCIe 4.0 for the NVMe.

Without any power management active, the system consumes

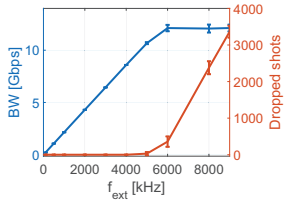

Fig. 7. NVMe measured bandwidth (blue) and number of dropped shots (red) during an acquisition with N_s=2000 number of samples per shot.

TABLE I
MAXIMUM DATARATE AND FRAME RATE ACHIEVED FOR DIFFERENT SHOT SIZES (2 OPTICAL LINKS, 192 US CHANNELS)

Max imaging speed	**Shot size**			
	500	***1000***	***2000***	***4000***
Frame rate [kHz]	69.68	37.24	19.28	9.82
Datarate [Gbps]	74.76	79.90	82.74	84.26

in total approx. 40 W (50% for the AFEs and FPGAs). Future designs could improve the power performance by relying on improved power management strategies and novel off-the-shelf components with enhanced energy efficiencies.

V. Conclusion and Future Work

We presented the implementation of a RDMA interface on a 192-channels, compact, open, high-speed, optical-link based UFUS probe. The RDMA-capable UFUS probe streams raw US data over two 100G Ethernet optical links with a maximum bandwidth of 84 Gbps, effectively enabling UFUS imaging at very high frame rates ($>$ 10 kHz). On the host side, data are stored by means of a 12 Gbps bandwidth NVMe storage unit. The NVMe appears as the current bandwidth bottleneck of the whole system, and as future work we will move to a different motherboard for increased communication speed with the NVMe, and we will implement compression algorithms to reduce the amount of data to be stored. As the proposed solution relies on commodity components that improve at very fast rates every generation [7], the performance of our platform can scale accordingly, and our approach effectively defines a roadmap for next-generation open, digital, UFUS systems.

Acknowledgment

This work was supported by the Swiss National Science Foundation (Project PEDESITE, No. 193813), by the European Research Council (Project DAPP, No. 678880) and (Project RED-SEA, No. 955776), and by Microsoft Research through its Swiss Joint Research Center.

References

[1] E. Boni *et al.*, "Ultrasound open platforms for next-generation imaging technique development," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 65, no. 7, pp. 1078–1092, 2018.
[2] T. Deffieux *et al.*, "Ultrafast imaging of in vivo muscle contraction using ultrasound," *Applied physics letters*, vol. 89, no. 18, p. 184107, 2006.
[3] Butterfly Network, "Butterfly IQ+ Ultrasound scanner," https://www.butterflynetwork.com/iq, Accessed: 14-09-2021.
[4] V. Chan and A. Perlas, "Basics of ultrasound imaging," in *Atlas of ultrasound-guided procedures in interventional pain management*. Springer, 2011, pp. 13–19.
[5] P. A. Hager and L. Benini, "Lightprobe: A digital ultrasound probe for software-defined ultrafast imaging," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 4, pp. 747–760, 2019.
[6] Ç. Özsoy *et al.*, "Lightspeed: A compact, high-speed optical-link-based 3d optoacoustic imager," *IEEE Transactions on Medical Imaging*, vol. 40, no. 8, pp. 2023–2029, 2021.
[7] InfiniBand, "InfiniBand Roadmap," 2020. [Online]. Available: https://www.infinibandta.org/infiniband-roadmap/
[8] Xilinx, Inc., "UG761 - AXI Reference Guide," Mar. 2011.
[9] InfiniBand, "InfiniBand Architecture Specification Volume 1, Release 1.4," Apr. 2020. [Online]. Available: https://www.infinibandta.org/
[10] Mellanox Technologies, "RoCE vs. iWARP Competitive Analysis," Tech. Rep. 15-4514WP, 2017.
[11] M. Walma, "Pipelined cyclic redundancy check (CRC) calculation," in *Proc. 16th Int. Conf. Comput. Commun. Netw.*, 2007, pp. 365–370.
[12] "Storage Performance Development Kit," 2018. [Online]. Available: https://spdk.io/
[13] "RDMA core userspace libraries and daemons," 2017. [Online]. Available: https://github.com/linux-rdma/rdma-core