White Paper

RDMA-Based Reflective Memory Low-Latency Real-Time Networks for XIL Systems

genüen
ideas taking flight

Introduction

The whitepaper Modern XIL Challenges [1] discussed the most typical problems currently constraining the advancement of distributed real-time XIL simulation systems. It demonstrated how low-latency real- time networks higher performants than the state-of-the-art are required for significant progress to be made on modern distributed real-time XIL systems. It presented the shortcomings of the current custom solution offered by industry vendors and alluded to the promise of Reflective Memory (RM) technology as a candidate solution to the problem.

This paper proposes a potential solution to the problem. It offers the utilization of Remote Direct Memory Access (RDMA) [2] to significantly improve the typical implementation of RM networks. It also shows how the two technologies can be combined onto an off-the-shelf open architecture solution that implements a low-latency real-time network on existing infrastructure that is orders of magnitude faster than the current state of the art.

The RDMA Technology

One of the obvious challenges of distributed XIL systems, real-time or not, is data exchange between its multiple nodes. Data traffic has been an area of study for several years. Some decades ago, the focus of the literature was on the creation of data transfer protocols that allowed sufficient data to be moved from point A to point B. This focus yielded different data exchange protocols that dominated the industry such as the typical RS-232 [3], RS-485 [4], and GPIB [5]. As data transfer requirements increased on more modern applications, these protocols somewhat fell out of favor and started to be replaced by PCI bus and consequently PCIe [6] bus, offering of order of magnitude higher data transfer specifications.

As the data volume capacity and transfer rate increased, one characteristic of the protocol-based approaches remained unchanged: the need for CPU time to be allocated to the data movement task from these buses to the processing unit memory space. As data volume and transfer rate requirements increased, so did the complexity of the data consuming applications, driving the need for processing power to be even higher. The increase of processing power on available low-cost CPU chips indeed followed Moore's law. However, this approach quickly found the wall of processor temperature control, hampering the long-term needs for the continuously increasing data processing capacity.

Direct Memory Access (DMA) [7] was the community's answer to this limitation. The concept of DMA was as simple as it was elegant. Data movement from the PCI/PCIe bus to the processing unit memory would be CPU-free, executed by a separate entity, the DMA controller. With this, CPUs were free from

the taxing data transferring task, and could be dedicated to the processing data application. DMA on PCI/PCIe bus data transfer has indeed been and still is currently used on data exchange systems. This technique however is applied to single processor systems.

In the realm of distributed systems, the data exchange problem remained unsolved as computational cost of network I/O is very high once the CPU kernel is involved in the

handling of network traffic. Unless a new paradigm was proposed, this problem would most likely remain open even as network bandwidth on existing off-the-shelf infrastructure continued to improve, as the burden of data exchange would only increase for the system CPUs. This fact drove the community to focus research on expanding the single- CPU DMA paradigm to systems of multiple nodes. The high-performance computing industry was one of the biggest drivers of advancements of the technology. GPU clusters [8] and some cloud computing experiments [9] were made on high-performance parallel computing applications with some success.

However, these approaches were still insufficient to meet the low-latency requirements for the specific case of distributed real-time XIL applications.

The bullseye for distributed real-time XIL systems continued to be on a technology that could offer CPU- free low-latency data exchange between multiple distributed nodes. Remote Direct Memory Access (RDMA) is a protocol aimed directly at solving the high-performance computing key steps [10]. Its chief concept is the utilization of hardware-based network interface cards (NICs), often FPGA-based, that are responsible not only for performing protocol control, but also for executing direct read/write memory operations on remote nodes. The concept is the natural expansion of the single-CPU DMA idea onto several distributed nodes.

As the CPU is not involved in the data exchange tasks and the dedicated NIC hardware is responsible for read/write remote node memory operations, latency of such networks can theoretically decrease significantly. Since the concept's inception in the early 2000s, significant progress was made on the commercialization of the technology. This allowed for practical implementations to confirm that RDMA indeed is a significant network traffic latency reducer. Pre-RDMA latency measurements typically revolved around the millisecond range started to migrate to the tens of microseconds [11], culminating on some sub-microsecond clock synchronization [12] over Infiniband [13] networks.

Reflective Memory

Another technology of interest to the problem of message passing in multi-node systems is Reflective Memory (RM) [14]. Like the RDMA technology presented in the previous section, RM's main target was the addressing of latency of distributed network applications. However, RM was designed to facilitate multi-cast operations, an advantage over RDMA for distributed real-time applications.

On an RM network, when a node needs to make a write operation to the RM network, it places a typical TX request to the next node in the ring. The next node in turn reads the data, updates its local reflective memory, and makes a TX request with the same data that it just read to the next node down the ring network. The process is repeated until the data makes a complete round in the network and is returned to the original writer node. One of the main advantages of this technique is that RM nodes can make updates to a mapped local memory range independently from the node's CPU [15]. Therefore, the local RM copy can be updated at a much higher speed than a given application running on the local CPU that will in turn consume the data. This allows for a higher-speed hardware-based operation to guarantee a level of real-time determinism in updating the virtual local memory.

Figure 1 illustrates an anecdotal RM network with four nodes. It is important to note that each NIC of the network is responsible for controlling traffic and maintaining a local copy of the RM. The application running on the computer can then access data from the RM as needed.

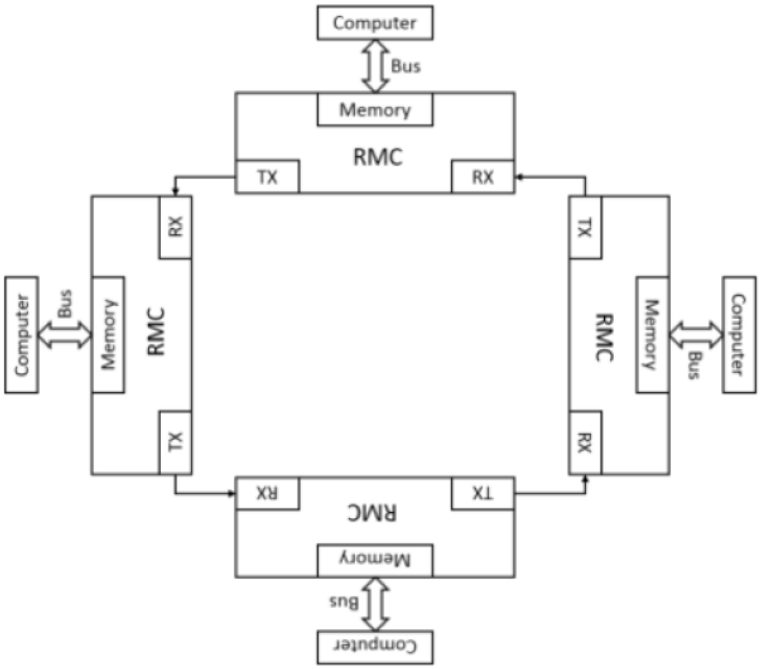

Figure 1. Reflective Memory Typical Network

The concept of maintaining a local copy of the network state information to accelerate data access and reduce latency indeed holds great potential as a solution to the distributed real-time network problem. As such, there were several RM implementation variations over the years in the search for a silver bullet to the problem. However, scalability of the network is a permanent issue across all different implementations. The RM topology illustrated in Figure 1 has each NIC being a single point of failure for the network. This fact drove the addition of redundancy nodes to accommodate mission-critical applications as one of the possible workarounds. However, since each node maintains a copy of the entire RM, the addition of redundant nodes creates an immediate increase in RM size, which in turn creates a detrimentally proportional relationship with latency.

A second obvious issue with this approach is that the RM network is not an off-the-shelf solution. It cannot utilize network infrastructure that may already be in place in the building, requiring specific and point-to-point installation.

The RM concept indeed carries promise, but a successful implementation must address the typical scalability and network infrastructure shortcomings.

Proposed Solution

The discussion presented so far suggests that the solution for the distributed low-latency real-time network problem may be found in a specific implementation of RM that solves its two typical shortcomings. This paper proposes the utilization of the RDMA technology on a RM implementation to address the issues.

As presented, each node on a RM network is a single point of failure. One path to address this limitation would be the implementation of a star topology RM network. However, once the RM nodes are distributed in a star topology, the original paradigm of read message->update RM->pass message along is no longer applicable. At this point, a switched solution becomes necessary, bringing back the problems of the other more typical data exchanged approaches explored previously.

This paper proposes leveraging RDMA technology for the data exchange layer of an RM implementation that potentially doesn't carry the same issues as its typical implementations. Since RDMA was demonstrated to be a viable low-latency data exchange approach, it holds the potential to enable a star topology RM implementation.

The second issue presented that is characteristic of RM networks is the requirement for a custom network that doesn't utilize existing infrastructure. RDMA was originally deployed over Infiniband hardware. More recently, however, the need for reuse of existing network infrastructure motivated the community to expand RDMA to be deployed over what is called converged ethernet, or RoCE [16].

An important positive characteristic of RMs that made them a good fit for distributed real-time applications is the easy of multicast operations. It was therefore important that a star topology RM could maintain the comparable multicasting facilities. The answer to this requirement comes from an atomic broadcast protocol over RDMA implementation [17]. Broadcast protocols make distributed services fault tolerant, as they keep a total order of messages, allowing that multiple service replicas are kept in sync. However, they are usually computationally expensive. The protocol utilized by the proposed approach performs communication using one-sided RDMA writes, which do not utilize the remote machine CPU, and is designed to minimize waiting on the critical path. Figure 2 illustrates the high-level architecture of the proposed solution.

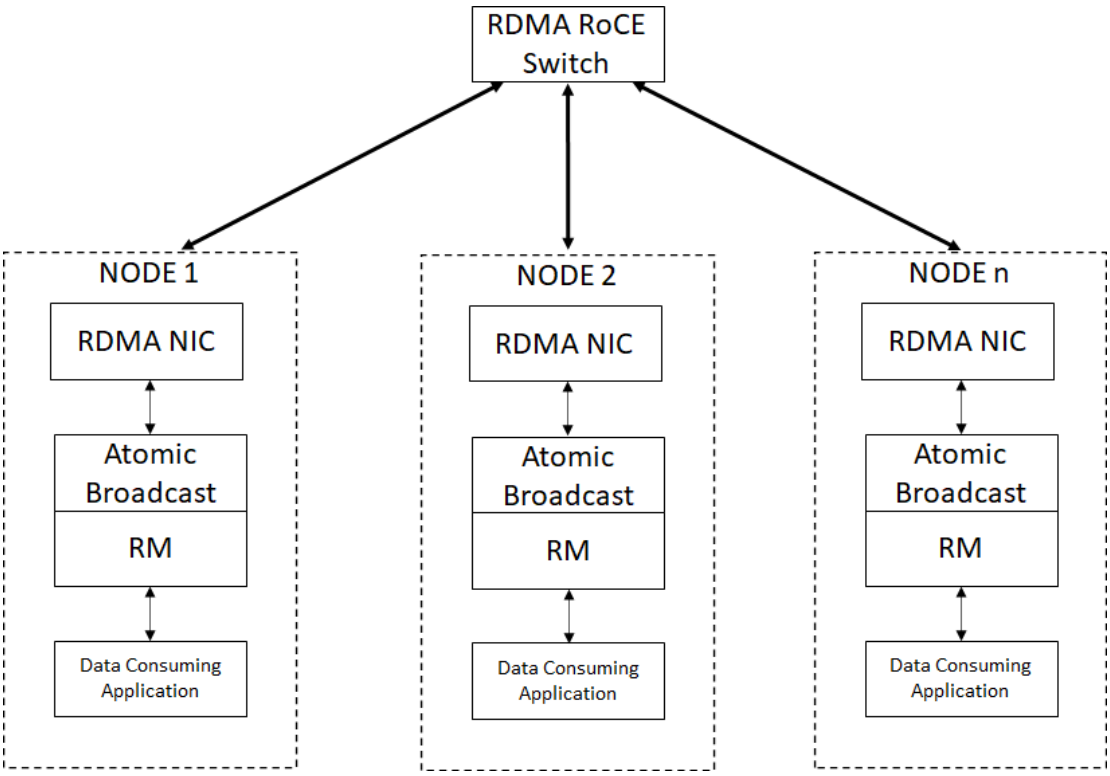

Figure 2. Proposed Solution Architecture

The proposed solution theoretically allows the RDMA-based RM implementation to maintain the strengths of the typical RM solutions: CPU disengagement, low-latency synchronization, and appropriate multicasting facilities. On the flip side, it strives to remove the constrains that arguably prevented typical RM implementations from being more widely used on distributed real-time XIL systems: single point of failure, scalability, and reuse of custom network infrastructure.

Experimental Results

Experiments were performed on Cloudlab, an open platform for running network experiments that gives exclusive access to the nodes [18]. In particular, the experiments used a cluster with nodes that have an Intel E5-2640v4 processor each running Ubuntu 18.04 (for reference, Cloudlab calls this cluster xl170). Each node has 64GB of DRAM and a dual-port Mellanox ConnectX-4 25 GB NIC. The experiment network is confined to a single chassis hosting a Mellanox 2410 switch that connects each core with 25Gb Ethernet links that support RDMA over Converged Ethernet (RoCE).

The RMs were all-to-all in the sense that all states of each node were broadcasted to every other node on the network. Therefore, a complete copy of all network states was kept as part of each node's RM. A state is defined as a C-style floating point number.

Figure 3 shows the plots of RM depth versus latency for an RM with 3, 5 and 7 nodes.

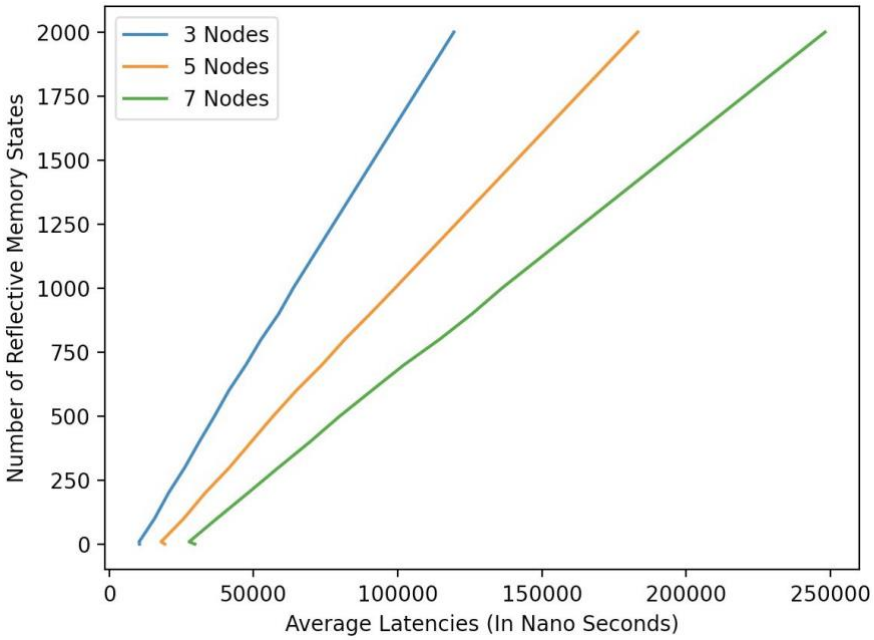

Figure 3. RM Depth Latencies

A linear ramp characterizes direct proportionality between the depth of the RM and network latency. In other terms, as the traffic on the network increases, the latency increases in linear proportion. As the number of nodes is increased, the linear proportionality is maintained. In conclusion, the latency of the RM network is directly proportional to the overall network traffic. Figure 4 characterizes the average latency per message size.

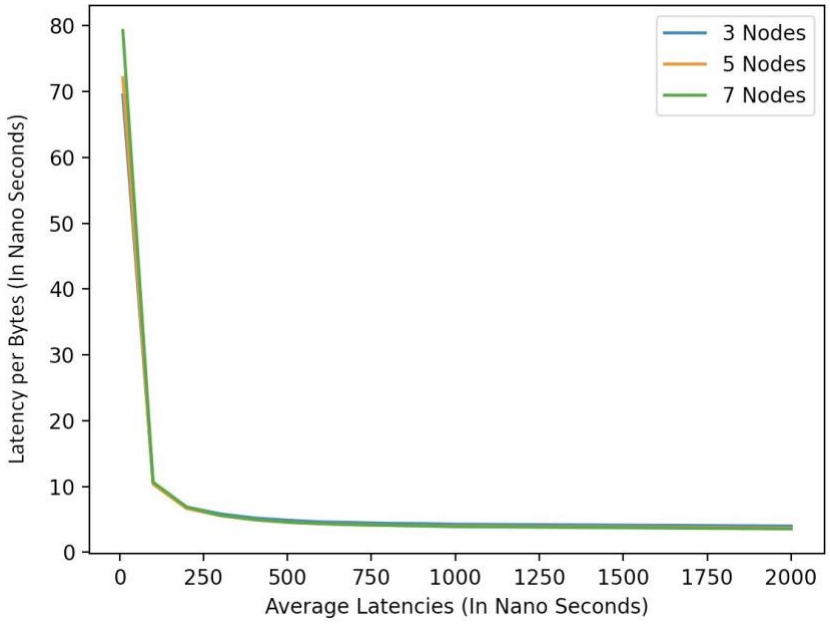

Figure 4. Latencies per Message Sizes

As the messages get larger in size, the per-byte computational cost goes down to a given limit fixed value. Below 100 nano seconds, the per-byte cost is high due to the minimum RDMA protocol packet size. Therefore, larger message transmissions shall take precedence over smaller more frequent ones on the application layer for an overall lower computational cost per byte transmitted.

Discussion and Future Work

Data exchange of distributed XIL applications doesn't require typically high network traffic as do more bandwidth-intensive applications such as data streaming. The main objective of the data exchange on distributed XIL systems stems from the fact that each node's computation may require information that is being generated by one or more of the distributed nodes, in the same real-time clock tick. Therefore, a network traffic of several thousand states can accommodate a significant number of remote nodes exchanging several states between each other.

As shown in Figure 3, latency values that would support rates well over 10kHz can be achieved to accommodate significant enough traffic volume for distributed real-time XIL applications. These initial experiment results show the proposed approach to be a promising candidate to implement real-time distributed networks for this application category.

Moreover, the current implementation can be optimized in a few specific ways to improve results further. The source code can be refactored for performance as the first pass of its implementation was focused on getting to the point of a qualitative order of magnitude analysis for approach feasibility as early as possible. The validation of the solution through the first pass analysis presented justifies refactoring of the prototype-level source code to one of production quality which very likely will improve the latency results further.

Since it was demonstrated that network traffic is directly proportional to communication latency, some techniques can be applied to expand the utility of the solution to a greater number of XIL applications. The first one that shall be considered is the implementation of a publish-subscribe approach to the creation of the multiple node RMs, as opposed to the all-to-all method used on the first analysis.

In the all-to-all approach, all states of all nodes are transferred and maintained in each RM node. In the publish-subscribe model, the network is configured with a priori information of which states from which nodes are relevant to a given node's computation. Each node's RM depth is then reduced to maintain only the states that will be used by the corresponding node computation, instead of being a complete copy of all network nodes' states. This evidently is an application specific improvement, albeit an important one. In the all-to-all implementation, the RM real-time updates need to be the same for all nodes. In the publish-subscribe model, the RM network gains an extra level of flexibility where different nodes' RMs may have different real-time update rates, depending on their depth. Without loss of generality, this indicates that multi-rate XIL systems can then be implemented.

Ultimately, data compression may also be a technique that can potentially further reduce the overall network traffic. This may also carry an application specific aspect to the implementation as it will depend on the type of state that a given application requires.

Therefore, data compression may have a bigger or smaller impact on the overall network traffic, depending on the characteristics of the system states for a given application.

Lastly, as noted in the experimental results section, the tests were executed on nodes running regular non-real-time operating systems. Once the above network traffic reduction techniques are implemented, the next natural step is the execution of the same tests on an RM network composed only of nodes running real-time operating systems for comparison of performance.

Conclusion

This paper presented a review of the existing network data exchange technologies, with special focus on potential solutions for the distributed low-latency real-time network problem. It presented RDMA as a low-latency data exchange technology of interest and RM as a candidate for a proposed solution, provided that two important shortcomings were addressed: scalability and lack of off-the-shelf offering.

It proposed an RDMA-based RM implementation as a possible solution to the problem. The approach suggests the utilization of RoCE to facilitate the use of existing network infrastructure. It implements RM on top of RDMA atomic broadcast operations to reduce the typical computational burden of broadcast operations that are required to keep distributed systems fault tolerant.

Initial experiment results demonstrated excellent latency performance on an all-to-all RM implementation. Data suggests that a vast majority of real-time distributed XIL systems could be implemented on tens of microsecond real-time loop rates. The data also showed latency to be directly proportional to network traffic. Therefore, future work shall focus on reduction of network traffic through either data compression, a publish-subscribe approach to reduce the amount of data kept by each local copy of RM, or a combination of both.

References

1. Altoe, F (2022). Modern XIL Challenges.
2. Romanow, A., & Bailey, S. (2003, February). An Overview of RDMA over IP. In Proceedings of the First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003).
3. Monteiro, A., & Jordan, T. R. (2004). Implementing communication between Windows PCs and test equipment using RS-232 and Borland C++ Builder. Behavior Research Methods, Instruments, & Computers, 36(1), 107-112.
4. Axelson, J. (1999). Designing RS-485 circuits. Circuit Cellar, 107, 20-24.
5. Tompkins, W. J., & Webster, J. G. (1988). Interfacing Sensors to the IBM PC (p. 2). Englewood Cliffs: Prentice Hall.
6. Bohm, P. (2010). Incremental and verified modeling of the PCI express protocol. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29(10), 1495-1508.
7. Riesbeck, C. K., & Martin, C. (1986). Direct memory access parsing. Experience, memory, and reasoning, 209-226.
8. Fan, Z., Qiu, F., Kaufman, A., & Yoakum-Stover, S. (2004, November). GPU cluster for high performance computing. In SC'04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing (pp. 47-47). IEEE.
9. Ledyayev, R., & Richter, H. (2014). High performance computing in a cloud using openstack. Cloud Computing, 108-113.
10. Von Eicken, T., Basu, A., Buch, V., & Vogels, W. (1995). U-Net: A user-level network interface for parallel and distributed computing. ACM SIGOPS Operating Systems Review, 29(5), 40-53.
11. Liu, J., Wu, J., Kini, S. P., Wyckoff, P., & Panda, D. K. (2003, June). High performance RDMA-based MPI implementation over InfiniBand. In Proceedings of the 17th annual international conference on Supercomputing (pp. 295-304).
12. Litz, H., Fröening, H., Nuessle, M., & Brüening, U. (2007). A hypertransport network interface controller for ultra-low latency message transfers. HyperTransport Consortium White Paper.
13. Pfister, G. F. (2001). An introduction to the infiniband architecture. High performance mass storage and parallel I/O, 42(617-632), 102.
14. Jovanovic, M., & Milutinovic, V. (1999). An overview of reflective memory systems. IEEE concurrency, 7(2), 56-64.
15. Bian, Z., Xie, L., & Wu, B. The FPGA Design and Implementation of Reflective Memory Card Based on the PCIE Bus.
16.
17. Beck, M., & Kagan, M. (2011, September). Performance evaluation of the RDMA over ethernet (RoCE) standard in enterprise data centers infrastructure. In Proceedings of the 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching (pp. 9-15).
18. Izraelevitz, J., Wang, G., Hanscom, R., Silvers, K., Lehman, T. S., Chockler, G., & Gotsman, A. (2022). Acuerdo: Fast Atomic Broadcast over RDMA.
19. University of Utah. 2022. CloudLab. (2022). https://www.cloudlab.us/.