Latency-Optimized Networks for Clustering FPGAs

Trevor Bunker Steven Swanson
Computer Science & Engineering
University of California, San Diego
{*tbunker,swanson*}*@eng.ucsd.edu*

***Abstract*—The data-intensive applications that will shape computing in the coming decades require scalable architectures that incorporate scalable data and compute resources and can support random requests to unstructured (e.g., logs) and semi-structured (e.g., large graph, XML) data sets. To explore the suitability of FPGAs for these computations, we are constructing an FPGA-based system with a memory capacity of 512 GB from a collection of 32 Virtex-5 FPGAs spread across 8 enclosures.**

This paper describes our work in exploring alternative interconnect technologies and network topologies for FPGA-based clusters. The diverse interconnects combine inter-enclosure high-speed serial links and wide, single-ended intra-enclosure on-board traces with network topologies that balance network diameter, network throughput, and FPGA resource usage. We discuss the architecture of high-radix routers in FPGAs that optimize for the asymmetry between the inter- and intra-enclosure links. We analyze the various interconnects that aim to efficiently utilize the prototype's total switching capacity of 2.43 Tb/s. The networks we present have aggregate throughputs up to 51.4 GB/s for random traffic, diameters as low as 845 nanoseconds, and consume less than 12% of the FPGAs' logic resources.

***Keywords*-FPGA; high-radix router; low latency network; high-speed serial; data-intensive applications;**

I. Introduction

Data-intensive applications are becoming central to discovery-based science in biology, medicine, security, and the social sciences. The increasing importance of these applications calls for new computing architectures that can analyze massive data sets (e.g., large graphs, unstructured corpora of text, and genome data) efficiently. These workloads issue many random memory requests that typically exhibit little locality. Therefore, traditional techniques for hiding memory latencies (e.g., caching, prefetching, out-of-order execution) provide little benefit.

Massive data-level and thread-level parallelism can reduce the penalty of long memory access latencies. Traditional microarchitectures allow each thread to issue multiple outstanding memory requests and use context-switching to sustain dozens of outstanding memory requests. However, traditional CPUs can not fully utilize the available memory bandwidth. Graphics Processing Units (GPUs) can schedule hundreds of threads, but these threads can only access a few GBs of memory. Specialized microarchitectures like the Cray XMT and IBM BlueGene offer large amounts of parallelism and scalable data, but have higher costs to achieve equivalent or greater performance.

FPGAs are an appealing alternative to traditional CPUs, GPUs, and specialized architectures. First, FPGAs are reconfigurable and can be programmed to handle specific computation-intensive or data-intensive applications. Second, they offer a large amount of configurable I/O pins that can be used for memory controllers or networking. Third, their interconnect fabrics are optimized for wide parallel pipelines, which gives them an advantage for data-level parallelism. Finally, FPGAs can minimize memory access latency by placing the memory controller as close as possible to the logic that utilizes the data.

To explore the suitability of FPGAs for data-intensive applications, we are constructing an FPGA-based system with a memory capacity of 512 GB from a collection of 32 Virtex-5 FPGAs spread across 8 enclosures. Each FPGA has 8 configurable high-speed serial lanes and wide, single-ended and differential connections to adjacent intra-enclosure FPGAs. Utilizing the high-speed serial lanes, we can create a custom network to connect all 32 FPGAs across all 8 enclosures. However, as the system scales to 32 FPGAs, reducing network diameter, and therefore remote memory access latency, becomes a critical design constraint.

In designing the interconnect technologies and network topologies of the systems, we faced many design trade-offs in balancing network diameter, network throughput, and FPGA resource usage.

- *Network Diameter* - High-radix routers minimize network diameter by increasing the connectivity at each FPGA [1]. However, increasing connectivity in our prototype implies using narrow, low-bandwidth high-speed serial lanes and this increases resource consumption.
- *Network Throughput* - Wider high-speed serial channels formed by bonding multiple lanes can sustain higher network bandwidth, but topologies based on these channels typically require more network hops.

- *FPGA Resource Usage* - FPGA resource usage increases as the number of channels an FPGA supports increases.

In this paper, we explore the design and performance of various interconnect technologies and network topologies in terms of their network diameter, network throughput, and FPGA resource usage. We describe the architecture of a hybrid network: a fast, fully-connected intra-enclosure network and a high-radix inter-enclosure network for scaling. We explain the design of a router that compensates for the asymmetric performance of inter-enclosure and intra-enclosure links. The networks we present have aggregate throughputs up to 51.4 GB/s, diameters as low as 845 nanoseconds, and consume less than 12% of the FPGAs' slices.

The organization of the paper is as follows. In Section II, we summarize the related work in designing networks for FPGA-based clusters. Section III presents the design of the various transceivers and routers that serve as architectural building blocks for our experimental topologies. In Section IV, we introduce the five network topologies and their routing protocols that we evaluate (Section V). Finally, we conclude in Section VI and mention the future work planned for this system.

II. Related Work

The interconnects that multi-FPGA systems use vary with their target application and the underlying hardware structure. Below, we survey a few previous efforts. Our driving goal is low latency on small packets to support low-locality reads and writes to remote memories.

Axel [2] is a heterogeneous supercomputer that targets compute-intensive applications. It comprises a cluster of heterogeneous nodes consisting of a CPU, an FPGA, and a GPU connected via PCIe. The CPUs of each node communicate over Gigabit Ethernet, and the FPGAs of each node communicate over an Infiniband network.

Maxwell [3] used a 2-D torus interconnect between 64 FPGAS to explore whether an FPGA-based cluster could support many types of high-performance computing applications. Each link in the interconnect used a single multi-gigabit transceiver.

Cube [4] is a 512-FPGA systolic array system. For applications that do not fit this paradigm, communications latencies are very long. The network topology was an 8x8x8 3-D Mesh, not optimized for latency.

The system in [5] attached multiple boards – each with 9 FPGAs fully-connected by multi-gigabit transceivers – to a central node via 10 Gb/s links. The authors focused on accelerating applications with high computation-to-communication ratios (e.g., molecular dynamics simulations), instead of data-intensive applications.

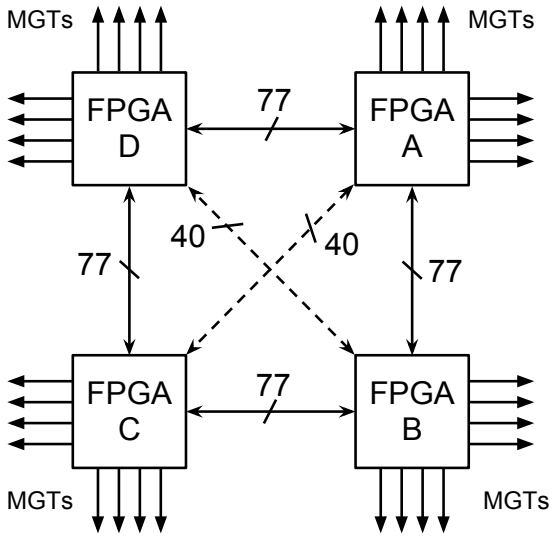


Figure 1. **BEE3 channels** The BEE3 is an FPGA prototyping platform that consists of four fully-connected Virtex-5 FPGAs. Each FPGA also has eight available multi-gigabit transceivers for implementing various high-speed serial protocols (e.g., Ethernet, Aurora, RapidIO).

The Reconfigurable Computing Cluster Project [6] built a prototype FPGA cluster to explore the effectiveness of using FPGAs in cost-effective petascale supercomputers. Their board [7] consists of a Virtex-4 FPGA with 8 multi-gigabit transceivers connected to SATA cables. This allowed for k-ary, n-cube network topologies.

The RAMP Blue Project [8] is most similar to our system. It uses the BEE2 board, which provides similar intra- and inter-enclosure bandwidth. It was designed to emulate a distributed-memory message-passing architecture. They use a lower-radix topology than our system, but they also optimized for latency.

III. Architecture

FPGAs serve as a powerful platform for experimenting with network technologies and topologies because they have a flexible switching fabric and a large number of customizable I/O pins. In fact, newer FPGAs include dozens of multi-gigabit transceivers that can operate at line rates of up to 28.05 Gb/s [9].

Our initial designs target the BEE3 [10] multi-FPGA prototyping platform that provides a unique combination of gate capacity, DRAM capacity, and communication bandwidth. It consists of four Xilinx Virtex-5 XC5LX155T FPGAs', each of which hosts up to 16 GB of DDR2 DRAM. In this section, we describe the various architectural modules we created to experiment with alternative topologies including the transceivers and routers.

A. Intra-enclosure Channels

Figure 1 shows the communication channels that the BEE3 provides. The solid lines connecting each pair of FPGAs in the horizontal and vertical dimensions form the *ring network* and each line represents a set of 77 single-ended wires.

In our design, each link of the ring network is a bidirectional, source-synchronous channel that can transmit and receive data at 1.86 GB/s with a latency of 20 ns. Each direction of the channel consists of 36 (32 for data, 4 for control) single-ended signals as well as a clock. Using the Xilinx IDDR and ODDR primitives, the wires switch at 500 mega-transfers/second. The channel self-calibrates on start-up using the Xilinx IODELAY primitives.

In addition, each FPGA attaches to the diagonally opposite FPGA via 40 differential pairs (the dashed lines in the figure). This cross channel is also a bidirectional, source-synchronous channel and runs at 1.86 GB/s with a latency of 24 ns. Each direction of the channel consists of 18 differential pairs (16 for data, 2 for control) and operates at a double-data rate of 1.0 GT/s. The transceiver module uses the Xilinx ISERDES and OSERDES primitives to perform 4:1 serialization and 1:4 deserialization of each pair. The receiver also uses the IODELAY primitives for adjusting the sampling window for initial calibration.

B. Inter-enclosure Channels

Each FPGA has eight multi-gigabit transceivers (MGTs) with a maximum line rate of 3.125 Gb/s. The MGTs from each FPGA connect to two 10GBASE-CX4 ports on the front panel of the BEE3. We implemented two different high-speed serial protocols that utilize the 10GBASE-CX4 ports for inter-enclosure communication.

1) 10 Gigabit Ethernet (10GbE): We created a 10 Gigabit Ethernet transceiver to utilize the 10GBASE-CX4 connectors. The transceiver uses the Xilinx XAUI [11] and 10GbE MAC [12] cores. The bidirectional channel operates at 10 Gb/s using raw Ethernet jumbo frames to support larger packet sizes. The cores implement link training, encoding, clock compensation, and checksum calculation/validation.

The latency of an Ethernet transmission depends on the size of the packet. This is because the Ethernet core validates a checksum before signaling that the packet is ready for consumption. In addition, the 10GbE switch introduces an additional latency penalty of 450 ns. Latencies for a 10GbE packet range from 840 ns to 7.324 μs.

2) Aurora: In addition to the 10GbE transceiver, we created four different versions of the Xilinx Aurora core [13]. This core is a lightweight, customizable high-speed serial transceiver that implements calibration, encoding, and clock compensation. Using the core generation tool, we generated 1-, 2-, 4-, and 8-lane versions of the transceiver. We also added XON/XOFF flow control and channel partner identification to each channel.

As we increase channel bonding, we increase the bandwidth of the channel yet decrease the number of available channels for networking. We will explore this trade-off in Section V.

We use copper 10GBASE-CX4 cables for the 4- and 8-lane versions of the Aurora transceivers. However, to use the 1- and 2-lane Aurora channels independently, we use a CX4-to-SATA breakout cable and use a coupler to bind the single-lane SATA cables.

The transceivers on the FPGAs are capable of transmitting and receiving at 3.125 Gb/s. Unfortunately, due to the length and shielding of the breakout cables, we were not able to achieve the expected line rate without observing bit errors. To tolerate poor signal integrity, we run each transceiver at 1.95 Gb/s. Although the 4- and 8-lane versions do not use the breakout cables, we use the same slow line rate for all versions when we present our results in Section V.

The latencies of the Aurora channels are proportional to the transceiver line rate. At a line rate of 1.95 Gb/s, the channel latency is 541 ns. To optimize for latency, we do not transmit or validate a checksum. The Aurora core provides 8B/10B encoding and checks for hard and soft errors.

C. Routers

We designed two types of routers that handle the intra- and inter-enclosure switching: a *full-crossbar* router and an *MGT* router. The full-crossbar router can handle routing to adjacent FPGAs within a BEE3 and to one or two inter-enclosure channels. However, creating a full-crossbar, high-radix router in an FPGA that is able to run at high clock frequencies is difficult due to long wires, multiple levels of logic, and limited resources. In addition, the large asymmetry between the bandwidths of narrow high-speed serial channels and the on-board links makes such a design overkill. To account for this asymmetry, we designed an MGT router that combines the full-crossbar router for the intra-enclosure links and a more frugal design for managing inter-enclosure traffic.

1) Full-crossbar router: Figure 2 depicts the block diagram of our full-crossbar router. It has a parameterizable number of input and output ports. For low-radix designs, all of the ports connect to their respective transceivers. The router runs at the system frequency of 250 MHz with a 64-bit datapath with a latency of 64 ns.

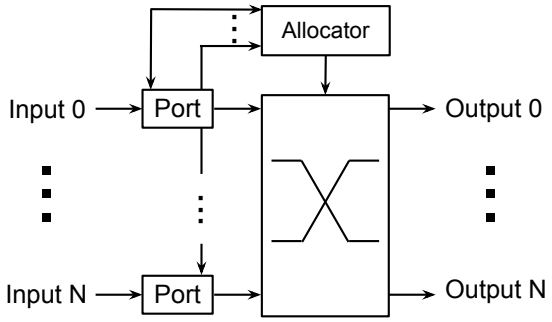


Figure 2. **Crossbar router block diagram** A full-crossbar router provides all-to-all switching between a configurable number of ports.

The module labeled *Port* maintains the state of the input port. For each incoming frame, the input port determines the destination route (dependent on network topology), requests time on the switch from the allocator, and monitors the flow control state of the downstream channel.

The allocator module uses simple iterative round-robin scheduling [14]. Once the allocator grants a request to an input port, that port's request becomes the lowest priority and the allocator will service all other pending requests before the priority cycles. This scheduling algorithm is strongly-fair and performs well under heavy, random traffic. We chose this scheduling algorithm because it would meet timing constraints with fewer resources than more advanced scheduling algorithms.

The switch is a mux-based switch that uses the output of the allocator as the select signals for each mux. Flip-flops on the inputs and outputs of the muxes alleviate timing problems.

2) MGT router: Figure 3 shows how the MGT router for high-radix topologies is based on two *subrouters*: a receive path for the serial inputs (left dotted box) and a transmit path for the serial outputs (right dotted box). The transmit and receive subrouters connect to the crossbar router (center dotted box), which provides the connectivity to the intra-enclosure FPGAs. The subrouters run at the system clock rate of 250 MHz. The MGT router hides the asymmetric performance of the inter-enclosure links by isolating them from the intra-enclosure router with large buffers.

On the receive path, all input ports arbitrate for access to the output buffer, which gives the high-speed serial ports access to the intra-enclosure router. This combines the input bandwidths of the high-speed serial channels (around 1.4 GB/s) into a single stream that roughly matches the bandwidths of the intra-enclosure channels (1.9 GB/s). The arbiter is a strongly-fair round-robin arbiter [14] that is equivalent to the intra-enclosure allocator's scheduling algorithm. The latency of the receive path is 48 ns.

On the transmit path, the input port removes data from the input buffer, calculates the destination port (dependent on the topology), and writes the data to the correct out-bound buffer. There is no arbitration for the decoder so the latency (40 ns) is slightly better than the receive path.

IV. Network Topologies

Using the components described above, we created various latency-optimized topologies that reflect different design goals: fewer FPGA resources, higher point-to-point throughput, and lower routing complexity.

All topologies use the same intra-enclosure network: the four FPGA nodes form a fully-connected network of low-latency, high-bandwidth channels.

The topologies we chose reflect the constraints of the BEE3 system and the types of interconnects it provides. While these topologies do not encompass every possible topology, we feel that they provide insight into how the entire class they represent would behave. We did not consider some traditional topologies, like tori and trees, because we are optimizing for latency and those topologies have inherently large diameters.

A. 10GbE Switch

Traditional clusters employ fully-switched topologies that leverage conventional switching protocols like Ethernet and InfiniBand. Switches let those systems offload routing logic to dedicated machines and release resources for more computation.

To understand the trade-off this entails, we connect one 10GBASE-CX4 port from each FPGA to one port of our two, 20-port 10GbE switches. Two of the four BEE3 FPGAs connect to one of the switches and the other two FPGAs connect to the other switch. This topology leverages the enormous switching bandwidth that commercial 10GbE switches offer and reduces the complexity of the FPGA routing protocol. With 32 FPGAs, there is only one Ethernet transmission for any inter-enclosure communication.

B. Aurora-based Topologies

The Aurora high-speed protocol offers the equivalent line rate performance of 10GbE without the overhead–in terms of FPGA resources and transmission latency–that Ethernet requires. In addition, a custom network based on the Aurora protocol would not require any additional switches, which would reduce system cost and failure points.

There are four variants of Aurora-based topologies, one for each channel width. In describing these topologies, we adopt a few naming conventions. First, we name each topology by the number of inter-enclosure channels each FPGA provides, as well as the

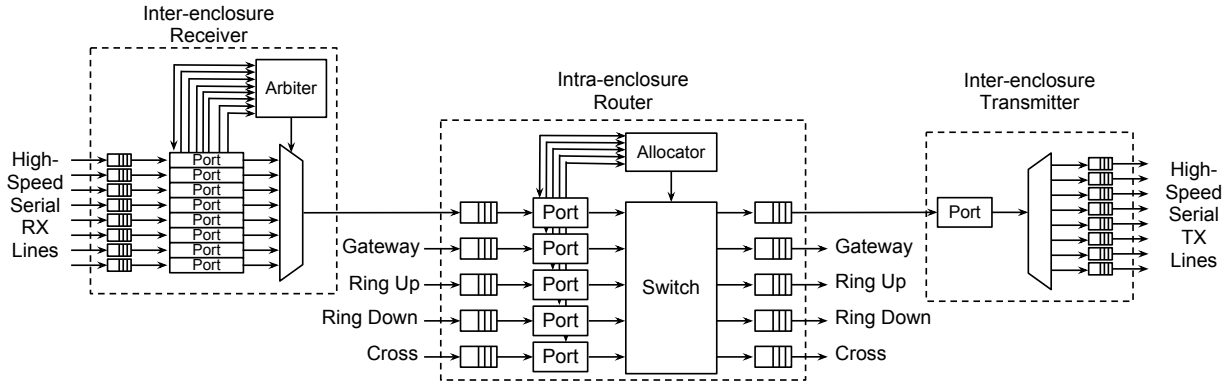


Figure 3. **MGT router block diagram** The MGT router combines three subrouters to form a high-radix router. The inter-enclosure transmit and receive routers provide the connectivity between the high-speed serial lines and the intra-enclosure network. The faster, full-crossbar router routes packets between FPGAs on the same BEE3 board and inter-enclosure routers.

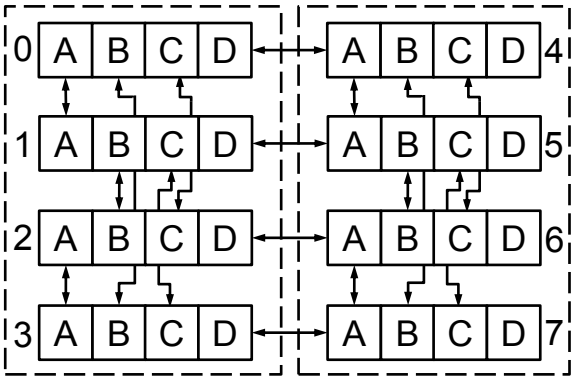


Figure 4. **One-x8-Aurora topology** The *One-x8-Aurora* topology divides the eight BEE3s into two groups of 4. The BEE3s within a group are fully-connected. Each BEE3 also connects to a BEE3 in the other group.

lane width of those channels (i.e., {*NumOfChannels*}-*x*{*LaneWidth*}-*Aurora*). Then, we assign a label "A"-"D" to each FPGA in a BEE3 based on its position on the board.

All of these topologies can be adapted to scale to systems larger than the 8-enclosure, 32-FPGA system we have assembled. We introduce the topologies and their routing protocols here and evaluate them in Section V.

1) One-x8-Aurora: Figure 4 shows the *One-x8-Aurora* topology. Each arrow represents a single, 8-lane Aurora channel. We logically split the BEE3s into two groups of four shown by the dotted boxes. Three of the four channels on a BEE3 connect to the other three BEE3s in its group. To provide connectivity between the groups, the available fourth channel on each BEE3 goes to the available channel on a BEE3 in the other group.

The routing protocol takes the following steps. If an incoming packet is destined for a BEE3 in the same group as the routing BEE3, then it routes that packet to the FPGA whose channel connects to the destination BEE3. However, if the packet is destined for a BEE3 that belongs to the other group, then the router routes the packet to the FPGA whose channel connects to the other group. Now, the packet is at the destination group and the first routing step is applied.

2) Two-x4-Aurora: When each FPGA provides two, 4-lane Aurora channels, there are a total of eight channels per BEE3. This topology connects each of the eight logical channels on a BEE3 to a separate BEE3.

The routing protocol is as simple as the topology design. If an incoming packet is destined for another BEE3, the router routes the packet to the correct FPGA which provides the connection to the destination BEE3. Once the packet reaches the destination BEE3, an intra-enclosure route moves it to its destination FPGA.

3) Four-x2-Aurora: In this topology, each FPGA has four, 2-lane Aurora channels for a total of 16 channels per BEE3. We logically combine the channels of FPGAs A and B to form 8 channels that connect to FPGAs A or B of all other BEE3s. The same is done with FPGAs C and D.

The routing protocol is similar to that of *Two-x4-Aurora*, except that there are now two paths to the destination BEE3: FPGAs A and B have one path and FPGAs C and D have an additional path. This increase in connectivity reduces contention for the inter-enclosure channels.

4) Eight-x1-Aurora: Figure 5 shows the *Eight-x1-Aurora* topology and how it combines two completely-connected networks. The inter-enclosure links provide full connectivity between corresponding FPGAs across the 8 enclosures.

The *Eight-x1-Aurora* topology has the simplest routing protocol because each FPGA has point-to-point connectivity to every other BEE3. An incoming packet

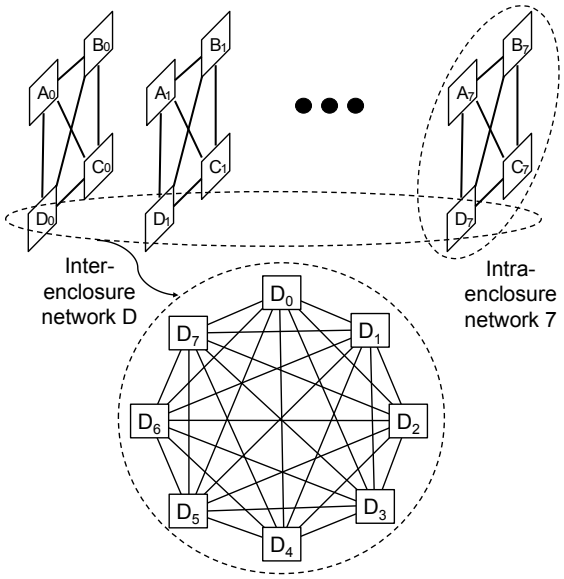


Figure 5. **Eight-x1-Aurora topology** The *Eight-x1-Aurora* topology is a two-level network: a fully-connected intra-enclosure network with low-latency, high-bandwidth channels; and a high-radix inter-enclosure network where each FPGA in a BEE3 communicates with an FPGA in every other enclosure.

can be routed to the destination BEE3 without any intra-enclosure hops. Once the packet reaches the destination BEE3, there is at most one intra-enclosure hop to the destination FPGA.

V. Evaluation

We evaluate the various topologies according to three metrics: network diameter, network throughput, and FPGA resource usage.

In our measurements, we show the results for the Aurora-based topologies using transceivers operating at a 1.95 Gb/s line rate.

A. Network Diameter

Table I depicts the diameters of the various topologies we created. The 10GbE topology has a network diameter up to 7.54 μs due to transmission protocol overhead required for using an Ethernet switch. The latency of the Ethernet switch (450 ns) is minimal compared to the cost of the Ethernet core.

Because the diameter of the *One-x8-Aurora* topology includes two inter-enclosure hops, it has the largest diameter of the Aurora-based topologies at 1.53 μs. By halving the width of the channels in *Two-x4-Aurora*, we have increased connectivity enough to necessitate only a single inter-enclosure hop. The increased connectivity and decreased contention on the inter-enclosure links reduces the diameter of *Two-x4-Aurora* to 845 ns.

As we increase connectivity further in the *Four-x2-Aurora* and *Eight-x1-Aurora* topologies, we would expect a decrease in the diameter. However, the MGT router used to compensate for asymmetry in the inter-enclosure and intra-enclosure links, adds additional delays. The effect is that the diameter of the *Four-x2-Aurora* and *Eight-x1-Aurora* topologies is no smaller than that of *Two-x4-Aurora*.

B. Network Throughput

To evaluate the throughput of our system under our target workloads, we connect a configurable packet generator and checker to the input and output ports on the gateway port of our router. The packet generator can generate random traffic patterns while injecting packets at a rate of 1.86 GB/s at each FPGA. While adding additional packet generators could increase the amount of traffic in the network and more fully utilize the available bandwidth, we desired to evaluate the networks as if they were operating as a data-intensive application engine.

Figure 6 shows the aggregate throughput for 8, 16, and 32 nodes (i.e., FPGAs) for variable packet sizes. Our peak aggregate throughput is 51.4 GB/s. Given that our aggregate injection throughput of 59.6 GB/s, we achieve an efficiency of 86% for our setup.

Initially, increasing packet size improves performance because it amortizes routing, arbitration, and allocation overheads across more bytes. However, for packets larger than 1 kilobyte, the aggregate throughput begins to decrease as the inter-enclosure links become saturated and the queues back up. This congestion causes some FPGAs to pause transmission, reducing bandwidth, and leaving the on-board connections under-utilized.

All of our topologies scale linearly to 32 nodes. Also, the Aurora-based topologies have throughputs that are up to 1.36$\times$ higher than the 10GbE topology.

One of the more interesting observations is the lackluster performance of the *Eight-x1-Aurora* for larger requests. At small packet sizes, the *Eight-x1-Aurora* topology has the highest throughput because there are more inter-enclosure channels with less contention. However, the narrow channels quickly become a bottleneck at request sizes greater than 256 bytes. This is because large packets cause congestion in the inter-enclosure subrouters, which causes under-utilization in the intra-enclosure router. Techniques to alleviate this problem, like virtual channels or more buffering, consume additional resources and logic complexity.

Finally, although the Aurora-based topologies have the same available physical throughput at each transceiver, bonding channels improves aggregate throughput and scales better with request size.

One-x8-Aurora and *Two-x4-Aurora* sustain a higher aggregate throughput for larger scales and larger request sizes.

Table I
TOPOLOGY DIAMETERS THIS TABLE IDENTIFIES THE LATENCY SOURCES THAT CONTRIBUTE TO THE DIAMETER FOR EACH TOPOLOGY. THE LATENCIES OF THE INDIVIDUAL COMPONENTS WERE IDENTIFIED IN SECTION III.

Latency Type	10GbE		One-x8-Aurora		Two-x4-Aurora		Four-x2-Aurora		Eight-x1-Aurora	
	Hops	Latency	Hops	Latency	Hops	Latency	Hops	Latency	Hops	Latency
Full-crossbar router routes	3	192 ns	6	384 ns	4	256 ns	4	256 ns	3	192 ns
MGT router routes	0	0 ns	0	0 ns	0	0 ns	1	88 ns	1	88 ns
Intra-enclosure hops	1	24 ns	3	64 ns	2	48 ns	2	48 ns	1	24 ns
Aurora hops	0	0 ns	2	1082 ns	1	541 ns	1	541 ns	1	541 ns
10GbE hop	1	840-7324 ns	0	0 ns	0	0 ns	0	0 ns	0	0 ns
Total	1.56-7.54 μs		1.53 μs		0.85 μs		0.93 μs		0.85 μs	

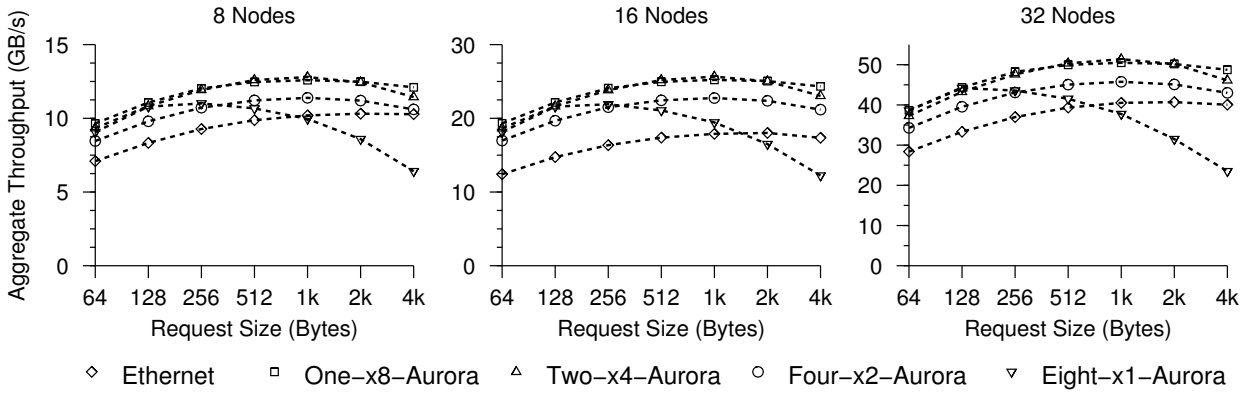


Figure 6. **Aggregate throughput vs. Packet size** This graph shows the maximum aggregate throughput as it scales from 8 to 32 nodes for variable packet sizes for the evaluated network topologies. Please note the vertical axis range for each graph.

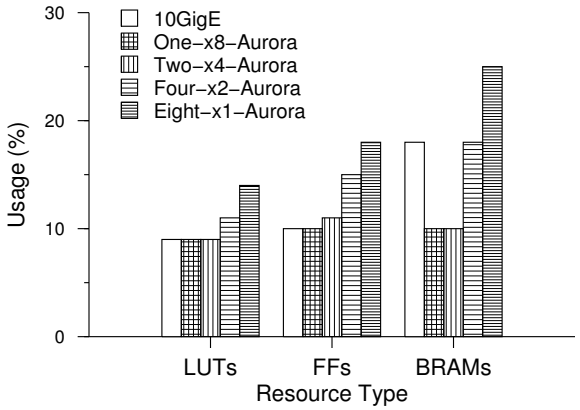
C. FPGA Resource Usage

Striking a balance between network performance and FPGA resource is critical since using FPGA resources for network infrastructure precludes using those resources for other purposes. Simply instantiating the maximum amount of processing elements that will fit in an FPGA will only lead to under-utilization, and therefore degraded performance if the network cannot sustaining enough network throughput.

The graph in Figure 7 depicts the resource usage of the various topologies as a percentage of the total resources for the Xilinx Virtex-5 XC5LX155T FPGA. In addition, the table below the graph provides the raw number of resources.

There are two key observations. First, using many, thin channels uses more resources than fewer, wider channels because a large portion of the resources comes from the routing logic.

Another observation is that the 10GbE core uses similar resources as the two, 4-lane Aurora transceiver core, even though the 10 Gigabit Ethernet core only uses four high-speed serial lanes instead of eight. We attribute this to logic overhead in the 10 Gigabit Ethernet MAC core,



Topology	LUTs	FFs	BRAMS
10GbE	9,563	10,699	39
One-x8-Aurora	8,958	10,099	23
Two-x4-Aurora	9,709	11,526	23
Four-x2-Aurora	11,489	14,658	39
Eight-x1-Aurora	18,003	18,122	55

Figure 7. **Resource Usage** This graph shows the resource usage for the various interconnect technologies and network topologies. FPGA resource usage increases as the number of network channels increases.

which provides additional features (see Section III) than the simpler Aurora cores. The result is that, although we have dedicated the Ethernet switch for inter-enclosure routing, we still consume the same number of resources on the FPGA as a high-radix topology.

VI. Conclusion and Future Work

Data-intensive applications are ubiquitous in science and society. As data scale, it becomes more difficult to analyze them with traditional architectures. Our work aims to understand how a cluster of 32 FPGAs can efficiently solve data-intensive applications. The cluster provides two unique features compared to previous FPGA-based clusters: access to large amounts of memory and a low-latency interconnect.

This paper has evaluated diverse low-latency interconnects for a cluster of 32 FPGAs that combine heterogeneous link types and an optimized topology to support fast, highly concurrent access to large amounts of remote data. The networks we present have aggregate throughputs up to 51.4 GB/s, diameters as low as 845 nanoseconds, and consume less than 12% of the FPGAs' slices.

References

[1] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, ISCA '05, pages 420–431, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Kuen Hung Tsoi and Wayne Luk. Axel: a heterogeneous cluster with FPGAs and GPUs. In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '10, pages 115–124, New York, NY, USA, 2010. ACM.

[3] Rob Baxter, Stephen Booth, Mark Bull, Geoff Cawood, James Perry, Mark Parsons, Alan Simpson, Arthur Trew, Andrew McCormick, Graham Smart, Ronnie Smart, Allan Cantle, Richard Chamberlain, and Gildas Genest. Maxwell - a 64 FPGA supercomputer. In *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems*, AHS '07, pages 287–294, Washington, DC, USA, 2007. IEEE Computer Society.

[4] O. Mencer, Kuen Hung Tsoi, S. Craimer, T. Todman, Wayne Luk, Ming Yee Wong, and Philip Leong. Cube: A 512-FPGA cluster. In *Programmable Logic, 2009. SPL. 5th Southern Conference on*, pages 51 –57, April 2009.

[5] A. Patel, C.A. Madill, M. Saldana, C. Comis, R. Pomes, and P. Chow. A scalable FPGA-based multiprocessor. In *Field-Programmable Custom Computing Machines, 14th Annual IEEE Symposium on*, FCCM '06, pages 111–120, April 2006.

[6] R. Sass, W.V. Kritikos, A.G. Schmidt, S. Beeravolu, and P. Beeraka. Reconfigurable computing cluster (RCC) project: Investigating the feasibility of FPGA-based petascale computing. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 127 –140, april 2007.

[7] A.G. Schmidt, W.V. Kritikos, R.R. Sharma, and R. Sass. AIREN: A novel integration of on-chip and off-chip FPGA networks. In *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, pages 271 –274, april 2009.

[8] Alex Krasnov, Andrew Schultz, John Wawrzynek, Greg Gibeling, and Pierre-Yves Droz. RAMP blue: A message-passing manycore system in FPGAs. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 54–61. IEEE, August 2007.

[9] Xilinx 7 Series FPGAs: http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_O%verview.pdf.

[10] BEE3: http://beecube.com/products/.

[11] Xilinx XAUI Core: http://www.xilinx.com/products/intellectual-property/XAUI.htm.

[12] Xilinx 10GbE MAC: http://www.xilinx.com/products/intellectual-property/DO-DI-10GEMAC.htm.

[13] Aurora: http://www.xilinx.com/products/intellectual-property/aurora8b10b.htm.

[14] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.