High Performance Communication on Reconfigurable Clusters

Jiayi Sheng* Chen Yang† Martin C. Herbordt†
*Falcon Computing Solutions, Inc. †Department of Electrical and Computer Engineering; Boston University

***Abstract*—FPGA clusters with the FPGAs directly linked through their Multi-Gigabit Transceivers (MGT) have a proven advantage over other commodity architectures for communication-bound applications. To date, however, communication infrastructure for such clusters has generally taken one of two approaches: nearest neighbor only, which is fast but has limited utility, and processor-based, which is general, but relatively slow. What is needed is for communication microarchitecture of these systems to be systematically explored, as has been done for HPC clusters and for Networks on Chip (NoC) on both FPGAs and ASICs. Our first contribution is finding that the properties of clusters of tightly coupled FPGAs substantially influence the router design space. We create a candidate router and generalize it so that it is parameterized by routing algorithm, arbitration policy, and virtual channels (VC). We have created a cycle-accurate simulator validated on a four-FPGA system. We evaluate the design space with respect to a number of standard communication patterns and packet sizes. These results enable selection of the appropriate router for any resource budget. We find that the optimality of the router design varies significantly with workloads. We present a framework that helps to determine appropriate parameters based on different applications and generate the HDL design. We observe that for a 512 FPGA cluster, compared with the router configuration with the best average performance, application-aware router selection can lead to substantial improvement in performance or reduction in area.**

***Keywords*-FPGA Cluster Communication; Router Microarchitecture; High-Performance Computing**

I. Introduction

Clusters with FPGAs connected directly through their MGT ports (F-Clusters) [1]–[3] have the advantage over all other commodity configurations in facilitating the collocation of compute, Network Interface Card (NIC), and router on the same device. A further advantage, the exploration of which is a contribution of this work, is that F-Clusters enable the communication infrastructure to be tailored to the workload. The problem we address is that to date communication infrastructure for F-Clusters has generally taken one of two approaches: nearest neighbor, which is fast but has limited utility, and processor-based, which is general, but relatively slow. What is needed is for the communication microarchitecture of these systems to be systematically explored, as has been done for HPC clusters and NoCs (both FPGAs [4], [5] and ASICs). The microarchitecture of communication routers has long been a fundamental concern in computer engineering with many basic principles having reached textbook status 20 years ago [6]. Even so, every new technology, change of scale, or workload, reopens this domain for further examination.

Our approach is to begin with a standard router design that is also the basis of most commercial HPC routers, adjust the design for F-Cluster-specific requirements, and parameterize that design for routing algorithm, arbitration policy, number of VCs, and buffer size. For evaluation workloads, we use a number of standard communication patterns derived from real applications and a variety of packet sizes. To traverse the design space efficiently, we have built a cycle-accurate simulator validated with a four-node F-Cluster.

We have two main sets of results. The first is that, somewhat surprisingly, the properties of F-Clusters substantially influence the router design space. These include such basic quantities as likely area budget, flit size, flow control, minimum appropriate buffer sizes, and indications of favored routing algorithms (near minimal). The second is the results from experiments. These include specifying the useful range of router sizes, removing from consideration small routers appropriate for NoCs, finding the best routing algorithm and arbitration policy for a mixed workload, finding the preferred configurations for different communication patterns, and demonstrating the benefits of application-aware design selection. The latter depends on the selection criteria. We explore several scenarios. In general, a 4%-5% benefit is likely to be achieved. Other cases show area/performance trade-offs resulting in 8% performance improvement or a 30% reduction in area with no loss of performance.

Among the other contributions are the following.

- A complete network stack including parameterized router, routing algorithms, arbitration policies, flow control, and link and physical connections,
- An F-Cluster-specific router microarchitecture, modified from the classical Dally router, and
- A framework for exploring design-space and selecting application-aware optimal router configurations.

II. Background

A. Related Work

To our best of knowledge, we are the first to implement wormhole VC-based routing on a multi-FPGA network with MGT interconnections. There have been many innovative studies on VC-based router designs for FPGA NoCs, but as we discuss immediately below and in the results section, there are great differences in design constraints that lead to different design decisions being preferred for F-Clusters.

B. Review: Hardware Support for General Communication

We begin with a common pipelined virtual-channel(VC)-based router [6]. VC-based routing has two benefits. It overcomes the blocking problems of traditional wormhole flow control by allowing other packets to use the channel bandwidth that would

This work was supported in part by the NSF through awards CNS-1405695 and CCF-1618303/7960; by the NIH through award 1R41GM128533; by a grant from Red Hat; and by Altera (now Intel) through donated FPGAs, tools, and IP.

otherwise be left idle when a packet blocks. And it is useful in solving deadlock by breaking channel dependency cycles.

C. F-Cluster Router Constraints

Properties of F-Clusters and their use provide a significant constraining on the F-Cluster router design space.

Network is direct. A network is commonly referred to as *direct* if the routers are associated with the nodes. Since a central reason for using F-Clusters is collocation of application and communication logic, this choice is obvious.

Topology is a 3D Torus. With direct networks for HPC systems, it makes more sense to use a D dimensional mesh or torus. For systems with hundreds of nodes, a $D = 3$ torus is a good fit.

General routing should be wormhole VC-based. These routers have long been preferred in commercial HPC so clearly we should do that with F-Clusters if possible.

A wide (128-bit) datapath is needed. Assuming a maximum operating frequency above 140MHz, this is the minimum size that can sink a flit per cycle from the MGTs.

There is one router per FPGA. This follows other collocated designs such as BlueGene [7].

Credit-based flow control. Since in F-Clusters there is no global clock, and the MGT links do not provide specific backpressure signals, we use credit-based flow control.

TABLE I
COMPARISON OF CONSTRAINTS BETWEEN F-CLUSTERS AND F-NOCS

	FPGA cluster	FPGA NoC	Impact
Topology	3D-torus	2D-mesh or torus	More ports, more complicated switch
Inter-node latency	50-100 CCs [8]	1-3 CCs	# of pipeline stages not important
Phit size	>128-bit	<32-bit	Flit size vs. Phit size
# of routers/chip	1	>10	More room for VCs and switch complexity
global clocking	No	yes	Credit-based flow control

We continue our constraint of the design space showing how that of F-clusters differs substantially from that of FPGA NoCs (see Table I). We note three key consequences: (i) while pipelined router design is not suitable for FPGA NoCs [4], it is for F-clusters; (ii) in F-Clusters the phit is bigger (128-bit) than the routing and control information (flit); and (iii) the bigger area budget on the FPGA cluster allows more complexity.

III. DESIGN

We have two goals: finding candidate router designs that meet the F-Cluster constraints described in the previous section and for those designs to be parameterizable. We begin with the textbook router (see Figure 1).

1. Two sets of asynchronous FIFOs are used for crossing the two clock domains (MGT and router).

2. Multiple ports for local injection and ejection. In F-Clusters, packets can be directly injected/ejected from/to user logic.

3. Connection between VCs and switch. In F-Clusters, with much larger phits and resource budget, the added cost of full switching is modest and so we do not multiplex VCs.

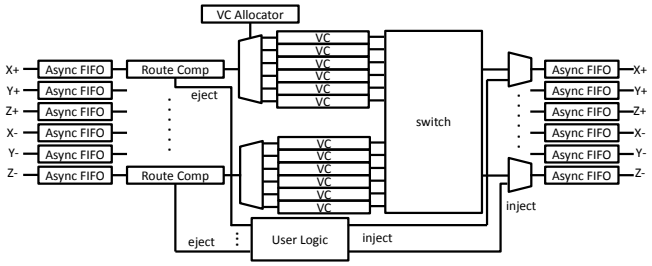

Fig. 1. Proposed router microarchitecture

4. Switch implementation. It is possible to have a large number of VCs and stay within the upper bound area budget, but only if the switching logic is optimized, in this case from a $M \times N$ to N crossbar where M is VCs/port and N is the number of ports. To address this issue, we use a reduction-tree-based switch.

We now describe how we address three standard design decisions: deadlock avoidance, packet format, and flow control.

Deadlock Avoidance. We use two standard methods, datelines [6] and turn prohibition. When implementing datelines, we set a VC class bit in the packet header, which is toggled when the packet crosses the dateline or changes dimension. A well-known problem with the dateline method is that the utilization of the two classes is imbalanced. We address this problem by overlapping the buffer spaces for the two classes [6]. This works as long as there is an escape route, i.e., at least one exclusive buffer, per class. Another category of deadlock is the dependency loop across dimensions. To prevent this we forbid six particular turns. The routing algorithms presented in the next section all either support these methods directly or have been modified to do so.

Packet Format. Our network supports arbitrary packet sizes. To achieve this, we support five types of flits: head, body, tail, single, and credit. A packet generally starts with a head flit, followed by several body flits, and ending with a tail flit. If the packet has only a single flit, its flit type is single. The credit flit is used to transfer credit information from downstream nodes to upstream nodes. All flits have a type field and a payload field. The payload field in the credit flit contains the credit information of a downstream node; the payload field in other flit types contains the payload data. Single and head flits also include the destination, VC class, and priority. The priority field is used when multiple packets are competing for the same output port and depends on the routing algorithm and arbitration policy. Its content might be packet age, distance from the destination, or both.

Flow Control. When the buffers on a downstream node are full, it must generate a backpressure signal to the upstream node to notify it to stop sending data. We use credit-based flow control to manage the inter-node backpressure. This mechanism differs substantially from that in NoC. MGT link IP does not generally provide support for backpressure so its design can be optimized. Since the credit flits must be multiplexed with the normal data flow, a parameter is the frequency at which backpressure credit flits are sent. In NoCs, credit flits are usually piggybacked on data flits [6]. Because of the short inter-node latency on NoCs, this solution can ensure the upstream node is always aware of the most recent credit information of neighboring nodes. But the overhead can be significant, with up to half of the link bandwidth consumed

with credits. Since the inter-node latency in F-Clusters is 50-100 cycles, it is impossible for the upstream nodes to be aware of the real-time credit information on neighboring nodes. Therefore, in our design, we sent one credit flit per 100 cycles. This increases the credit threshold by that amount and requires somewhat larger buffer space, but drastically improves throughput.

IV. Implementation

A. Overview

In this project we seek to explore the space of F-Cluster communication. We find there is no universally optimal routing algorithm and switch arbitration policy. Because of the large number of alternatives (including those for future study), we have created a cycle-accurate simulator to evaluate different combinations of these algorithms and policies. Based on this simulator, we propose a framework that can help users find the best router configuration for their applications. We define an interface standard for new router components, such as for routing or switch arbitration. Users can easily include their own components into our framework.

The space of communication mechanisms is vast. In this paper, we restrict ourselves to unicast workloads (see [9], [10] for work on collectives), although we have created a version of the router that supports collectives as well. We select five routing algorithms, four oblivious and one adaptive, and three arbitration policies for a total of 15 combinations. We concentrate on the simple mechanism, first because they are an obvious starting point, but also because they generally result in good performance (see, e.g., Leighton on the performance hit from collisions due to added turns [11]).

B. Routing Algorithms

The first four algorithms are oblivious and can be implemented either with a small amount of logic or with routing tables. The adaptive algorithm can only be implemented dynamically using on-chip FPGA logic.

1. Dimensional Ordering Routing (DOR).

2. Randomized, Oblivious, Multi-phase, Minimal Routing (ROMM) [12]. ROMM randomly selects p intermediate nodes in the minimum submesh constrained by source and destination.

3. Orthogonal One-turn Routing (O1TURN) [13]. O1TURN randomly selects one of all the possible DOR routes.

4. Randomized Load Balance Routing (RLB) [14]. In each dimension, packets do not necessarily follow the minimum path.

5. Credit Count Adaptive Routing Algorithm (CCAR) [15]. In selecting one adaptive algorithm to be representative, we choose one that looks at congestion information in neighboring nodes, is simple, and matches our microarchitecture. CCAR gathers credit information from neighboring nodes using the mechanism already in place for flow control.

C. Switch Arbitration Policies

Whenever two head flits collide within a switch, an arbitration policy decides which packet should proceed.

1. Farthest First (FF).

2. Oldest First (OF).

3. Mixed. The default arbitration is FF. But if a packet's age is above a threshold, it is granted higher priority than all the packets below the threshold, no matter how close it is from its destination.

V. Experimental Setup and Methods

A. Test Fixture

We have implemented, tested, and verified our designs on a four FPGA system. Each FPGA card hosts an Altera Stratix V 5GSMD8 chip and supports six links that use the FPGA's Multi-Gigabit Transceivers (MGT). Each MGT link has a bandwidth of 40 Gbps and latency of 175 ns. Results are for an 8^3 torus. For larger systems, we concentrate here on a 512 FPGA 8^3 cluster, we use our cycle-accurate simulator, which we now describe.

B. Cycle-Accurate Simulator

Given the challenges of HDL coding, an efficient cycle-accurate simulator is essential for exploring a large design space. Ours is implemented in C++; every hardware module in the RTL model has a corresponding class in the simulator. These classes are organized in the same hierarchical structure as the RTL model. To give the cycle-accurate simulator good extensibility with new algorithms and policies, we define an interface standard for all hardware modules. In particular, this allows us to use a producer-consumer model with every module being both a producer and consumer. This simulator has been fully validated with respect to the four node test fixture. This is substantially simplified by our simulator design: it is a mirror copy of the RTL code, which 100% matches the behavior in the RTL simulation.

C. Workloads

We evaluate the design using two types of workloads: batch and continuous. In batch mode, we inject a fixed number of packets and measure the batch latency, i.e., the number of cycles elapsed from the injection of the first flit to the arrival of the last flit. This is representative of most HPC applications. In continuous mode, each node injects packets with a certain probability; this determines the applied load. Since latency is a function of load, the best measure here is the maximum load [16].

We conduct experiments using five different communication patterns all derived from real applications and representative of workloads commonly used in communication studies. (1) 3-Hop Nearest Neighbor (3H) is used in many stencil applications, (2) Cube Nearest Neighbor (CUBE) is used in physical simulations such as Molecular Dynamics [17], [18], (3) Bit Complement (Bit-Comp) is used in many DSP applications, (4) Transpose (Trans) is used in many applications, e.g., the FFT [19], [20], and (5) Random, which is used generally for predictable routing and specifically in NWChem [21].

VI. Experiments and Evaluation

In the first part of this paper, the emphasis is on constraints derived from properties of F-Clusters and their implications on the F-Cluster router design space. In this section, we evaluate that remaining design space with respect to both performance and area cost. We address three sets of questions.

1. What is the performance for each major cost point on the Pareto Optimal frontier? Is there an inflection on this frontier (i.e., knee in the curve), which would indicate a likely preferred design? These results also answer a number of secondary questions, including: How big is the largest router before the benefit becomes marginal? and How do very small routers, as are used for NOCs, compare with the F-Cluster routers proposed here?
2. How do the different design alternatives behave with respect to the different communication patterns?
3. How valuable is configurability in the space of F-Cluster routers? That is, given a single best ("global optimal") design–or a set of best designs, one per area cost–how do these best designs compare, in terms of performance and cost, with designs selected based on communication pattern?

A. Performance Versus Area Cost

TABLE II
RESOURCE CONSUMPTION WRT NUMBER OF VCS PER PORT

VC#	2	3	4	5	6	7	8	9	Hoplite
ALMs	7.5k	9.0k	10k	12k	14k	15k	17k	18k	0.5k
area%	3%	4%	5%	5%	6%	6%	7%	7%	0.2%

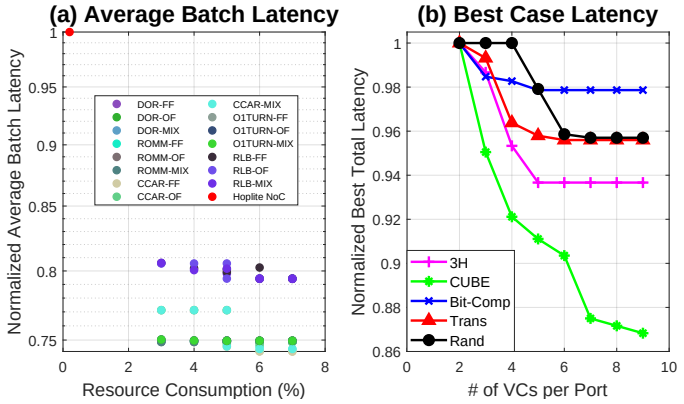

Fig. 2. Batch mode performance: (a) Geometric mean of batch latency across 5 different communication patterns with respect to resource usage. Latency value is normalized to the longest latency among the 16 policies (including Hoplite NoC) for each pattern. (b) The best (smallest) latency among 16 policies under different workload with regarding to number of virtual channels. Latency value is normalized to the best number when VC = 2.

With constant buffer size per VC, the router area is dominated by the number of VCs. Table II confirms this by showing router area almost linear in the number of VCs. To find the preferred number of VCs, we turn to the relationship between area and performance for different workloads (Figure 2(a)). We also select the best performing configuration for each router size (the Pareto-optimal frontier) with respect to different workloads (Figure 2(b)). From these graphs we determine the preferred router size for each application: we select the number of VCs after which the performance benefit is negligible.
What is the maximum useful size of an F-Cluster router? This varies by application from 5 VCs (5% area) to 7 VCs (6% area). In selecting the VC number we do not simply use continued improvement in performance, but rather choose the distinct knee that appears in all of the curves. For the applications where more than 5 VCs yield improved performance, the benefit of the large router is about 5%.

TABLE III
THROUGHPUT COMPARISON OF APPLICATION-AWARE AND GLOBALLY OPTIMAL IN PERCENT IMPROVEMENT OF APPLICATION-AWARE

VC#	Optimal Conf	3H	CUBE	Bit	Tran	Rand	Avg
2	ROMM-O	0	2.07	12.2	2.37	4.13	4.06
3	ROMM-O	0	0.84	8.80	8.75	4.13	4.44
4	ROMM-F	0	2.34	8.51	7.87	4.03	4.50
5	CCAR-O	0	1.98	9.10	2.42	5.95	3.84

How do NoC-type routers perform? The Hoplite-type router takes $< 1\%$ resource consumption in contrast with the 3% for the smallest F-Cluster router (see Table III). However, there is a large performance cost, as shown in Figure 2(a). The reasons for the reduced performance are exactly what motivate the F-Cluster router design, including long link latency, which drastically increases the cost of misrouting.

B. Performance of Algorithm-Policy Combinations

We examine the impact of routing algorithm and arbitration policy on performance. We set the number of VCs to 5 because of the small benefit from using more than 5 VCs. Figures 3 and 4 show performance for the different workloads.

We note the wide variation in performance with respect to routing pattern with six different combinations being preferred for at least one mode/pattern. While there are many interesting observations on which combination is preferred, we limit these to the following: the variation indicates that application-aware configuration may be beneficial. This is evaluated next.

TABLE IV
BATCH LATENCY COMPARISON OF APPLICATION-AWARE AND GLOBALLY OPTIMAL IN PERCENT IMPROVEMENT OF APPLICATION-AWARE

VC#	Optimal Conf	3H	CUBE	Bit	Tran	Rand	Avg
2	DOR-O	3.44	0.61	15.4	1.71	0	4.08
3	ROMM-O	2.18	0	13.3	8.44	0	4.65
4	CCAR-F	13.9	2.55	8.08	5.69	0.06	5.95
5	CCAR-O	0	1.84	6.30	1.15	1.15	2.07

TABLE V
COMPARISON IN PERCENT OF LARGE APPLICATION-AWARE (VC# = 5) AND COMPACT GOBALLY OPTIMAL (VC# = 2) CONFIGURATIONS

	3H-NN	CUBE-NN	Bit-com	Tran	Random	Avg
Latency	7.83	5.20	14.55	8.84	4.32	7.43
Throughput	11.24	6.53	10.04	7.59	8.83	8.69

TABLE VI
COMPARISON IN PERCENT OF SMALL (VC# = 2) APPLICATION-AWARE AND LARGE GLOBALLY OPTIMAL (VC# = 5) CONFIGURATIONS

	3H-NN	CUBE-NN	Bit-com	Tran	Random	Avg
Latency	-5.07	-3.27	5.89	6.61	-4.14	0
Throughput	-10.1	-3.56	8.47	8.29	-4.18	-0.49

C. Globally Optimal Versus Application-Aware

Given that many applications are dominated by a single communication pattern, it follows that application-aware configuration could be beneficial. Here we evaluate how much. The alternative to application-aware is to select a single router, which we refer to as *Globally Optimal*. This decision is complicated, however, because we do not know *a priori* the area budget of the designer. We examine three possible scenarios.
1. The designer knows the budget for both cases and the budget is the same for both. While this might not be realistic, it leads

to the most fair comparison. Tables III and IV show an average performance improvement from 0% to 9% with size-dependent averages ranging from 3% to 5% (for different number of VCs). Overall likely benefit is little over 4%.

2. The cluster provider selects a single router where the area budget is as small as viable, i.e., 3% of chip area. The application designer, however, has as a priority to maximize performance and so is willing to use more area, say, 60% more. Table V indicates an average performance improvement of around 8%.

3. The cluster provider selects a single router where the area budget is generous, i.e. 5% of chip area. The application designer, however, has a priority to minimize area. Table VI shows an area reduction of 38% with, on average, no loss of performance.

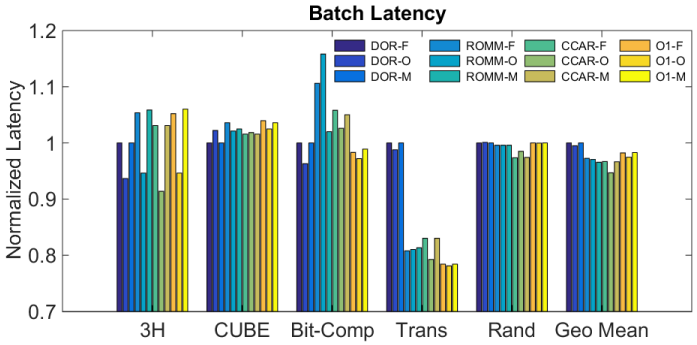

Fig. 3. Batch latency for different combinations of routing algorithm and arbitration policy (smaller is better)

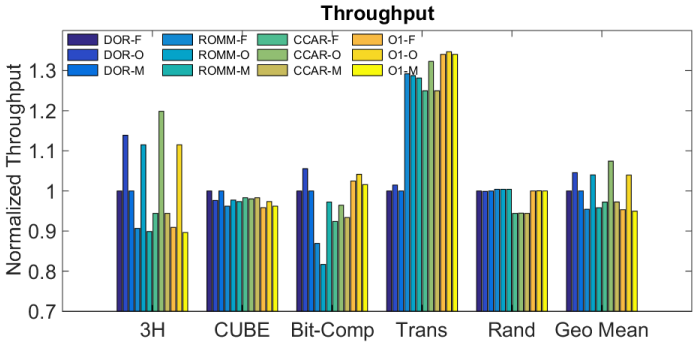

Fig. 4. Throughput for different combinations of routing algorithm and arbitration policy (larger is better)

VII. Discussion and Future Work

In this paper, we present what we believe to be the first study of F-Cluster communication for general HPC workloads. We begin by examining F-Cluster properties and deriving a series of design constraints dealing with topology, routing mode, datapath size, flow control, and resource budget. We have tailored the textbook VC wormhole router to account for these constraints making design decisions about deadlock avoidance, packet formats, and switch implementation. The design is parameterized so that it is easy to vary the remaining design choices such as number of VCs, buffer sizes, routing algorithm, and arbitration policy. Among the experimental results are the following: the likely preferred number of VCs, the best "universal" routing algorithm and arbitration policy, the preferred configurations for different communication patterns, and the benefits of application-aware design selection under various scenarios.

The overall significance is, first, that we have shown how communication on F-Clusters needs to be done and, in particular, how different this domain is from the more familiar NoC. Second we have demonstrated significant benefit of application-aware design. While these results are not the huge numbers we are used to seeing in application-specific acceleration, they are general and of similar order to that of leading work in communication. Finally, all designs and simulators are publicly available on Github (https://github.com/caadlab/F-Cluster_Router_Simulator_v0.1).

References

[1] R. Sass, et al., "Reconfigurable computing cluster (RCC) project: Investigating the feasibility of FPGA-based petascale computing," in *Proc. Field Programmable Custom Computing Machines*, 2007, pp. 127–138.

[2] A. George, M. Herbordt, H. Lam, A. Lawande, J. Sheng, and C. Yang, "Novo-G#: A Community Resource for Exploring Large-Scale Reconfigurable Computing Through Direct and Programmable Interconnects," in *IEEE High Perf. Extreme Computing Conf.*, 2016.

[3] J. Sheng, B. Humphries, H. Zhang, and M. Herbordt, "Design of 3D FFTs with FPGA Clusters," in *IEEE High Perf. Extreme Computing Conf.*, 2014.

[4] M. Papamichael and J. Hoe, "CONNECT: Re-examining Conventional Wisdom for Designing NoCs in the Context of FPGAs," in *Proc ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, 2012, pp. 37–46.

[5] N. Kapre and J. Gray, "Hoplite: Building Austere Overlay NoCs for FPGAs," in *Proc. FPL*, 2015, pp. 1–8.

[6] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[7] D. Chen, et al., "The IBM Blue Gene/Q Interconnection Network and Message Unit," in *Proc. Supercomputing*, 2011.

[8] J. Sheng, C. Yang, and M. Herbordt, "Towards Low-Latency Communication on FPGA Clusters with 3D FFT Case Study," in *Proc. Highly Efficient and Reconfigurable Technologies*, 2015.

[9] J. Sheng, Q. Xiong, C. Yang, and M. Herbordt, "Collective Communication on FPGA Clusters with Static Scheduling," *Computer Architecture News*, vol. 44, no. 4, 2016.

[10] J. Stern, Q. Xiong, J. Sheng, A. Skjellum, and M. Herbordt, "Accelerating MPI_Reduce with FPGAs in the Network," in *Proc Workshop on Exascale MPI*, 2017.

[11] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufman, 1992.

[12] T. Nesson and S. Johnsson, "ROMM routing on mesh and torus networks," in *Proc. Symp. on Parallel Algorithms and Architectures*, 1995, pp. 275–286.

[13] D. Seo, A. Ali, W. Lim, N. Rafique, and M. Thottethodi, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," in *Int. Symp. on Computer Architecture*, 2005, pp. 432–443.

[14] A. Singh, W. Dally, B. Towles, and A. Gupta, "Locality-preserving randomized oblivious routing on torus networks," in *Proc. ACM Symp. on Parallel Algorithms and Architectures*, 2002, pp. 9–13.

[15] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the Design Automation Conference*, 2005, pp. 559–564.

[16] M. Herbordt, K. Olin, and H. Le, "Design trade-offs of low-cost multicomputer networks," in *Proc. 7th Symp. on the Frontiers of Massively Parallel Computation*, 1999, pp. 25–34.

[17] M. Chiu, M. Herbordt, and M. Langhammer, "Performance potential of molecular dynamics simulations on high performance reconfigurable computing systems," in *Proceedings High Performance Reconfigurable Technology and Applications*, 2008.

[18] M. Chiu and M. Herbordt, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Trans. Reconfigurable Tech. and Sys.*, vol. 3, no. 4, pp. 1–37, 2010.

[19] J. Sheng, C. Yang, A. Caulfield, M. Papamichael, and M. Herbordt, "HPC on FPGA Clouds: 3D FFTs and Implications for Molecular Dynamics," in *Proc. IEEE Conf. on Field Programmable Logic and Applications*, 2017.

[20] A. Sanaullah and M. Herbordt, "FPGA HPC using OpenCL: Case Study in 3D FFT," in *Proc. Int. Symp. on Highly Efficient Accelerators and Reconfigurable Technologies*, 2018.

[21] M. Valiev, E. Bylaska, N. Govind, K. Kowalski, T. Straatsma, H. V. Dam, D. Wang, J. Nieplocha, E. Apra, T. Windus *et al.*, "NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations," *Computer Physics Communications*, vol. 181, no. 9, 2010.