Exploring the Impact of Switch Arity on Butterfly Fat Tree FPGA NoCs

Ian Lang
University of Waterloo
Waterloo, ON, Canada
Email: ielmorla@uwaterloo.ca

Ziqiang Huang
University of Waterloo
Waterloo, ON, Canada
Email: ziqiang.huang@uwaterloo.ca

Nachiket Kapre
University of Waterloo
Waterloo, ON, Canada
Email: nachiket@uwaterloo.ca

***Abstract*—Overlay Networks-on-Chip (NoCs) for FPGAs based on the Butterfly-Fat Tree (BFT) topology with lightweight flow control deliver low LUT costs and features such as in-order delivery and livelock freedom. BFT NoCs make it possible to configure network bandwidth to match application requirements, by choosing switch types with different numbers of ports (arity) for the layers of the tree hierarchy. We increase the design space of BFT NoC configurations available to designers by constructing networks with larger arity-4 switches, in addition to the arity-2 switches explored by previous works. When synthesized for the Xilinx UltraScale+ VU9P FPGA, our proposed BFT NoCs consume 38-45% fewer LUTs and 33-50% smaller wiring lengths than arity-2 BFT NoCs with the same Rent parameter, in exchange for a reduction in maximum clock frequency in up to 25%. We simulate the operation of our proposed NoCs when routing various real-world workloads with 64 network clients, and show that they consistently achieve better Throughput / LUT cost ratios, when compared to arity-2 BFT NoCs with the same Rent parameter, with improvements of 15 to 120% depending on the benchmark and NoC topology.**

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are being widely adopted by industry: examples include Microsoft Catapult [1], the Amazon EC2 F1 Instances [2], the Baidu FPGA Cloud Server [3] and the Huawei FPGA Accelerated Cloud Server [4]. FPGAs have also become more accessible to software developers, due to High Level Synthesis tools such as Vivado HLS [5] and Intel a++ [6].

This context of increased popularity and accessibility, as well as the growing complexity of applications implemented in FPGAs (in e.g. machine learning, networking), has led to the rise of modular workflows and data movement paradigms, where intellectual property (IP) cores are instantiated and connected in the reconfigurable fabric to fulfill application requirements. These trends, in turn, have created an important role for overlay networks-on-chip (NoC), which transport data between modules in a scalable way, and are assembled from the same reconfigurable components as the rest of the application deployed in a FPGA. An overlay NoC may be used either as the main substrate for communications, or as a complement to an hard (ASIC) NoC present in the chip [7], providing tighter connectivity.

This paper builds upon previous works that explored the Butterfly-Fat Tree (BFT) topology as an option for overlay NoC [8], [9]. BFT are hierarchical structures for building networks, where the clients (known as Processing Elements - PE) are present at the bottom of the hierarchy, and messages sent from a PE climb layers of switches until an appropriate level, and then turn and descend to reach their destination. An example packet trajectory is shown in Figure 1. These previous works focused on BFT NoCs composed of switches with two ports facing downhill in the tree hierarchy (arity-2 switches): Figure 1(a) shows one such NoC, with 64 clients at the bottom.

This paper explores arity-4 switches as an alternative to construct BFT NoC that require fewer resources (in terms of LUTs, Flip-Flops and wiring) than their counterparts based on arity-2 switches. Its contributions include:

- Register-transer level (RTL) implementation of four BFT NoCs based on arity-4 switches, with different bandwidths as expressed by the Rent parameter [10]. One such BFT NoC is depicted in Figure 1(b).
- Characterization of the performance of new BFT NoCs when routing various realistic workloads.
- Comparison with BFT NoCs based on arity-2 switches in terms of Throughput, Worst-Case Packet Latency and resource costs.
- A repository containing our RTL implementations of the BFT NoCs and associated code for simulation and synthesis, available at https://git.uwaterloo.ca/watcag-public/bft-flow-arity4.

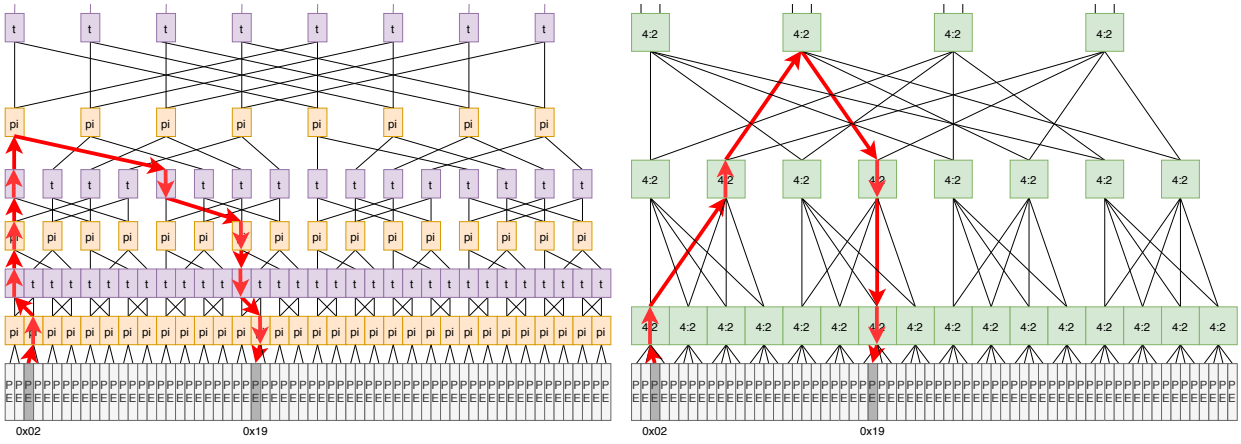
II. Background

First introduced in 1985 [11], Butterfly Fat Trees (BFT) have been employed in the design of communication networks for data centers [12] and multiprocessors [13], among other applications.

FPGA NoCs with BFT topologies are explored in previous works [8] [9]. BFT variations offering different amounts of bandwidth are constructed by configuring each layer of switches with either one uphill-facing port (t switches) or two uphill-facing ports (pi switches).

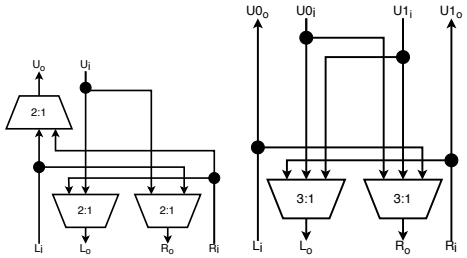
This notion of bandwidth is expressed by the Rent parameter p, which stems from a recursive model of the number of connections that cross the network bisection as an exponential function $C \times N^p$ [10]. In this model, N is the number of leaf nodes (network clients) and $0 \leq p \leq 1$ is the Rent parameter.

2576-2621/20/$31.00 ©2020 IEEE
DOI 10.1109/FCCM48280.2020.00019



(a) Arity-2 (b) Arity-4

Fig. 1. Two BFT NoC with a Rent parameter of 0.5 (equivalent to a mesh), one based on arity-2 switches (a), the other based on arity-4 switches (b). The red highlight shows the route taken by a packet from the network address 0x02 to the network address 0x19.



(a) t Switch (b) pi Switch

Fig. 2. Internal structure of the arity-2 t and pi switches.

A group of four BFT variants, which we refer to as BFT-A2, was studied by the previous works [8] [9]:

- **BFT0 (Arity = 2)** composed only of layers of t switches. It has a Rent parameter $p = 0$, the same as a tree network.
- **BFT1 (Arity = 2)** composed of alternating layers of t and pi switches (pictured in Figure 1(a)). It has a Rent parameter $p = 0.5$, the same as a regular mesh.
- **BFT2 (Arity = 2)** composed of a repeating pattern of two layers of pi switches followed one layer of t switches. It has a Rent parameter $p = 0.67$.
- **BFT3 (Arity = 2)** composed only of layers of pi switches. It has a Rent parameter $p = 1$, the same as a crossbar.

Figure 2 shows the internal structure for the switches proposed by the more recent work to study the BFT-A2 [9]. Inside the pi switch (Figure 2 (b)), incoming packets from a specific downward-facing input are directed to a specific upward-facing output: this ensures that a sequence of packets sent from a PE to another is received in-order. The same work also implemented a lightweight flow-control mechanism for the switches: contention for the output of the multiplexers (muxes) shown in Figure 2 is managed by round-robin arbiters. Packets that lose the contention for an output are stored in shadow-registers, and backpressure signals are propagated to signal when shadow registers are occupied.

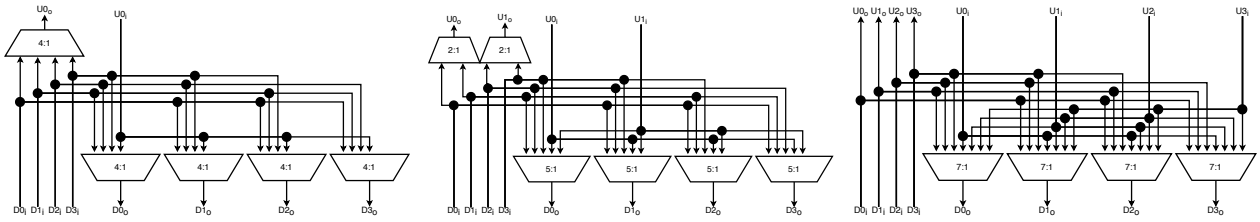
III. BFT NoC based on arity-4 switches

As an overlay NoC is assembled from the same reconfigurable fabric as the rest of the application deployed on an FPGA, it must also factor into the budget of reconfigurable elements used to implement the complete application. In this context, one reason to consider arity-4 switches is that they may harness the wide muxes present in some modern devices, such as the F7 muxes featured in Xilinx 7 Series FPGAs [14], which make it possible to create 7-input functions by switching between the outputs of two 6-LUTS.

Should these wide muxes contribute to limit the resource costs of arity-4 switches when compared to arity-2 switches, the overall resource cost of arity-4 BFT NoCs would be smaller, as fewer arity-4 switches are required to construct a NoC with a given Rent parameter.

Figure 3 shows the internal structure we define for the three arity-4 switches: 4:1 (four downhill-facing duplex links, one uphill-facing link), 4:2 and 4:4. Similarly to the strategy adopted in the more recent work to study the BFT-A2 [9], packets entering through a certain downward-facing input port can only exit through a certain uphill-facing output port (to guarantee in-order delivery), or through one of the downward-facing output ports, in case of a turn.

The routing of packets in the arity-4 BFT NoC network follows the same basic rules as in the arity-2 BFT NoC. The major difference is that, instead of determining which of the two downhill ports to send a descending packet through by reading one address bit, the switches decide between four ports by reading a pair of address bits. The lightweight flow-control mechanism used for the arity-2 switches is also present in the arity-4 switches.



(a) 4:1 Switch (b) 4:2 Switch (c) 4:4 Switch

Fig. 3. Internal structure of the three arity-4 switches explored in this work.

We define a group BFT-A4 of four BFT NoC based on arity-4 switches, with the same Rent parameter values as those in the BFT-A2 group:

- **BFT0 (Arity = 4)** composed only of layers of 4:1 switches (Rent parameter $p = 0$).
- **BFT1 (Arity = 4)** composed only of layers of 4:2 switches, pictured in Figure 1(b) ($p = 0.5$).
- **BFT2 (Arity = 4)** composed of a repeating pattern of one layer of 4:4 switches followed by two layers of 4:2 switches ($p = 0.67$).
- **BFT3 (Arity = 4)** composed only of layers 4:4 switches ($p = 1$).

For a given number of PEs, each BFT NoC from the BFT-A4 group requires fewer switches and muxes than its BFT-A2 equivalent. For instance, the arity-2 BFT1 depicted in Figure 1(a) requires 56 pi switches, each with two internal muxes as seen in Figure 2, and 56 t switches, each with three internal muxes, for a total of 112 switches and 280 muxes. The arity-4 BFT1 depicted in Figure 1(b), on the other hand, requires 28 4:2 switches, each with six internal muxes as seen in Figure 3, or 168 muxes overall.

IV. Methodology

In order to perform experiments with our proposed BFT NoC, we first produce Verilog RTL implementations of the arity-4 switches and BFT-A4 NoC, instantiating F7 mux [14] components to implement the wide muxes found inside the 4:2 and 4:4 switches. A purely behavioural implementation is also provided to support simulation, and synthesis for devices that don't support F7 muxes.

We then employ Verilator [15] to perform cycle-accurate simulations of the BFT-A2 and BFT-A4 NoCs as they route various benchmarks. Each simulation reports the number of packets transported, the total number of clock cycles required to complete the workload, and the worst-case latency suffered by a workload packet between being sent and being received. The benchmarks are adapted from two sources:

- Benchmarks in graph analytics adapted from [16]: wiki, stanford, soc, roadnet, human, google, amazon.
- Benchmarks in sparse matrix-vector multiplication (SpVM) adapted from [17]: ram2k, hamm, dac, bomhof 1, bomhof 2, bomhof 3, add20.

Graph analytics and SpVM are the application domains chosen for the benchmarks due to their suitability for FPGA acceleration. Each workload consists of a list for each PE, detailing the sequence of single-flit packets that the PE should send, and the address of the PE each message should be destined to. Each encompasses between 10K and 1M total messages exchanged.

TABLE I
Synthesis results for the BFT-A2 and BFT-A4 NOCs (data width of 32 bits, 64 PE).

BFT NoC	LUTs	FFs	Mux F7	Freq. (MHz)	Wire Len.
BFT0 - Arity=2 ([9])	22K	21K	0	490	84
BFT1 - Arity=2 ([9])	39K	37K	0	489	192
BFT2 - Arity=2 ([9])	49K	47K	0	535	288
BFT3 - Arity=2 ([9])	66K	63K	0	523	448
BFT0 - Arity=4 (this work)	12K	14K	0	483	56
BFT1 - Arity=4 (this work)	22K	19K	5K	438	96
BFT2 - Arity=4 (this work)	31K	27K	6K	402	160
BFT3 - Arity=4 (this work)	36K	33K	8K	401	224

We also perform synthesis, placement and routing of the NoCs with Vivado 2019.1 [18], targeting the Xilinx Ultra-Scale+ VU9P FPGA. Table I shows the results in terms of resource costs (LUTs, FFs, and F7 muxes) and maximum achievable frequency. It also includes estimates of the relative wiring length required by each design, supposing an H-tree layout.

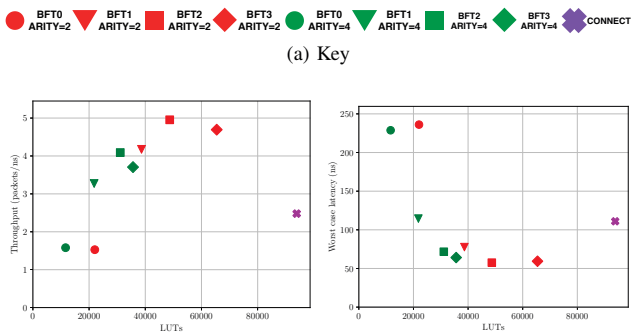
As expected, the BFT-A4 NoCs are less expensive in terms of resources than their BFT-A2 equivalents: 38-45% smaller LUT costs and 33-50% smaller wiring lengths, depending on the topology. However, they also achieve lower maximum frequencies (up to 25% lower frequencies), due to the higher complexity of each switch (e.g. in terms of arbitration logic).

By combining the operating frequency provided by the synthesis with the results of the cycle-accurate simulations, we determine the Throughput and Worst-Case Packet Latency (WCPL) achieved by each NoC when routing each benchmark.

V. Evaluation

Figure 4 shows the LUT cost of each BFT NoC on the horizontal axis, and the Throughput (packets / ns) and WCPL (ns) achieved when routing the soc benchmark on the vertical axis. These plots are representative of the results observed for most benchmarks. For comparison, the data for a contemporary NoC, the CONNECT [19], is also depicted.

We see that the BFT-A4 NoCs require fewer LUTs than their BFT-A2 equivalents, but achieve higher WCPLs and lower



(a) Key

(b) Throughput

(c) Worst-Case Packet Latency

Fig. 4. LUT cost plotted against Throughput (packets / ns) and Worst-Case Packet Latency (ns) for the soc benchmark, system size of 64 PE.

Throughputs due to their lower clock frequencies. Thus, their interest would tend to lie in use-cases where designers have a constrained budget in terms of FPGA resources to implement the NoC.

A designer may not be working with a specific maximum budget of LUTs for the NoC, but rather may be interested in maximizing the ratio of throughput to the number of LUTs allocated for the network. The plot in Figure 5 shows the ratios of Throughput to LUT cost achieved by each BFT NoC for each benchmark, normalized to the ratio achieved by the **BFT0 (Arity=2)** NoC for the benchmark. For all benchmarks, the NoC from the BFT-A4 group always achieve higher ratios than those achieved by their BFT-A2 counterparts. Ratios are 85 to 120% higher for BFT0, 39 to 91% higher for BFT1, 15 to 31% higher for BFT2 and 37 to 54% higher for BFT3.

Finally, external factors may constrain the NoC frequency to be inferior to the maximum frequencies of BFT-A4 NoCs. In this case, we may express performance metrics in terms of clock cycles instead of ns, as the BFT-A2 and BFT-A4 NoCs may all be operated at the same frequency. Figure 6 shows the results of the same experiment depicted in Figure 4, but with Throughput expressed in packets / cycle and WCPL expressed in cycles. In this case, the BFT-A4 NoCs achieve Throughputs and WCPLs equal or better than those achieved by equivalent BFT-A2 NoCs, while requiring fewer LUTs.

VI. Conclusions

In this work, we added BFT NoC based on arity-4 switches to the collection of FPGA BFT NoCs with lightweight flow control. We also compared the performance of arity-2 and arity-4 BFT NoCs in terms of the throughput to LUT cost ratio, showing that the arity-4 BFT NoCs achieve higher ratios.

Because arity-4 BFT NoCs consume fewer LUTs and wiring length than their arity-2 counterparts, they likely also outperform them in power consumption. A future work could verify this hypothesis by augmenting our experimental infrastructure. Higher switch arities(e.g. arity-8) could also be explored, to verify if the trend of exchanging clock frequency for lower resource costs continues.

RTL → https://git.uwaterloo.ca/watcag-public/bft-flow-arity4

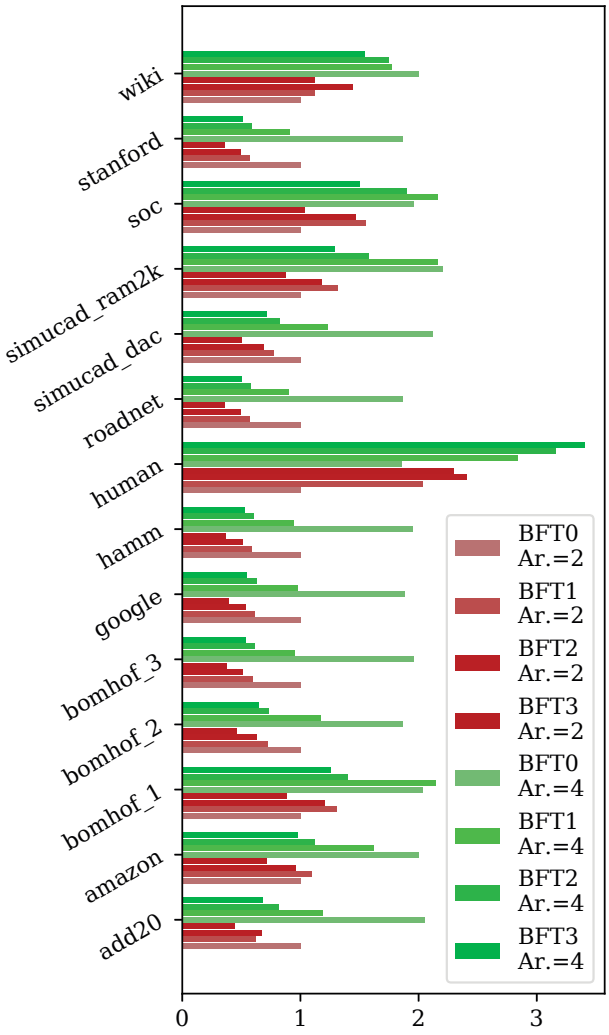
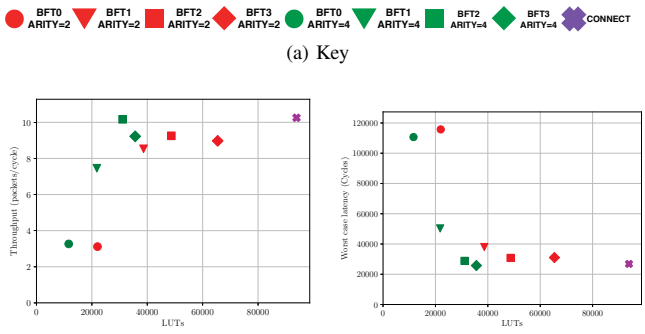


Fig. 5. Normalized ratio of Throughput (packets / ns) to LUT cost achieved by each BFT NoC for the benchmarks.



(a) Key

(b) Throughput

(c) Worst-Case Packet Latency

Fig. 6. LUT cost plotted against Throughput (packets / cycle) and Worst-Case Packet Latency (cycles) for the soc benchmark, system size of 64 PE.

References

[1] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, D. Firestone, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo,

T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, "Configurable clouds," *IEEE Micro*, vol. 37, no. 03, pp. 52–61, may 2017.

[2] Amazon Web Services, "Amazon EC2 F1 Instances." [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[3] I. Xilinx, "Baidu deploys xilinx fpgas in new public cloud acceleration services," https://www.xilinx.com/news/press/2017/baidu-deploys-xilinx-fpgas-in-new-public-cloud-acceleration-services.html, 2017.

[4] L. Huawei Technologies Co., "Fpga accelerated cloud server-huawei cloud," https://www.huaweicloud.com/en-us/product/fcs.html, 2019.

[5] Xilinx Inc., "Vivado High-Level Synthesis." [Online]. Available: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[6] Intel Corporation, "High-Level Synthesis Compiler - Intel HLS Compiler." [Online]. Available: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/hls-compiler.html

[7] I. Swarbrick, D. Gaitonde, S. Ahmad, B. Gaide, and Y. Arbel, "Network-on-chip programmable platform in versaltm acap architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: ACM, 2019, pp. 212–221. [Online]. Available: http://doi.acm.org/10.1145/3289602.3293908

[8] N. Kapre, "Deflection-routed butterfly fat trees on fpgas," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–8.

[9] G. S. Malik and N. Kapre, "Enhancing butterfly fat tree nocs for fpgas with lightweight flow control," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: ACM, 2019, pp. 308–308. [Online]. Available: http://doi.acm.org/10.1145/3289602.3294002

[10] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Transactions on Computers*, vol. C-20, no. 12, pp. 1469–1479, Dec 1971.

[11] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, Oct 1985.

[12] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787508

[13] Z. Wang, J. Xu, X. Wu, Y. Ye, W. Zhang, M. Nikdast, X. Wang, and Z. Wang, "Floorplan optimization of fat-tree-based networks-on-chip for chip multiprocessors," *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1446–1459, June 2014.

[14] Xilinx Inc., "Xilinx 7 Series FPGAs: The Logical Advantage." [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp405-7Series-Logical-Advantage.pdf

[15] Veripool, "Intro - Verilator - Veripool." [Online]. Available: https://www.veripool.org/wiki/verilator

[16] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[17] R. F. B. et al, "the matrix market: A web resource for test matrix collections," Quality of Numerical Software: Assessment and Enhancement, 1997.

[18] Xilinx Inc., "Vivado Design Suite." [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html

[19] M. K. Papamichael and J. C. Hoe, "Connect: Re-examining conventional wisdom for designing nocs in the context of fpgas," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12. New York, NY, USA: ACM, 2012, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145703