Appearing in *IEEE International Conference on Field-Programmable Technology* (FPT 2023), December 11–14, 2022

Asymmetry in Butterfly Fat Tree FPGA NoC

Dongjoon Park, Zhijing Yao, Yuanlong Xiao and André DeHon
Dept. of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA
Email: dopark@seas.upenn.edu, zyao@seas.upenn.edu, yuanlongxiao24@gmail.com, andre@ieee.org

***Abstract*—Among various topologies for FPGA overlay Network-on-Chip (NoC), the Butterfly Fat Tree (BFT) is known to be fast and lightweight. The BFT has a hierarchical structure that allows the routing capacity of each level to be configured with bandwidth-reducing t switches and bandwidth-preserving π switches, and this configuration can be exploited to customize the NoC resources, spending area as needed to match the bandwidth requirements of the application. However, a traditional BFT is symmetric: switch types in all subtrees in the same level are identical; this does not fully exploit the customization offered by the FPGA. We evaluate asymmetric BFTs that have different bandwidth in their subtrees, and we develop a *converging switch* built with t switches that connects subtrees with different bandwidths. Given the same resource budget, asymmetric BFTs perform better than symmetric BFTs when NoC traffic is highly unbalanced. In realistic workloads and statistical traffic patterns, asymmetric BFTs achieve up to 32% and 76% more throughput than symmetric BFTs, respectively.**

I. Introduction

The growing capacity of the modern FPGA and design complexity make the placement and routing process more challenging. As the design size increases and routing becomes complicated, interconnect can introduce significant overhead in area and energy. A packet-switched Network-on-Chip (NoC) can be a solution to routing-dominant SoC design. Instead of using a dedicated interconnect for all operators, we can time-share the NoC, and the NoC dynamically routes packets to the destination at runtime.

FPGA vendors provide a "Hard" NoC, an embedded NoC on the FPGA [1], [2], and researchers have long studied "Soft" NoC, an overlay NoC built on top of the commercial FPGA [3], [4]. While hard NoC can provide better performance per area, soft NoC is more flexible because topology or NoC configuration can be reconfigured according to the application needs. Among various topologies for soft NoC, Butterfly-Fat Tree (BFT) is cost-effective [5]–[7] and outperforms other state-of-the-art network topologies [8].

When a graph is mapped on a BFT, it is intuitive to bi-partition the graph in a way that the inter-partition communication is minimized to prevent the unnecessary traffic over the NoC. Then, each sub-graph can be assigned to each subtree of a BFT, and the bi-partitioning process continues for each subtree. However, there is no guarantee that the communication in each partition is the same. After the bi-partitioning, some portion of the graph could require more bandwidth. Fig. 1 is a real example of a graph workload where one-fourth of the nodes are located in each quadrant, and the thickness of the edge represents the communication volume. In Fig. 1's case, inter-partition communication is reduced with bi-partitioning (between the green nodes and the red nodes), but the communication requirements of the one half (within the green nodes) are heavier than the other half (within the red nodes). We observe that an asymmetric BFT performs better in such cases, providing more bandwidth where needed.

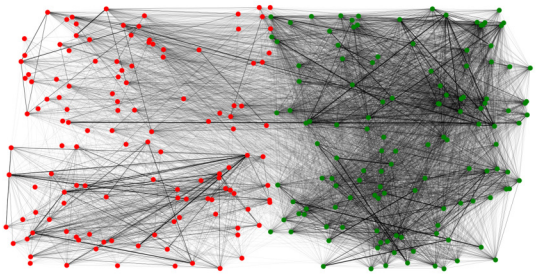

Fig. 1. An example of unbalanced sub-graphs after partitioning, `deezer-europe` workload from [10]

There is no single best soft NoC for all applications, but there are soft NoCs with different compositions that can perform better for specific applications. We expand the design space of soft NoC so that users can tailor the NoC to their applications, more fully exploiting FPGA's reconfigurability. Unlike previous literature that recommends a specific type of BFT based on the LUT budget on the FPGA [8], [9] independent of the application, we propose asymmetric BFT architectures that could exhibit better throughput than symmetric BFT for applications where the loads are unbalanced.

II. Background

Although many previous publications on FPGA overlay NoC employ a mesh topology for its simplicity, researchers have shown that BFT performs better than other topologies at equivalent area [8]. The BFT has a hierarchical structure with the Processing Elements (PEs) located on the lowest level. When a packet climbs up the hierarchy, it has exponentially increasing choices of paths, and when it climbs down, the path is determined by the address bits included in the packet.

Wiring capacity of a network can be described with Rent parameter p [11] where the larger value of p indicates the larger bisection bandwidth ($IO = cn^p, 0 \leq p \leq 1$). The primitive building blocks for BFT are t switches that have one parent port and π switches that have two parent ports as shown in Fig. 2. *Arity* refers to the number of the children ports, and in this paper, we consider arity-2 switches like

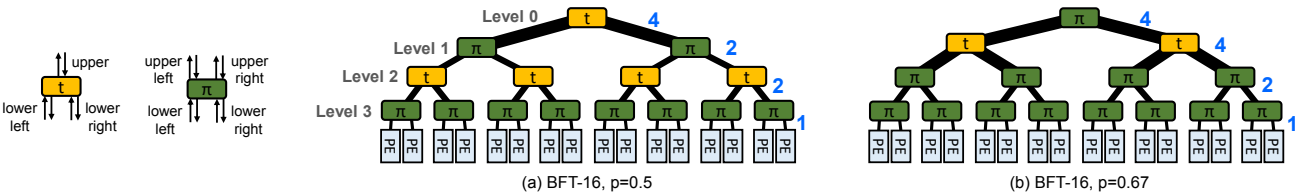

Fig. 2. Symmetric BFT-16 with different p values. By configuring switches in each level, we can adjust the bandwidth between the levels. However, symmetric BFT uniformly provides more bandwidth and forces additional resource costs.

the ones in Fig. 2 while different arity values are possible in the BFT architecture [12]. One differentiating factor of BFT compared to other topologies is that the bandwidth of each level of BFT can be configured by properly selecting t switches and π switches [13]. If we want more bandwidth between the specific levels, we can simply compose the layer with π switches. This flexibility sharply contrasts with other topologies like mesh where all channel widths need to increase together to support more bandwidth.

We build upon the packet-switched, deflection-routed BFT in [8]. Fig. 2 shows BFTs with 16 PEs. The width of connection between layers in Fig. 2 represents the communication bandwidth, and in the Fat-Tree-based topology, the communication is thicker at the higher level in the hierarchy. Blue numbers represent the channel widths in the communication, so in the non-lowest level in the hierarchy, there are multiple switches that make up each switching node. BFT-16 with $p = 0.67$ (Fig. 2 (b)) provides more bandwidth between level 1 and level 2 than BFT-16 with $p = 0.5$ (Fig. 2 (a)).

BFT's hierarchical structure offers finer-grained control on network channel bandwidth compared to other topologies. Nevertheless, in a symmetric BFT, like the ones in Fig. 2, since each level is homogeneously composed of either t switches or π switches, to provide more bandwidth, all t switches in the level have to be replaced with π switches, requiring more switch area. For instance, when placed and routed on Xilinx UltraScale+ ZU9EG, BFT-16 with $p = 0.67$ (Fig. 2 (b)) costs 7276 LUTs and 5991 FFs while BFT-16 with $p = 0.5$ (Fig. 2 (a)) costs 6248 LUTs and 5223 FFs.

III. Asymmetric BFT

When graph workloads are mapped on the network, we can place communicating nodes close to each other to exploit locality [14]. A fast and simple approach is a placement based on recursive bi-partitioning. A graph is bi-partitioned to minimize edge communication, and then each partition is assigned to a subtree of a BFT. In this way, subtrees are likely to communicate locally. However, we observe that in some applications, after the bi-partitioning, one partition exhibits more traffic than another. The problem is that, in a traditional symmetric BFT, as shown in Fig. 2, the layers of BFTs are composed of single type of switches, and it is not possible to selectively provide more bandwidth to some portion in the NoC than others.

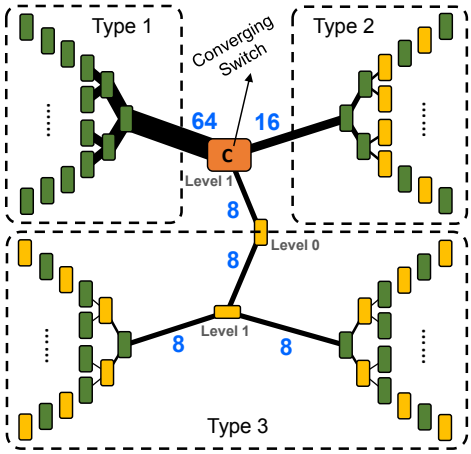

Fig. 3. Example of Asymmetric BFT-256

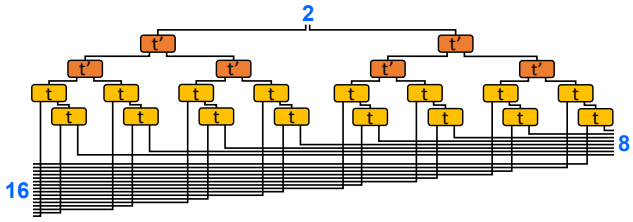

Fig. 4. 16-8-2 Converging switch built with t switches and $t-random$ (t') switches. Note that the two lowest levels consist of t switches and the upper levels consist of $t-random$ switches.

To support heterogeneous bandwidth with the similar resource usage, we explore asymmetry in the BFT. We define an asymmetric BFT as a BFT that has different switches in a given level. Fig. 3 is an example of an asymmetric BFT that has 256 PEs. The color scheme for the switches is consistent with that of Fig. 2. Type 1's subtree consists of π-π-π-π-π-π, Type 2's subtree consists of π-t-π-t-π-π, and Type 3's subtree consist of t-π-t-π-t-π-t switches. Therefore, the Type 1 subtree is denser and provides more bandwidth for the PEs, and the Type 3's subtree is sparser and provides less bandwidth.

In Fig. 3's example, Type 1's subtree results in the channel width of 64 to the level 1, and Type 2's subtree results in the channel width of 16 to the level 1. Type 3's subtree results in the channel width of 8 to the level 0. Thus, we need a component in the level 1 that reduces the channel width to match the bandwidth of the top and the bottom subtrees. We introduce a *converging switch* that reduces the channel width

from the top subtrees to the bottom subtree.

While it is possible to build the converging switch out of the original t switches only, using only t switches would not spread out the traffic to exploit the wider channels; instead it would concentrate the traffic and leave portions of the rich bandwidth unused. Therefore, we use standard t switches in the lowest level and use $t-random$ switches (t') in the higher levels as shown in Fig. 4. In the switches of deflection-routed BFT, there are the *desirable direction* where the packet wants to go and the *final direction* where the packet ends up going if the contention exists. As all the t switches in the converging switch belong to the same level (level 1 in Fig. 3's case), in the strawman implementation of the converging switch that uses only standard t switches, the *same* bits in the packet are used to specify the desirable direction in all the stages of the converging switch. This means the desirable direction for a packet that is climbing down is either the lower left for all the t switches or the lower right for all the t switches, potentially causing congestion inside the converging switch. In the $t-random$ switch, on the other hand, the desirable downward direction ignores the destination bit and is set to the lower left at the one cycle and set to the lower right at the next cycle, alternately. These $t-random$ switches in the non-lowest levels of the converging switch spread out the traffic, and in the lowest level, the t switches send the packet based on the address bits in the packet to make sure that the packet is delivered to the correct subtree. The only difference between a $t-random$ switch and a t switch is the desirable downward direction (left or right). The rest of the architecture, like the packet arbitration, stays the same. For example, a deflected packet takes priority and is immediately turned back in the next cycle, as done in the base deflection-routing. Because we build on top of the deflection-routed BFT, deflection-routed scheme's randomness already requires reorder buffers (reassembly buffers) in PEs. Our $t-random$ switch adds no new requirements for reordering beyond those that already exist for the deflection-routed BFT. The benefit of $t-random$ is evaluated in Sec.V-C.

All the PEs in an asymmetric BFT can still communicate with each other but with higher bandwidth on specific subtrees and lower bandwidth on other subtrees. Fig. 4 is a converging switch that reduces the channel width of 16 and 8 to the channel width of 2, but the architecture can be extended to other power of two combinations, like 64-16-8 in Fig. 3.

IV. Methodology

Tab. I describes two symmetric BFTs (S0, S1) and two asymmetric BFTs (AS0, AS1) that we use for the evaluation. These BFTs are all deflection-routed BFTs [8]. The number of PEs is 256. When BFT-256 has four subtrees with 64 PEs (st-0,1,2,3), both AS0 and AS1 have two dense subtrees and two sparse subtrees. Subtrees of different densities are connected with a converging switch. The switch composition column refers to the switch type from the lowest level (leftmost) to the second-highest level (rightmost), so Fig. 3's asymmetric BFT corresponds to AS1. For Rent parameter p of symmetric BFTs (S0, S1), $p = 0.5$ is chosen because $p = 0.5$ is known to be area-universal, meaning that the networking resources are relatively well-balanced and scalable with the computation [5], [15]. While we provide two examples of asymmetric BFTs, the idea can be extended to any number of different subtrees with appropriate converging switches.

We run synthesis, placement and routing with Xilinx Vivado 2022.1 targeting UltraScale+ ZU9EG FPGA to extract resource usage. Dummy PEs are attached to the NoC for testing purposes. A packet consists of a single flit with 1 valid bit, 8 bits of PE address, an 11-bit sequence number and 32 bits of data. The packet composition can change depending on design requirements, but it should be the same for both symmetric and asymmetric BFTs for a fair comparison. Both symmetric and asymmetric BFTs need sequence bits in the packet for reorder buffers in the PEs as they use the deflection-routed scheme.

TABLE I
Symmetric BFT-256s (S0, S1) and Asymmetric BFT-256s (AS0, AS1) example (52b, single flit packet)

	LUTs	Switch Composition	Cnvg. Switch	
			Comp.	LUTs
S0	122778	$p = 0.5$ π-t-π-t-π-t-π	-	-
S1	143870	$p = 0.5$ π-π-t-t-π-π-t	-	-
AS0	142896	st-0,1: π-π-π-t-π-π-c	32-32-8	15829
		st-2,3: t-π-t-π-t-π-t		
AS1	143029	st-0: π-π-π-π-π-π-c	64-16-8	21480
		st-1: π-t-π-t-π-π-c		
		st-2,3: t-π-t-π-t-π-t		

st-i: subtree-i / c: converging switch

The converging switch columns (Cnvg. Switch) in Tab. I shows the configuration of the converging switch and the resource usage of the converging switch. AS1's converging switch (channel width of 64 is converged to the channel width of 8) has a deeper hierarchy than AS0's (channel width of 32 is converged to the channel width of 8), and the resource usage for the converging switch increases accordingly. We have a script[1] to generate Verilog codes for asymmetric BFTs, given the switch configurations. Asymmetric BFTs for the evaluation are selected so that they consume fewer LUTs than symmetric BFT-256 (S1) to be a fair comparison with the symmetric BFTs.

TABLE II
Worst Negative Slack (ns) for different switch types when placed and routed on ZU9EG, packet size = 52b

SW type	clock period (ns)						
	1.0	1.2	1.4	1.6	1.8	2.0	2.2
t	-1.21	**0.012**	**0.174**	**0.244**	**0.381**	**0.459**	**0.585**
t rnd	-0.181	-0.055	**0.157**	**0.168**	**0.328**	**0.487**	**0.703**
π	-1.217	-0.878	-0.720	-0.434	-0.340	-0.003	**0.053**

We also run synthesis, placement and routing on t switches, $t-random$ switches, and π switches separately, setting the

[1]scripts for RTL generation for asymmetric BFTs and all the experiments are open-sourced in https://github.com/icgrp/asym_bft

switch level as 7 (the lowest level). Tab. II shows the worst negative slacks when switches are routed with different system clock speeds. Max clock frequencies of t switch, $t-random$ switch, and π switch are estimated as 833 MHz, 769 MHz, 476 MHz based on the clock period and the slack. As π switch has more complex routing within the switch, max clock frequency is slower than t switch, consistent with the results from [8]. LUT costs for t switch, $t-random$ switch, and π switch are 171, 172, and 285–287 respectively. FF costs are 156, 157, and 209. The $t-random$ switch has almost the same logic complexity as the t switch; since the $t-random$ is still faster than the π switch, it will not limit the clock frequency of the system with a proper floorplanning.

We use `iverilog` to run simulations for realistic workloads and synthetic traffic patterns. After the simulation is finished, we check whether the messages are all properly transferred. Then, the worst-case latency and throughput are recorded.

V. Evaluation

A. Realistic workloads

Fig. 5 illustrates the throughput advantage of asymmetric BFTs for realistic, Graph Analytics workloads from [10]. The datasets are undirected graphs, and each graph edge counts for two packets, swapping the sender and the receiver. The total number of packets for each benchmark ranges from 16K to 485K. We use `metis` [16] to cluster the graph into 256 parts using a recursive bisection scheme with the objective to minimize the edge cuts. We check the number of messages whose destination is in the dense subtree (st-0,1) and the number of messages whose destination is in the sparse subtree (st-2,3). If the number of messages traveling to the sparse subtree is larger, we simply reverse the placement so that node 0 becomes node 255, node 1 becomes node 254, and so on. Within the dense subtree, if the number of messages traveling to st-1 is larger than st-0, we try reversed placement as well to benefit from AS1 which offers a large bandwidth in st-0. The number of total datasets is 60 including 32 reversed node placement versions. Thus, the number of unique datasets is 28. The *injection rate*, the rate that each PE sends messages to the network, is set to 100%, which means that all the PEs attempt to send valid packets every cycle. The throughput (pkt/cycle/PE) is the average packet delivery rate computed as the total number of packets divided by the total elapsed cycles, divided by the number of PEs.

Fig. 5 (a) shows that in realistic workloads, the asymmetric BFTs can achieve up to 32% higher throughput than the symmetric BFTs. Not all real-world applications exhibit asymmetric traffic, and for applications where the loads are relatively balanced, it is natural that symmetric BFTs are the better options. But even after the graph is bi-partitioned, one partition's communication can be heavier than another partition's like the benchmarks in Fig. 5 (a), and in such cases, asymmetric BFTs have an advantage over symmetric BFTs. In Fig. 5 (a), we selectively include benchmarks that exhibit at least 10% improvement in throughput with

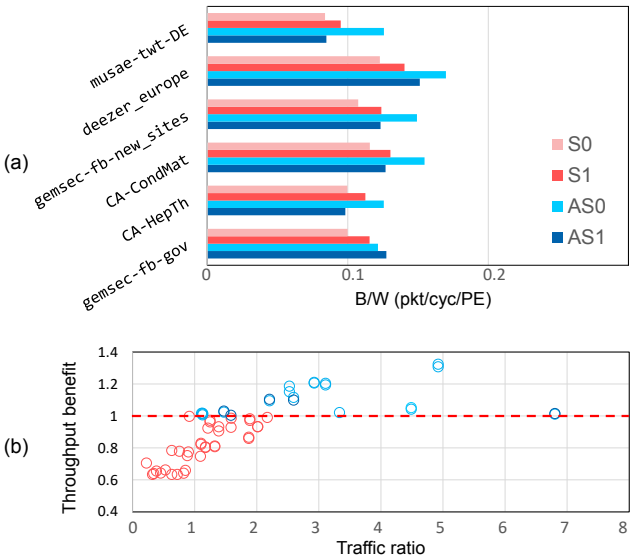

Fig. 5. (a): Throughput comparison on selected realistic benchmarks, (b): Throughput benefit (max(AS0,AS1)/max(S0,S1)) for different traffic ratios (# of messages ending in st-0,1/# of messages ending in st-2,3) in all realistic benchmarks

asymmetric BFTs (`musae-twt-DE`, `deezer-europe`, `gemsec-fb-new-sites`, `CA-CondMat`, `CA-HepTh`, `gemsec-fb-gov`).

We also evaluate the correlation (Fig. 5 (b)) between the throughput benefit of asymmetric BFTs and the traffic ratio of the dense subtrees (st-0,1) and the sparse subtrees (st-2,3) for all benchmarks. Throughput benefit is computed as the maximum throughput achieved by AS0 and AS1 divided by the maximum throughput achieved by S0 and S1. The traffic ratio is the number of messages delivered to PEs in dense subtrees divided the number of messages delivered to PEs in sparse subtrees. The color schemes of the markers correspond to the BFT type that performs the best for the graph workload. Therefore, markers whose throughput benefits are less than 1 are colored red (S0 or S1) and markers whose throughput benefits are greater than 1 are colored blue (AS0 or AS1). `musae-twt-DE` that shows 32% better throughput in asymmetric BFTs has a traffic ratio of 4.92. This is consistent with our expectation that asymmetric BFTs are better when there exists more traffic in the dense subtrees (traffic ratio $>$ 2.2) and there exists less traffic in the sparse subtree.

B. Random traffic

To better characterize the underlying phenomena, we also evaluate asymmetric BFTs with four different synthetic traffic patterns:

- **Test-0**: each PE randomly sends to another.
- **Test-1**: all PEs in st-0,1 are active and only 1/4 of PEs in st-2,3 are active. Each PE randomly sends to another.
- **Test-2**: each PE in st-0,1 randomly sends to another PE in st-0,1. The PEs in st-2,3 randomly send to PEs in st-0,1 with slow injection rate.

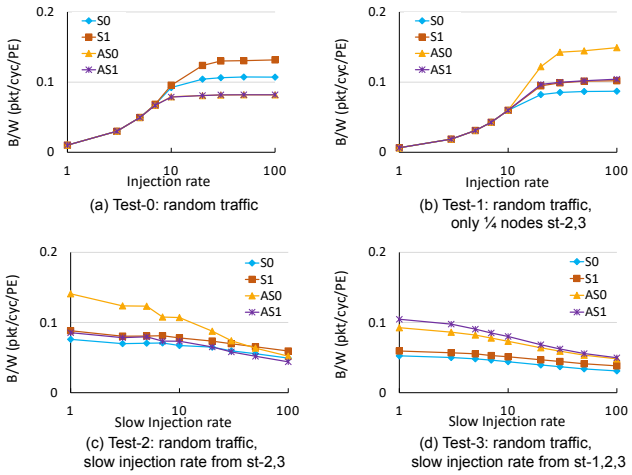

Fig. 6. Throughput comparison on different random traffic patterns

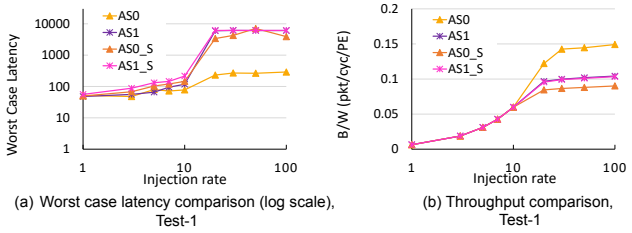

Fig. 7. Effect of $t-random$ switches in converging switch for Test-1

- **Test-3**: each PE in st-0 randomly sends to another PE in st-0. The PEs in st-1,2,3 randomly send to PEs in st-0 with slow injection rate.

Fig. 6 illustrates the throughput performances for symmetric BFTs and asymmetric BFTs on different traffic patterns. The number of messages per PE is set to 1024. We see that symmetric BFTs are better in Test-0 because the lower bandwidth in s-2,3 causes congestion in asymmetric BFTs. Test-1 and Test-2 are the simulated scenarios that st-0,1 are more active than st-2,3. As expected, AS0 that has more bandwidth in st-0,1 exhibits up to 46% (Test-1) and 60% (Test-2) improved throughput than S0 and S1. Test-3 is the simulated scenario that st-0 is more active than the others. As expected, AS1 that has more bandwidth in st-0 exhibits up to 76% improved throughput than S0 and S1. In Test-2 and Test-3, when the slow injection rate from the sparse subtrees is low enough, asymmetric BFTs perform better. But as the slow injection rate increases, the benefit in throughput decreases because the sparse subtrees in asymmetric BFT become congested.

C. t-random switch

To characterize the benefits of using $t-random$ switches inside the converging switch as described in Sec. III, AS0_S and AS1_S in Fig. 7 refer to corresponding asymmetric BFTs with a strawman implementation of a converging switch that consists of only standard t switches. In Test-1, $t-random$ switches significantly improve the worst-case latency (orders of magnitude) and the throughput (up to 65%), relieving the congestion in the leftmost switches and the rightmost switches.

D. Limitation

To take full advantage of asymmetric BFT, the users need to have some understanding of the application's traffic pattern because they need to configure the asymmetric BFT to provide more bandwidth where needed. We believe such a constraint is acceptable as soft NoC can always be reconfigured with other logic on FPGA. Because the resource utilization of symmetric BFT and the resource utilization of asymmetric BFT are similar, users can select the NoC overlay, leaving PEs untouched.

VI. Conclusions

The advantage of soft NoC on top of reconfigurable fabric is that users can customize the NoC to the applications, and the asymmetric BFT provides more options to the users. We demonstrate that given the same LUT budget, in realistic workloads and different random traffic patterns, asymmetric BFTs with converging switches built with $t-random$ switches can achieve up to 32% and 76% more throughput than symmetric BFTs.

References

[1] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versal architecture," in *FPGA*, 2019.
[2] *Speedster7t Network on Chip User Guide (UG089)*, Achronix Semiconductor Corporation, 2903 Bunker Hill Lane, Santa Clara, CA 95054, 2019.
[3] N. Kapre, N. Mehta, M. deLorimier, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, and A. DeHon, "Packet-switched vs. time-multiplexed FPGA overlay networks," in *FCCM*. IEEE, 2006, pp. 205–213.
[4] T. Marescaux, V. Nollet, J.-Y. Mignolet, A. B. W. Moffat, P. Avasare, P. Coene, D. Verkest, S. Vernalde, and R. Lauwereins, "Run-time support for heterogeneous multitasking on reconfigurable SoCs," *INTEGRATION, The VLSI Journal*, vol. 38, no. 1, pp. 107–130, 2004.
[5] C. E. Leiserson, "VLSI theory and parallel supercomputing," MIT, 545 Technology Sq., Cambridge, MA 02139, MIT/LCS/TM 402, May 1989.
[6] A. DeHon, "Compact, multilayer layout for butterfly fat-tree," in *Proc. SPAA*. ACM, July 2000, pp. 206–215.
[7] ——, "Rent's rule based switching requirements," in *Proc. SLIP*. ACM, March 2001, pp. 197–204.
[8] N. Kapre, "Deflection-routed butterfly fat trees on FPGAs," in *FPL*, Sept 2017, pp. 1–8.
[9] G. S. Malik and N. Kapre, "Enhancing butterfly fat tree NoCs for FPGAs with lightweight flow control," in *FCCM*, 2019.
[10] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, 2014.
[11] B. S. Landman and R. L. Russo, "On pin versus block relationship for partitions of logic circuits," *IEEE Trans. Comput.*, vol. 20, 1971.
[12] I. Lang, Z. Huang, and N. Kapre, "Exploring the impact of switch arity on butterfly fat tree FPGA nocs," in *FCCM*, 2020.
[13] A. DeHon, "Unifying mesh- and tree-based programmable interconnect," *IEEE Trans. VLSI Syst.*, vol. 12, no. 10, pp. 1051–1065, October 2004.
[14] N. Kapre and A. DeHon, "An NoC traffic compiler for efficient FPGA implementation of sparse graph-oriented workloads," *Int. Jrnl of Reconf. Comp.*, vol. 2011, March 2011, article ID 745147.
[15] A. DeHon, "Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% lut utilization)," in *FPGA*, February 1999, pp. 69–78.
[16] G. Karypis and V. Kumar, "MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0," http://www.cs.umn.edu/~metis, University of Minnesota, Minneapolis, MN, 2009.