Enhancing Butterfly Fat Tree NoCs for FPGAs with lightweight flow control

Gurshaant Singh Malik
University of Waterloo
Ontario, Canada
gsmalik@uwaterloo.ca

Nachiket Kapre
University of Waterloo
Ontario, Canada
nachiket@uwaterloo.ca

***Abstract*—FPGA overlay networks-on-chip (NoCs) based on Butterfly Fat Tree (BFT) topology and lightweight flow control can outperform state-of-the-art FPGA NoCs, such as Hoplite and others, on metrics such as throughput, latency, cost and power efficiency, and features such as in-order delivery and bounded packet delivery times. On one hand, lightweight FPGA NoCs built on the principle of bufferless deflection routing, such as Hoplite, can deliver low-LUT-cost implementations but sacrifice crucial features such as in-order delivery, livelock freedom, and bounds on delivery times. On the other hand, capable conventional NoCs like CONNECT provide these features but are significantly more expensive in LUT cost. Butterfly Fat Trees with lightweight flow control can deliver these features at medium cost while providing bandwidth configuration flexibility to the developer. We design FPGA-friendly routers with (1) latency-insensitive interfaces, coupled with (2) deterministic routing policy, and (3) round-robin scheduling at NoC ports to develop switches that take 311-375 LUTs/router. We evaluate our NoC under various conditions including synthetic and real-world workloads to deliver resource-proportional throughput and latency wins over competing NoCs, while significantly improving dynamic power consumption when compared to deflection-routed NoCs. We also explore the bandwidth customizability of the BFT organization to identify best NoC configurations for resource-constrained and application-requirement constrained scenarios.**

RTL $\rightarrow$ *https://git.uwaterloo.ca/watcag-public/bft-flow*

I. Introduction

FPGAs have become first-class citizens in the data-center as illustrated by the Microsoft Catapult [5] project and rapid deployment in cloud service providers such as Amazon F1. FPGAs are gaining prominence as a programmable computational platforms for contemporary applications such as Machine Learning, Graph Accelerators, Network Virtualization, among others. Modern FPGAs also support a growing number of diverse, high-speed interfaces for programmable communication with the outside world and also integrate optional features such as deep pipelining and lightweight absorption FIFO support that can boost design frequencies and increase throughputs. Programming FPGAs through High-Level Synthesis tools like Vivado HLS, Intel a++, and OpenCL workflows have put FPGAs within reach of software developers. As a result of this, FPGAs have not only become more accessible, but also support the design of large and complex applications. With increasing complexity, diversity of external interfacing, and tighter design schedules of software-like development, modular design flows have become necessary. NoC (Network-on-Chip) overlays are necessary to support connectivity requirements of these modular designs, while providing smooth access to system-level interfaces.

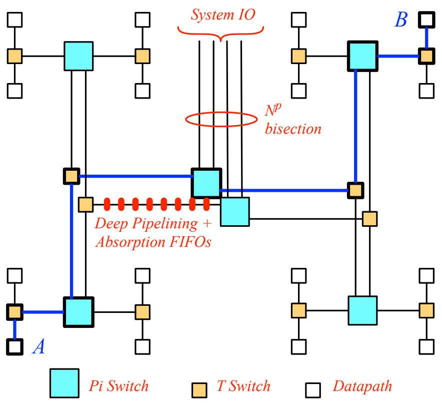


Fig. 1. High-Level Picture of the Butterfly Fat Tree (BFT) NoC. Configurable bisection bandwidth N^p with p=Rent parameter. Deterministic $A \rightarrow B$ path highlighted. Parametric pipelining of wires in upper stages.

Overlay NoCs complement hardened NoC resources by providing much-needed configurability and *last-mile connectivity* options to the FPGA developer. The design of efficient overlay NoCs can also help us customize the programmable nature of FPGA resources to different FPGA application requirements. Overlay NoCs such as Hoplite [13] demonstrate a radically different approach towards NoC design for FPGAs built around deflection routing principles that are low-cost and significantly better in cost and throughput compared to contemporary alternatives such as CMU CONNECT [20] and Penn Split-Merge [11] designs. However, deflection-routed NoCs are prone to livelock, unable to deliver packets in-order, and impose reordering cost on the destination. Deflection-Routed Butterfly Fat Trees [12] do marginally better by allowing the bisection bandwidth of the NoC to be customized to match application requirements, but otherwise suffer from rest of the compromises of a deflection-routed design.

In this paper, we show how to design an in-order, low-cost, FPGA-friendly Butterfly Fat Tree (BFT) NoC that supports customization of bisection bandwidth, without suffering the compromises of a deflection-routed NoC, while exploiting FPGA-specific features. As shown in Figure 1, we can configure the bisection bandwidth of the NoC by choosing a combination of pi (🟦) and t (🟧) switches. This is in stark contrast to the Fat Tree topology option in CMU CONNECT [20] that does not provide this parameterization and has a rigid organization. We add deterministic routing support to the Fat Tree routing algorithm to ensure packet flow between a source-destination endpoint takes identical paths. This is highlighted by a candidate path from A→B in Figure 1 which takes that strict path even when the pi switch hops provide alternatives on the uphill climb. We adopt a lightweight latency-insensitive interface on the NoC ports to (1) ensure packets are delivered in-order, (2) do not suffer the penalty of deflections, and (3) are AXI-S compatible. We augment the latency-insensitive interfaces with absorption FIFOs to allow deep pipelining in the upper stages of the tree, illustrated as ⊢⊢⊣ in Figure 1.

They key contributions of this work include:

- Design and RTL implementation of BFT NoC routers with latency-insensitive AXI-S I/O, deterministic routing function, round-robin arbitration, LUT-fracturing friendly mapping, and NoC assembly with configurable bisection bandwidth tuning and optional pipelining support.
- Performance evaluation of the different NoC configurations under synthetic and real workloads. Measurement and analysis of sustained throughput, average latency, worst-case latency, power reduction due to avoided deflections.
- FPGA floorplanning and mapping characterization of resource cost, frequency, and dynamic power.

II. Background

Modern FPGA chips pack millions of LUTs, thousands of BRAMs and DSPs, and a rich set of I/O interfaces on the same chip. This high compute density is matched by a corresponding increase in connectivity choices at the system-level. Modern FPGA IO interfaces include high-bandwidth DDR4 memories (21.3 GB/s), HBM and HMC memories (460–480 GB/s), along with next-generation network ports (100–400 Gbps). Besides matching the high data rates of these system-level interconnect, we often require guarantees on ordering, latency, and synchronization on these communication channels. These growing connectivity requirements have long suggested a need for low-cost, high-bandwidth overlay NoCs [13] or hardened NoCs [1] for FPGA chips. In fact, the HBM AXI IP [25] for the Xilinx VU37P integrates two 16×16 256b AXI crossbars to ensure uniform access to the HBM memory stack. Beyond one-off solutions for such interfaces, we need FPGA overlay and hard NoCs to support intra-FPGA traffic patterns.

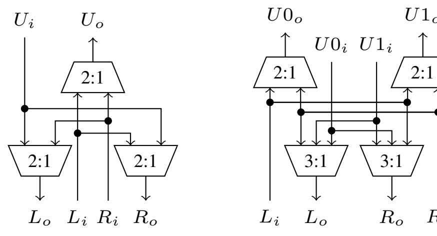
A. Survey of Contemporary FPGA NoCs

We now review key state-of-the-art FPGA overlay NoCs to better understand the underlying tradeoffs. Deflection-routed NoCs such as Hoplite [13] [14] and BFTs [12] have upended conventional wisdom in the design and engineering of FPGA overlay NoCs. These NoCs are low-cost, FPGA-friendly solutions that emphasize economy of LUT usage while sacrificing other features of the communication network. Fully-featured NoC frameworks like CONNECT [20] provides developers with a host of choices including topology, buffering, virtual channel support, flow control options, and others to match developer requirements but do so at significant LUT expense. Off-the-shelf IPs such as the Xilinx AXI4-Stream Interconnect exploit mux cascades in Xilinx logic fabric to deliver monolithic AXI-compatible router blocks but have limited scalability (16-endpoints) and poor scheduling.

1. **Implementation Cost**: Deflection-routed NoCs are significantly cheaper to realize on FPGAs and even on ASICs [7]. By avoiding the use of buffers, Hoplite and Deflection-routed BFTs offer a cheaper alternative to classic buffered NoCs. They are easier to pipeline owing to fewer logic stages in the internal crossbar and can easily run at high speeds subject to appropriate pipelining. However, deflections keep the NoC switching activity high by forcing the packets to keep traveling in the network.
2. **Performance**: Deflection routing offers high absolute throughputs when considering implementation frequency (GB/s) but low relative throughputs (packets/cycle). They also suffer from higher worst-case routing latency due to the overhead of deflections. In contrast, conventional NoCs like CONNECT supports higher packet throughput (packets/cycle) but suffers on absolute throughput metrics (GB/s) due to significantly lower implementation frequency/fmax. The Xilinx AXI4-Stream Interconnect IP only scales to 16 endpoints but offers competitive throughput.
3. **Packet Delivery**: Deflection routers are unable to offer in-order packet delivery due to the unpredictable order of deflections. Deflection routers misroute packets along unintended directions to manage contention without the need for buffering. It is possible to provide ordering in presence of deflection [19] at the cost of a significant reduction in throughput. Stateful streaming FPGA applications require ordered packet delivery and order DMA transactions on external DRAMs. Deflection routed NoCs are also susceptible to **livelocks** where unlucky packets keep getting misrouted perpetually. While this behavior is very uncommon, even a possibility of a livelock makes the design of coherence protocols, implementation of synchronization operations on FPGAs, and realization of real-time applications with strict timing guarantees difficult. Conventional buffered NoCs like CONNECT, and Xilinx AXI4-Stream Interconnect IP can provide in-order delivery at great cost.

B. Butterfly Fat Trees

Most modern NoC topologies are either 2D meshes or tori due to the geometric simplicity of mapping to a 2D VLSI substrate. However, these topologies are unable to scale bisection bandwidth to meet differing application requirements. Instead of making all chip-spanning NoC links wider to boost throughput, Butterfly Fat Trees (BFTs) offers customizable bisection bandwidth to the communication workload with configurable multi-level NoC topology. Through proper selection of switch richness at different levels of the tree we can tune the network to any bisection bandwidth $O(N^p)$ where p is the Rent parameter [16] of the architecture. BFTs have a



(a) t switch (b) pi switch

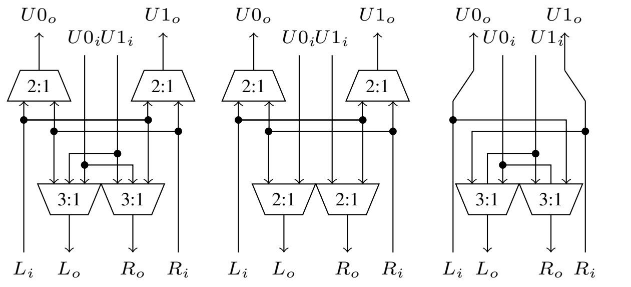
Fig. 2. Microarchitecture of BFT t and pi switches.

long history [18] [17] and have found use in traditional ASIC designs [22] [3] [6], Multi-Processor SoC designs [24] [10] [9] and off-chip networks in the data-center [23] [2].

Switch Microarchitecture: A popular version of 2-ary BFT NoCs is constructed from two types of building blocks: t and pi switches as shown in Fig 2. All switches at a given level of the tree are of the same type. A t switch has two ports to/from the lower levels of the tree and only a single uphill port. A pi switch has two ports from the lower levels of the tree, and two ports to the upper levels of the tree, thereby preserving total bandwidth. Even if the IO ports are buffered [15], the NoC still delivers packets out-of-order as the uphill ports still provide two choices at each climb. If the ports are bufferless [12], with a suitable choice of routing algorithm, the NoC switches are low-cost but, as discussed previously, guarantee neither packet ordering nor livelock freedom.

Bandwidth of a BFT: Fat trees allow the network bisection of the NoC ($O(N^p)$) to be adjusted as required to reflect Rent properties (p) of the communication workload. We can consider two extreme configurations of a BFT to better understand the tunable range of bisection bandwidth configurations available to us. On one hand, a binary tree (**BFT0**), configured by simply choosing t switches at all levels at a cost of $O(N)$ switches, can only sustain nearest-neighbor style lightweight communication (bisection bandwidth of $O(1)$). On the other hand, a multi-stage crossbar (**BFT3**), configured by choosing only pi switches at all levels at a cost of $O(N \times log(N))$, supports all-to-all communication patterns (bisection of $O(N)$) in a cost effective manner. By choosing the right mix of t and pi switches, we can construct the right NoC to suit application demands, a key requirement of FPGA workloads. For this paper, we will refer to the (t-pi-t-pi) topology as **BFT1** and the (t-t-pi-pi) topology as **BFT2**.

Routing Policy of a BFT: For packet-switched operation, we assume single-flit operation where the destination address and payload are part of a single indivisible unit of communication. The routing algorithm used on BFTs [12] is straightforward and can be separated into three phases: (1) climbing, (2) turning, and (3) descent. During the climbing phase, packets can take **any** available uphill port until they reach the common ancestor of both source and destination clients. The level of the NoC that represents the common ancestor is the bit position of the common prefix of both addresses. The packet must turn here and has only one choice



(a) Orig. pi switch, all turns allowed. (b) Downhill locked $U0 \rightarrow L$, $U1 \rightarrow R$. (c) Uphill locked $L \rightarrow U0$, $R \rightarrow U1$.

Fig. 3. Pruning the pi switch to enforce deterministic routing policy and save multiplexing cost.

of outgoing port. During descent, at level i of the NoC, extract bit i of the destination address and turn left (0) or right (1). Again the downhill path is fixed based on destination address and no routing freedom is available during descent.

III. BFTs WITH LIGHTWEIGHT FLOW-CONTROL

Butterfly Fat Trees offer a configurable platform for the design of FPGA-friendly NoCs. Conventional buffered variants offered by CONNECT are too large and slow, while bufferless deflection-routed variants are cheaper but suffer from livelocks, lower packet throughputs and absence of bounds on packet delivery. *Can we retain the features of a buffered NoC while approaching the low cost potential of bufferless deflection-routed NoCs?* Let us find out by first understanding the root causes of problems on deflection routed BFTs. Then we will walk through the proposal for rectifying these limitations with our BFT NoC design that employs lightweight flow control.

A. Designing the BFT Switch Microarchitecture

The broad design goals for the switch include in-order delivery, livelock freedom, and bounded routing times.

Quest for Deterministic Routing The key feature of the classic BFT routing policy allows packets routing via pi switches to choose uphill routes as required, depending on local congestion. This baseline switch, shown in Figure 3 (a) costs us 2 `2:1` multiplexers (uphill ports), and 2 `3:1` multiplexers (downhill ports). This generous routing freedom can lead to packet reordering that is typically handled with reassembly buffers at the endpoints. Instead, we can strategically limit this freedom of path selection while distributing bandwidth fairly across the NoC. For datacenter BFT networks, deterministic routing can be enforced [8] by scattering packets based on destination address. This is applicable to larger-scale networks as the cost of switch has already been paid and no optimization of switch richness is necessary. Instead, for an efficient FPGA mapping, we investigate LUT-friendly deterministic routing choices as shown in Figure 3.

- We can limit routing freedom by pruning the downhill muxes and forcing packets to strictly take $U0 \rightarrow L$ and $U1 \rightarrow R$ paths as shown in Figure 3b. This reduces the

LUT mapping cost of the switch to 2 `2:1` multiplexers (uphill ports) and 2 `2:1` multiplexers (downhill ports).

- Alternatively, we can restrict uphill packets to take $L \rightarrow U0$ and $R \rightarrow U1$ paths as shown in Figure 3c. This will eliminate the uphill multiplexers entirely. We now only need to pay the cost of 2 `3:1` multiplexers for the downhill ports.

By restricting routing freedom, packets take deterministic paths along the tree and packets for a particular source-destination pair will travel along identical routes in the tree.

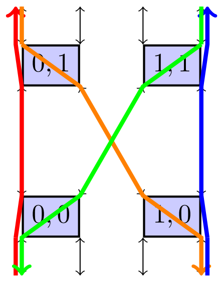


Fig. 4. Livelock in BFTs: mixing deflection and determinism.

However, if we add this restriction to deflection-routed BFTs, *the NoC loses livelock freedom*. A concrete example of this manifestation is shown in Figure 4. In this example, the red path is forced to take $L \rightarrow U0$ route at 0,0 and blue path must take $R \rightarrow U1$ at 1,0 thereby creating a cyclic dependency. The orange and green downhill paths also have a fixed route during BFT descents. The four co-dependent flows can lead to livelock. Deflection-routed BFTs force packets to return along the arriving links and create a local dependency on the output resource. When we enforce uphill routing restrictions, packets no longer have the ability to route around congestion. Recall, that during BFT routing, downhill paths are already constrained to a specific path based on the destination address. If a set of packet flows request resources in a cyclic fashion, they are now susceptible to a livelock that previously did not exist under the relaxed routing policy of the BFT NoC.

Quest for Low-Cost Handshaking: The key to breaking cyclic livelocks is to eliminate the dependency on the deflected outgoing port. Instead of sharing the outgoing port for both deflected packets and nominally routed packets, we choose to use a lightweight latency-insensitive interface for handling contentions. With latency-insensitive protocol, the contending packet is stored in a shadow register at the input port itself rather than deflecting it on the output port. This removes the dependence on the output port and eliminates livelock. This makes the NoC handshake-driven (**valid, backpressure protocol**) rather than built on deflection-routing principles.

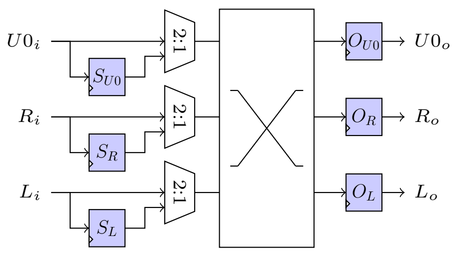


Fig. 5. Lightweight Latency-Insensitive Interface integration with the BFT t switch. Each input port has a cheap shadow register (single FF, or an SRL32 absorption FIFO).

Previous embodiments of this handshake [4] suffered from some key limitations. They were (1) mapped to cheap SRL FIFOs but limited by the logic cost of complex flow control, (b) throttled throughput by 50% to simplify the control logic for handshaking, and (c) compromised frequency of the mapped design in the quest for low LUT footprint. Neither of these alternatives are effective for NoCs where the cost of every extra LUT matters. We adapt the latency insensitive interface to compose seamlessly with the multiplexing stage of our internal switching crossbar to lower overheads. A side-effect of the adoption of handshaking, combined with the proposed deterministic routing, also enforces ordering guarantees on the packets in addition to livelock-free deterministic routing.

Round-Robin Arbitration: The final piece of the puzzle is the choice of arbitration policy. We manage contentions using a fair, work-efficient arbiter that provides service to contending packets in a round-robin fashion. Since packets are not deflected, they just stay in the shadow registers, service is guaranteed and no local livelocks are possible. Conventionally deflection-routed NoCs [12] can suffer from local livelocks due to mismatch between pipelining latency on the link and service cycle period of the round robin arbiter.

B. *FPGA Mapping*

We now investigate the FPGA LUT cost and implementation frequency of the different BFT router variants in Table I for the Xilinx UltraScale+ VU9P FPGA with Vivado 2017.4 running synthesis, placement and routing. In contrast to the Deflection BFT router [12], we introduce a lightweight shadow register module and a different round robin arbiter. For the pi switch proposed in this paper with deterministic uphill paths, we are able to save the cost of the muxes but still need to pay for the latency insensitive interface blocks. Thus, our proposed t and pi switches are 1.5–3× larger than their vanilla deflection-oriented counterparts but provide 1.) livelock-free routing, 2.) in-order delivery, and 3.) a manyfold increase in throughput, latency and power performance, as illustrated in Section IV. We also augment our lightweight router with deep pipelining along the links supported with lightweight 32-deep absorption SRL FIFOs. This further increases area cost by 1.5× and drops frequency slightly. Later, we see in Section IV, that the designer may choose between wire-rich or LUT-rich alternatives for a given workload.

In contrast to a recent [21] partial-reconfiguration friendly implementation of a 32-node 64b deflection-routed BFT taking 29K LUTs on a Xilinx ZU9EG MPSoC, our topology takes ≈30K LUTs on a VU9P FPGA and provides multiple extra benefits in exchange for the modest increase in LUTs and identical wiring requirement.

IV. Results

In this section, we report the outcomes of the experimental evaluation of the our Flow-controlled BFT FPGA overlay NoCs and other contemporary NoCs under different traffic patterns. We quantify metrics such as throughput, latency (source queuing+inflight), resource cost and power. We aim

TABLE I
BFT t AND pi SWITCH COST (32B DATA + 4B ADDR).

Architecture	**Kind**	**LUTs**	**FFs**	**Freq.** (MHz)
Deflection BFT	t	122	146	1176
[12]	pi	207	145	965
This paper	t	311	216	780
	pi	375	288	881
This paper	t	530	282	751
+ SRL FIFOs [4], [15]	pi	548	378	875

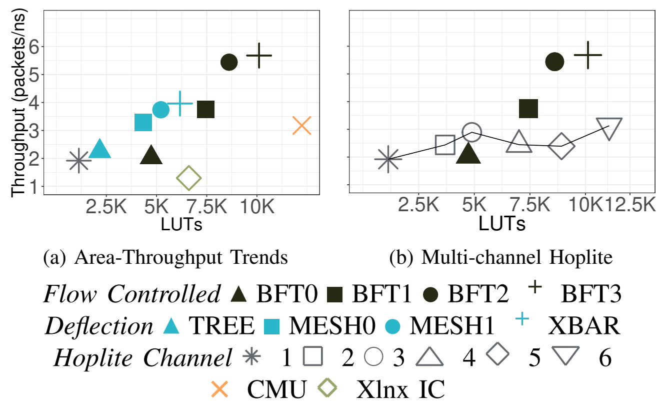
to establish the tradeoffs between the different NoCs and how to choose between them under various constraints.

Experimental Setup: For synthetic workloads, we investigate uniform random and locality-aware traffic patterns that send single-flit messages (1024 packets/client). We extract real-world traces from FPGA accelerators for Sparse matrix-vector multiplication, graph processing, and Token LU factorization graphs with 10K–1M messages/benchmark. We identify Hoplite, Deflection BFT, CMU CONNECT, and Xilinx AXI4-Stream Interconnect (IC) as the set of NoCs for head-to-head comparison across a range of system sizes from 2–64. For the Flow-controlled BFT, we choose four configurations (BFT0–3) between and inclusive of the binary tree (BFT0, all t switches) and multi-stage crossbar topologies (BFT3, all pi switches). We mimic similar four BFT configurations from the Deflection BFT work [12] (TREE, MESH0, MESH1, and XBAR). The CMU CONNECT BFT is configured to have 8-deep buffers, peek flow control, and separable input-first round-robin arbitration to closely match our testing conditions. We run cycle-accurate simulations using a combination of `iverilog`, `verilator`, and `Xsim` as required by the different NoCs. All hardware mapping experiments targeted the Xilinx UltraScale+ VU9P FPGA (xcvu9p-flga2104) with a -3-e speed grade using Vivado 2017.4 under default synthesis and implementation settings. We floorplan the NoCs to span the chip dimensions and optimize layout for wiring distribution.

A. *Cost-Power-Performance Tradeoffs*

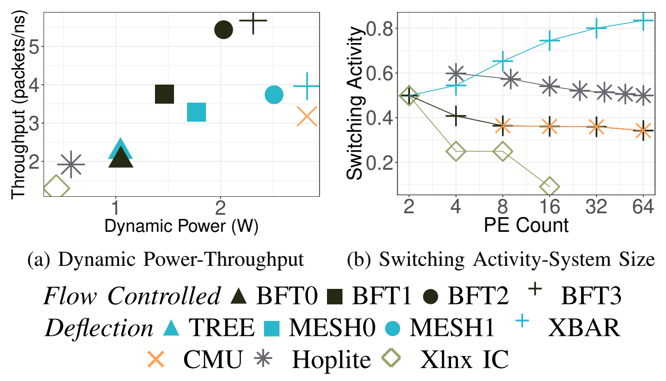
In Figure 6a, we show sustained throughput (packets/ns) as a function of LUT utilization at 16 client system size. It is clear that Hoplite has the smallest LUT footprint of all NoCs, but the superior bisection bandwidth of the BFT NoCs help boost their sustained rates by as much as 3× over Hoplite. The BFT proposed in this paper is more expensive than the deflection variant by 1.5–2× but delivers 1.5× better throughput in exchange. The CMU CONNECT NoC is significantly more expensive (1.5–3×) than the BFT NoCs and delivers 1.8× lower throughput , while the Xilinx IC has remarkably poor 6× less throughput than our BFTs.

It may seem tempting to exploit the low-cost nature of Hoplite to create replicas of the NoC and use it instead of paying for the more expensive BFT. In Figure 6b, we quantify the fallacy of this thinking. The baseline Hoplite design consumes 1.1K LUTs compared to the 10K LUTs of our fully-feature BFT. For `RANDOM` workload with 100% injection rate, we observe that multi-channel Hoplite with 3 channels is able to deliver a better throughput result compared



(a) Area-Throughput Trends (b) Multi-channel Hoplite

Fig. 6. FPGA LUT cost-performance trends for different NoCs. 16 clients, `RANDOM` traffic at 100% injection.



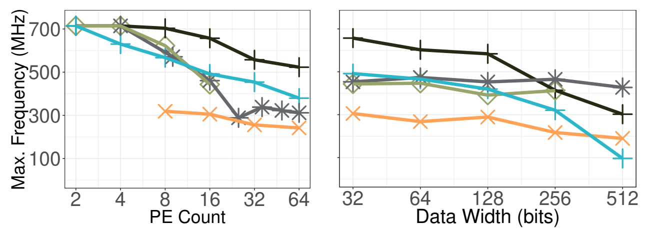
(a) Dynamic Power-Throughput (b) Switching Activity-System Size

Fig. 7. FPGA Dynamic Power-performance trends for different NoCs. 16 clients, `RANDOM` traffic at 100% injection.

to a BFT0 NoC. As we scale channel count, the Hoplite throughput saturates and is not competitive with rest of the BFT NoC configurations. This is for two reasons: (1) a drop in implementation frequency as we add more wires to the multi-channel NoC, and (2) ejection bottleneck where packets try to exit the NoC from the multiple channels. Thus, a bisection configurable network like a BFT offers better use of wiring resources than naive replication of a low-cost topology.

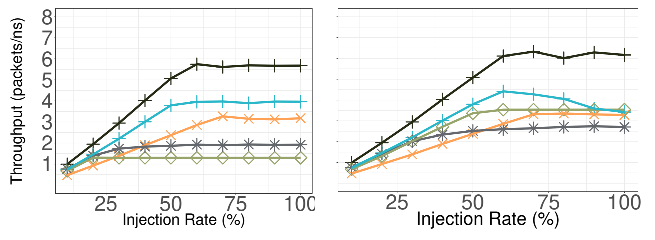
Next, we investigate power-throughput tradeoffs to primarily quantify the benefit of avoiding deflections. We expect the use of lightweight handshakes to eliminate the switching activity due to deflections. We compute dynamic power using `Xilinx Xpower` by supplying switching activity rates. We extract these switching activity rates from simulation data. In Figure 7a, for `RANDOM` workloads at 100% injection rate, we observe a simultaneous 1.1–1.2× reduction in dynamic power and 1.1–1.5× boost in throughput when comparing our BFTs with their deflection variants. While Hoplite has low dynamic power due to its dramatically lower LUT cost, it has high switching activity and struggles to meet the throughput capabilities of the BFT NoC.

In Figure 7b we measure switching activity as a function of system size for `RANDOM` workloads at 100% injection rate. As expected, deflection-based topologies like Hoplite and Deflection BFTs are characterized by high 60–80% activity rates. Our BFTs exhibits the low switching rates of 40–50%.



(a) System Size Scalability (b) Datawidth Scalability, 16 PEs

Fig. 8. FPGA frequency characterization as a function of system size and datawidth.



(a) RANDOM traffic (b) LOCAL traffic

Fig. 9. Sustained Throughput trends for different NoCs for 16 clients and for RANDOM and LOCAL traffic patterns.

The Xilinx IC is a single-stage crossbar and thereby exhibits the lowest activity rates.

B. Understanding FPGA Mapping Trends

To understand the source of these LUT cost efficiency trends, we now investigate the LUT implementation cost and operating frequency of the different NoC configurations.

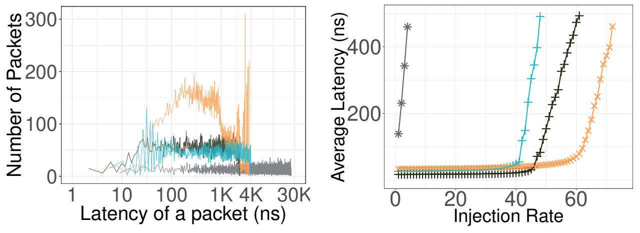
In Figure 8a, we implement the different NoCs with 32b datawidth and scale PE counts. We observe that our BFT is the fastest NoC and runs as fast as 700 MHz (2–8 PEs) and degrades to 500 MHz (64 PEs). It is also better than the deflection variant of the BFT by ≈1.25×. The Hoplite frequency generally keeps pace with the BFT and only slows down at larger system sizes. With aggressive link pipelining even at large system sizes, it is possible to achieve frequency-parity between Hoplite and BFT topologies. The CMU interconnect is the slowest NoC in this comparison due to complex router microarchitecture. The Xilinx AXI4-Stream Interconnect is fast at small system sizes but its frequency drops to 400 MHz at 16 clients due to a simple, low-cost design.

In Figure 8b, we scale the datawidth of the NoC with 16 clients. As expected, all NoCs suffer a frequency drop with widening links. Below 128b links the BFT NoC proposed in this paper offers the highest frequency across the set of NoCs. As we scale beyond 128b, the frequency of Hoplite and Xilinx AXI4-Stream interconnect is faster at around 500 MHz while the BFT drops to around 300 MHz. Note that with aggressive pipelining (extra register cost) of the links all NoCs can be made to achieve similar operating frequencies.

C. Effect of Injection Rate of Synthetic Traffic

When considering synthetic traffic patterns, we must examine the effect of injecting packets into the NoC at varying rates and measure sustained rates and average latency at saturation.

For RANDOM (Figure 9a) and LOCAL (Figure 9b) workloads, on a 16 client system, the BFT NoC outperforms all other NoCs in our comparison by 1.5–2× (worst case) and by 3–6× (best case). The key reason for this difference is the ability of our NoC to avoid the penalty of deflections (Hoplite, Deflection BFT), faster frequency (CMU CONNECT), and fairer arbitration (Xilinx AXI4-Stream Interconnect). As expected, the LOCAL traffic pattern suits the BFT architecture nicely as locality in the communication trace matches the hierarchical structure of the NoC.



(a) Latency distribution (b) Average Latency

Fig. 10. NoC packet latency trends for 64 client NoCs, RANDOM traffic.

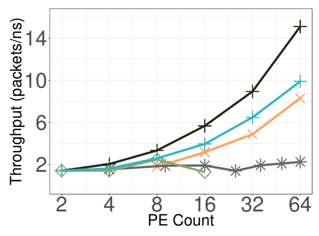
In Figure 10a, we investigate the latency distribution trends of the different NoCs for 64 clients with RANDOM traffic and 100% injection rate. The long tail distribution of deflection-routed Hoplite NoC is clearly evident in this plot. We are able to marginally outperform the buffered CMU CONNECT NoC by 1.1× due to a faster implementation frequency of our design. We are also 1.7× better than a Deflection BFT.

In Figure 10b, we plot average latency (before network saturation) for varying injection rates under RANDOM traffic. As expected, deflection-routed NoCs like Hoplite saturate very quickly at 2% injection rate. The CMU CONNECT NoC delivers much better average latency behavior due to buffers. However, the BFT NoCs are ultimately superior due to richer wiring bandwidth of the upper stages.

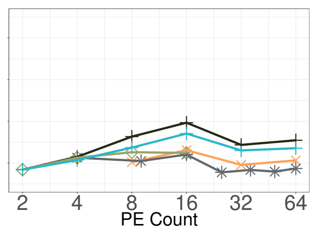
D. Effect of System Size

In Figure 11, we quantify the effect of PE scaling on NoC throughput and worst-case latency for synthetic (RANDOM) and real (geometric mean of various benchmarks) workloads at 100% injection rate. The flow controlled BFT NoC is the clear winner with highest throughputs for both synthetic and real traffic and the lowest worst-case packet routing latencies.

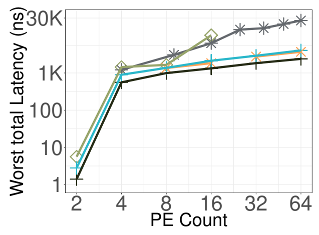
The sustained throughput of our proposed BFT is able to outperform CONNECT by 1.75× and the deflection variant (XBAR) by 1.4× at 64 clients with RANDOM workload. For real workloads the gap is similar at 1.8× (vs CONNECT) and 1.2× (vs deflection variant) suggesting that the wins hold even for realistic traffic patterns. Deflection overheads eventually choke Hoplite and yield low sustained throughputs and



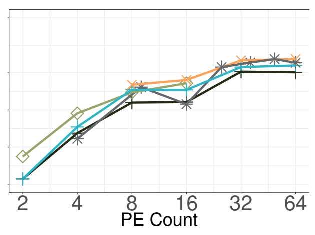
(a) Throughput for `RANDOM`



(b) Throughput for `REAL`



(c) Worst-Case Latency for `RANDOM`



(d) Worst-Case Latency for `REAL`

× CMU + Defl. XBAR ✳ Hoplite + BFT3 ◇ Xlnx IC

Fig. 11. Impact of PE scaling on sustained rate and worst-case latency for `RANDOM` and real workloads `(geomean)`.

longer packet latencies. CMU CONNECT is a buffered NoC and should offer higher throughputs but suffers from lower operating frequencies as observed previously in Figure 8a. The Xilinx AXI4-Stream Interconnect does not scale beyond 16 clients, and suffers from high contention due to single-stage design and lower frequency.

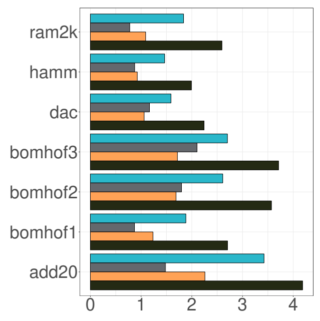
The worst-case latency wins for the BFT under synthetic `RANDOM` traffic are as large at 6× over deflection-prone Hoplite and Xilinx AXI4-Stream interconnect while as larger as 1.5× when compared to deflection XBAR and CMU CONNECT. We see a similar trend for real workloads (`geomean`), where the latency gap is 1.5-2.3× smaller than other NoCs.

E. Routing Realistic Workloads

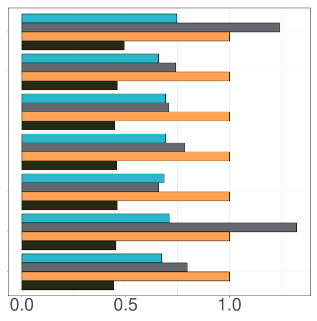
We evaluate 64 client systems running realistic FPGA accelerator workloads and show sustained throughput and worst-case latency in Figure 12. These communication traces for real workloads have 1K–1.5M packets depending on the dataset. We exclude Xilinx AXI4-Stream Interconnect since it does not scale beyond 16 clients.

For Sparse Matrix-Vector Multiplication (SpMV) workload shown in Figure 12a, our BFT NoC is able to outperform CMU CONNECT by 2–2.35×, deflection routed XBAR by nearly 1.2–1.5× and Hoplite by 1.8–3.2×. Graph Analytics workloads, shown in Figure 12c, have a high degree of locality which allows our BFT to outperform CONNECT by 1.25–2×, deflection XBAR by 1.65-2.25×, and Hoplite by 1.7×.

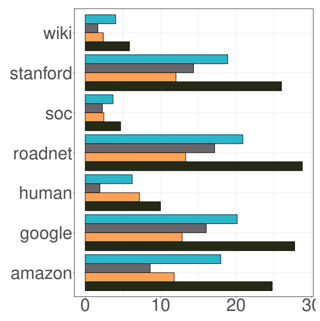
We measure worst-case end-to-end latency (source queueing + in-flight) for routing realistic workloads. The worst-case latency figure is scaled with the CMU CONNECT latency as baseline. Our BFT is able to reduce worst-case latency for SpMV workloads shown in Figure 12b by around 2.3× over CONNECT , 1.5× over Deflection XBAR and 3× over Hoplite. For Graph analytics, shown in Figure 12d, the latency



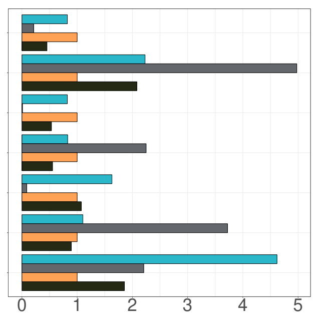
(a) Sustained Throughput (SpMV)



(b) Relative Latency (SpMV)



(c) Sustained Throughput (Graph)



(d) Relative Latency (Graph)

■ Defl. XBAR ■ CMU ■ BFT3 ■ Hoplite

Fig. 12. Throughput-Latency trends for `REAL` traffic.

wins range between 1.1-2× over CONNECT and 1.1-1.7× over Deflection BFT. The `stanford` and `amazon` workload graphs have extremely high locality with very little message communication over the NoC (most messages ≈80% stay in the same client). Hence, the routing latency observed is in the order of a few cycles due to very light NoC injection rate. Hoplite outperforms our BFTs for Graph analytic workloads such as `wiki`, `soc` and `human`, but not on throughput as we saw previously on Figure 12c.

F. Choosing a BFT that is best for you

Constrained by cost: Given a particular resource and power budget, we hope to show how to select a particular BFT topology and its associated bisection bandwidth richness. We consider four configurations with richness varying between a binary tree to a multi-stage crossbar. We evaluate the NoCs under `RANDOM` traffic and 100% injection rate. In both Figure 13a (area) and Figure 13b (power), we see three regions of operation:

- In the first region below 10K LUTs (1 Watt), there is little to distinguish between the variants in terms of throughput, so the BFT0 topology (binary tree) is adequate.
- In the second region, between 10–20K LUTs (1 to 2.5 Watts), the BFT topology that matches the bisection bandwidth of a mesh (BFT2) offers the best result.
- In the extreme case where resource costs are not a factor $>$ 20K LUTs (2.5 Watts), the BFT3 (multi-stage crossbar) is best.

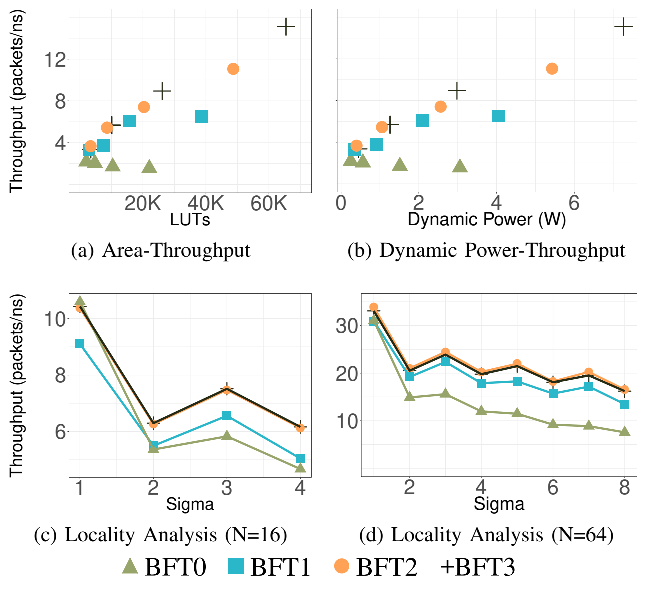


Fig. 13. Selecting the BFT that is right for you. Evaluating FPGA LUT cost and Dynamic Power tradeoffs for different system sizes, RANDOM traffic pattern.

Constrained by application locality: We provide further insight by analyzing the effect of locality on topology selection. For this experiment, we vary the locality parameter SIGMA of our synthetic workloads from 1 (nearest neighbor) to $\sqrt{N}$ (uniform RANDOM), for a topology with system size N. The locality parameter restricts the destination address of a packet to lie within the SIGMA×SIGMA rectangle of the source address. We plot observations for N=16 and N=64 in Figure 13c and Figure 13d respectively. For nearest neighbor style communication, all topologies perform equally well due to little contention among packets. As packets travel larger distances, the degree of contention in the NoC increases and richer topologies like BFT2 and BFT3 perform better. An important observation here is that BFT3 is matched in performance by BFT2, suggesting that the BFT3 (multi-stage) topology is excessive and not needed for real-world locality.

G. Analyzing BFTs with SRL FIFOs

Conventional BFT NoCs [15], [20] use buffers along the links of the tree to hold contending packets. We wish to evaluate the effectiveness of this approach. We have previously quantified the effect of adding 32-deep SRL FIFOs in Table I and concluded that they increase cost by 1.5×.

The number of pipeline stages are doubled every two levels to account for longer wires at the top assuming an H-tree layout. We analyze two pipeline configurations with one and two pipeline registers at the leaf nodes used for seeding the doubling every two levels.

Figure 14a shows sustained throughput (packets/s) as a function of LUT utilization for two pipeline configurations: 1-1-2-2-4-4 and 2-2-4-4-8-8 for a NoC of system size 64. We compare these configuration against our flow-controlled BFT by injecting RANDOM traffic at 100% rate. We observe that the buffered BFTs offer higher throughput

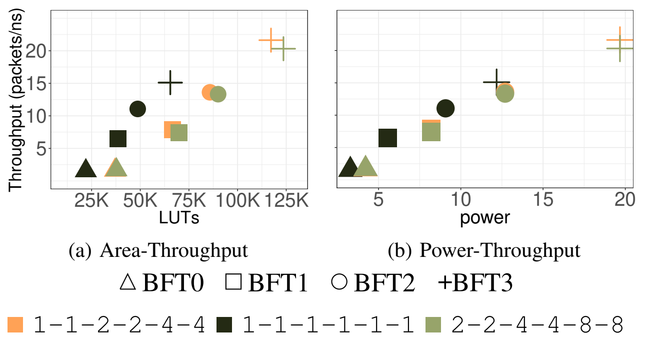


Fig. 14. FPGA LUT cost and Dynamic Power tradeoffs for different NoC configurations. 64 clients, RANDOM traffic at 100% injection rate

but at a disproportionate increase in cost. For instance, we can slightly exceed the throughput of a Buffered BFT2 design with a richer flow-controlled BFT3 design while needing fewer LUT resources. This allows us to tradeoff wiring bandwidth for improved performance. FPGA developers who wish to further boost performance of the BFT3 design can certainly add buffering to do so. Another observation here is that over-pipelining the links hurts rather than helping the NoC, with the 1-1-2-2-4-4 design consuming fewer LUTs and sustaining higher throughput.

Figure 14b captures the power-throughput tradeoffs for the same NoC configurations. Again, we observe that the buffered NoCs consume more dynamic power due to the increased activity of the FIFO resources and associated link pipelines. The buffered BFT2 is again delivering lower throughput while consuming more power than the flow-controlled BFT3 design.

Thus, wire-rich flow-controlled BFTs offer a better alternative to buffered BFT NoCs when considering resource costs as well as power consumption. It remains to be seen whether these conclusions will apply to the Intel Stratix-10 Hyperflex register fabric. It is a different cost model than the LUT-oriented pipelining implementations explored here, and will be studies as part of future work.

V. Conclusions

We show how to construct FPGA-friendly BFT NoCs with lightweight flow control to deliver improved throughput, and latency outcomes while adding in-order delivery and bounded delivery on routing time. In contrast to contemporary FPGA NoCs like Hoplite and CONNECT which either sacrifice NoC features or NoC implementation cost, the BFT is able to deliver both and offers bandwidth configurability to the FPGA developer. We augment the BFT topology with a lightweight latency insensitive interface, deterministic routing policy, and round-robing arbitration to deliver the desired features. Across a range of synthetic and realistic workloads, our topology outperforms others by 1-6×.

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

References

[1] M. S. Abdelfattah and V. Betz. Networks-on-chip for fpgas: Hard, soft or mixed? *ACM Trans. Reconfigurable Technol. Syst.*, 7(3):20:1–20:22, Sept. 2014.

[2] A. Andreyev. Introducing data center fabric, the next-generation facebook data center network. https://code.facebook.com/posts/360346274145943/, 2014. [Online].

[3] A. Bouhraoua and M. Elrabaa. An efficient network-on-chip architecture based on the fat-tree (ft) topology. In *Microelectronics, 2006. ICM'06. International Conference on*, pages 28–31. IEEE, 2006.

[4] E. Caspi. *Design Automation for Streaming Systems*. PhD thesis, University of California, Berkley, 2005.

[5] A. M. Caulfield et al. Configurable clouds. *IEEE Micro*, 37(3), 2017.

[6] A. DeHon. Compact, multilayer layout for butterfly fat-tree. In *Proceedings of the Twelfth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '00, pages 206–215, New York, NY, USA, 2000. ACM.

[7] C. Fallin, C. Craik, and O. Mutlu. Chipper: A low-complexity bufferless deflection router. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 144–155, Feb 2011.

[8] C. Gomez, F. Gilabert, M. E. Gomez, P. Lopez, and J. Duato. Deterministic versus adaptive routing in fat-trees. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, March 2007.

[9] C. Grecu, P. P. Pande, A. Ivanov, and R. Saleh. Structured interconnect architecture: A solution for the non-scalability of bus-based socs. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, GLSVLSI '04, pages 192–195, New York, NY, USA, 2004. ACM.

[10] H. Gu, J. Xu, and W. Zhang. A low-power fat tree-based optical network-on-chip for multiprocessor system-on-chip. In *2009 Design, Automation Test in Europe Conference Exhibition*, pages 3–8, April 2009.

[11] Y. Huan and A. DeHon. FPGA optimized packet-switched NoC using split and merge primitives. In *Field-Programmable Technology*, pages 47–52, Dec. 2012.

[12] N. Kapre. Deflection-routed butterfly fat trees on fpgas. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, Sept 2017.

[13] N. Kapre and J. Gray. Hoplite: Building austere overlay nocs for fpgas. In *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, Sept 2015.

[14] N. Kapre and T. Krishna. Fasttrack: leveraging heterogeneous fpga wires to design low-cost high-performance soft nocs. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pages 739–751. IEEE Press, 2018.

[15] 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 *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 205–216, April 2006.

[16] B. S. Landman and R. L. Russo. On a Pin Versus Block Relationship For Partitions of Logic Graphs. *Computers, IEEE Transactions on*, (12):1469–1479, 1971.

[17] B. Lebiednik, A. Mangal, and N. Tiwari. A survey and evaluation of data center network topologies. *arXiv preprint arXiv:1605.01701*, 2016.

[18] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

[19] J. Mische, C. Mellwig, A. Stegmeier, M. Frieb, and T. Ungerer. Minimally buffered deflection routing with in-order delivery in a torus. In *2017 Eleventh IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8, Oct 2017.

[20] 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*, pages 37–46. ACM, 2012.

[21] D. Park, Y. Xiao, N. Magnezi, and A. Dehon. Case for fast fpga compilation using partial reconfiguration. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, Sept 2018.

[22] S. Scott, D. Abts, J. Kim, and W. J. Dally. The blackwidow high-radix clos network. In *33rd International Symposium on Computer Architecture (ISCA'06)*, pages 16–28, June 2006.

[23] 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.*, 45(4):183–197, Aug. 2015.

[24] 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*, 63(6):1446–1459, 2014.

[25] M. Wissolik, D. Zacher, A. Torza, and B. Da. Virtex ultrascale+ hbm fpga: A revolutionary increase in memory performance. *Xilinx Whitepaper*, 2017.