Deflection-Routed Butterfly Fat Trees on FPGAs

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

***Abstract*—**
Bufferless, deflection-routed, Butterfly Fat Trees (BFTs) can outperform state-of-the-art FPGAs overlay NoCs such as Hoplite by as much as 2–5× on throughput and ≈5× on worst-case latency at identical PE counts, and by ≈1.5× on throughput at identical resource costs >16K LUTs for statistical traffic patterns. In this paper, we show how to modify the tree connectivity and routing function to support deflection routing on the BFT topology. We introduce the idea of *localized deflections* that trap deflected packets within a single level of a multi-level BFT to avoid the long round-trip penalty traditionally associated with deflection routing. Across a range of statistical traffic patterns, we show a sustained throughput improvement of 2–5× over for Hoplite for system sizes as large at 512 PEs at above 20% injection rates when using localized deflections. We also show how the configurable bisection bandwidth of the BFT, modeled with the Rent parameter 0<p<1, allows us to choose the best performing NoC at a desired cost. For instance, our NoC generator can produce simple trees (p=0) for low-cost applications <2K LUTs, mesh-equivalent BFTs (p=0.5) for real-world applications with locality at <10K LUTs, and crossbars (p=1) when cost is not a constraint >64K LUTs. For workloads with locality, we recommend the BFT topology with $p = 0.67$.

I. Introduction

The exponential growth in transistor capacity has made it possible to make FPGAs with hundreds of thousands of LUTs and FFs, thousands of DSP and RAM blocks, hundreds of IO ports all supported by a statically configured, rich interconnect network. One way to tame the rising capacity of the FPGA chip is to adopt a modular design approach with standardized communication interfaces such as AXI (part of ARM's AMBA Advanced Microcontroller Bus Architecture specification). This allows the FPGA developer to compose large FPGA designs by stitching together modules using these standard interfaces and a flexible communication fabric for data movement between the modules. Furthermore, modern FPGA are able to interface with a wide variety of IO protocols and interface specifications such as PCI Express, multi-channel DRAM busses, gigabit Ethernet ports, among others. Efficient movement of data from these system-level interfaces to various portions of the chip is also a challenge. FPGA-based overlay NoCs can solve both these problems. Communication over a NoC can be switched allowing the design to share communication infrastructure in different phases of the FPGA execution.

Typical real-world designs exhibit traffic patterns with locality that can be explained using Rent's rule. Rent's rule [1] (IO=cN^p) is an empirical observation of communication growth in a design hierarchy. For real-world designs, communication requirements tend to decrease as we ascend the hierarchy *i.e.* traffic tends to be concentrated in local subregions. Most contemporary FPGA-based NoCs [2], [3] built on 2D mesh and torus topologies do not directly exploit this observation. For instance, the CMU Connect Mesh [2], Penn Split-Merge Mesh [3], use network topologies with a Rent parameter p=0.5. These structures may match the rectangular FPGA organization but do not reflect communication requirements of FPGA applications. In contrast, the CMU Connect [2] Fat Tree and Butterfly Fat Tree (BFT) [4] use fat-tree-based topology [5] that is better matched to communication locality observed in real-world communication workloads. They can outperform their mesh counterparts by 1.5–2× [6] but do not reduce FPGA implementation costs significantly.

With the advent of the Hoplite [7], it is possible to implement FPGA overlay NoCs with very low cost (60 LUTs + 100 FFs for a 32b router) that is up to 25–30× smaller than CONNECT and Split Merge NoCs. Under this new reality, we investigate whether BFT topologies still remain competitive against 2D topologies. In this paper, we adapt the BFT routing function to support bufferless deflection routing in a manner inspired by Hoplite. A naive formulation of the routing function to exploit bufferless deflections does not deliver the full benefits against Hoplite. We redesign the routing function to support localized deflections within each level of the BFT to lower switch costs without sacrificing performance. In fact, we show how to *exceed* the performance and area efficiency of the Hoplite NoC using our design for large system sizes.

We make the following key contributions in this paper:

- We formulate a bufferless deflection routing function for BFTs that lowers FPGA implementation cost of switches.
- We develop the idea of *local deflections*, which lowers the penalty of deflected packets at the expense of extra LUTs. We quantify the area-performance tradeoffs due to this modification of the switch architecture.
- We characterize the throughput, latency trends of the NoC under various traffic patterns and demonstrate its superiority over contemporary state-of-the-art NoCs like Hoplite.

II. Butterfly Fat Trees on FPGAs

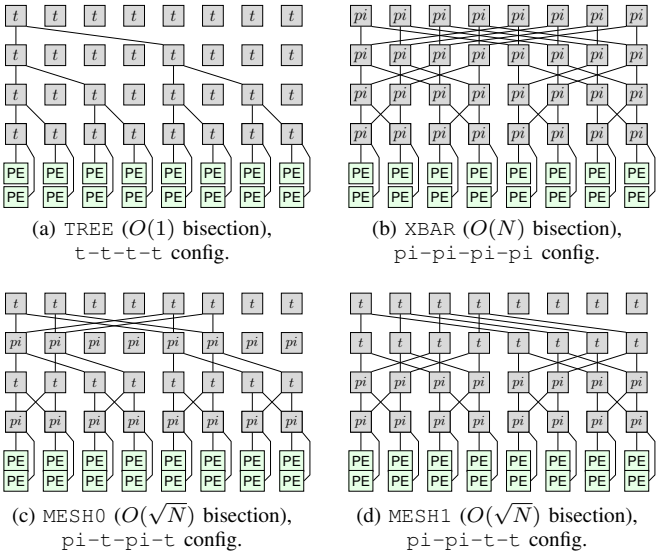
A Butterfly Fat Tree [5] (BFT) is a multi-level switching topology with statically configured bisection bandwidth that can be adapted for engineering efficient FPGA NoCs. Bisection bandwidth represents the ability of the NoC to transport traffic across a cut along the middle of the NoC topology. To capture varying bandwidth needs, a BFT is constructed using

two kinds of switches with different routing capabilities: a t switch (2 in:1 out with half the outgoing bandwidth) and a pi switch (2 in:2 out with identical incoming and outgoing bandwidth). All switches in a given level are of the same kind, but each levels can be independently configured as either t or pi types. For this paper, we consider arity-2 (2-input) switches but higher arity networks are possible. Thus, to support N processing elements (PEs) at the leaves, we need a BFT with $\log_2 N$ levels. For instance, the various NoC topologies shown in Figure 1 represent the range of network configurations that can be generated using the BFT template by simply adjusting the switching structure in the NoC levels.

The Rent parameter for the application is computed by recursively bisecting the application and measuring the communication required (data transfers in the form of messages, streams, packets) in the various partitions. Similarly, by recursively partitioning the NoC, we can measure bisection bandwidths of wires at various local partitions to compute the Rent parameter of the NoC architecture. A balanced NoC will match the bisection bandwidth in the physical topology of the chip to the communication requirements in the application. An ordinary binary tree (Figure 1a) has a bisection bandwidth of $O(1)$, and a Rent parameter p=0, making is equivalent to a simple ring. This topology will support the application with little communication requirements, or nearest-neighbour style systolic communication. A crossbar (Figure 1b) has a bisection bandwidth of $O(N)$, and a Rent parameter p=1, which allows all-to-all communication between the application partitions. This means that in a given cycle, a non-conflicting permutation of N messages can be delivered to their destinations. Since the BFT is a multi-stage network, the number of switches required to support this is $O(N \log_2 N)$ instead of $O(N^2)$ at the expense of extra $\log_2 N$ switch hops. For most real-world applications this is overprovisioned and expensive. An ordinary mesh (Figure 1c) has a bisection of $O(\sqrt{N})$, and a Rent Parameter p=0.5, due to the rectangular $\sqrt{N} \times \sqrt{N}$ layout of the NoC. A BFT can be setup to support all of these topologies by careful configuration of the switch connectivity in each layer.

We tabulate the various BFT NoC topologies for a 16-PE (Processing Element) design in Table I. We can deliver a crossbar by only using pi switches at all levels in the NoC (32 pi switches), and a binary tree by only selecting t switches (16 t switches). In addition to these, we can also configure a topology equivalent to a Mesh in bisection bandwidth in two ways shown in Figure 1c (alternating t and pi switches for a total cost of 12 t and 12 pi switches) or Figure 1d (pi switches in lower levels, and t switches in upper levels for a total cost of 12 t switches and 16 pi switches). This preserves the top-level bisection bandwidth at $O(\sqrt{N})$ but reallocates bandwidth to the lower levels at the expense of more pi switches. Depending on application requirements and cost constraints, we can target other bandwidth configurations.

Traditional designs for FPGA-based overlay NoCs have focussed on rectangular layout-friendly topologies such as Meshes [2] and Tori [7]. These offer a simpler mapping to rectangular 2D fabrics, but do not provide a cost-effective way to scale bandwidth. The only way to add more bandwidth to such fabrics at a fixed system size N is to add parallel channels c. Increasing channel count uniformly adds bandwidth to all segments of the NoC, often where it may not be needed. This is particularly the case when the NoC link width is configured to match the system-level interface widths of PCIe, DRAM, or Ethernet IPs. Instead, we can use the configurable structure of the BFT topology to provide wider links in the top-level of the NoC and use these to split and distribute traffic to leaf-level compute blocks. Additionally, we can configure each level of the BFT to match user's communication demands.



(a) `TREE` ($O(1)$ bisection), `t-t-t-t` config.
(b) `XBAR` ($O(N)$ bisection), `pi-pi-pi-pi` config.
(c) `MESH0` ($O(\sqrt{N})$ bisection), `pi-t-pi-t` config.
(d) `MESH1` ($O(\sqrt{N})$ bisection), `pi-pi-t-t` config.

Fig. 1: 16-PE Butterfly Fat Tree Topologies.

III. Deflection Routed BFTs on FPGAs

In this section, we describe the FPGA design of the switching elements that compose a BFT NoC and discuss the implementation of deflection routing in the switch. For our bufferless, deflection-routed scenario, we consider single-flit packets carrying address and payload in a single wide packet.

A. Routing in BFTs

Routing packets on a switched BFT is simple. Unlike mesh and torus topologies, where we must supply X and Y addresses, a BFT only requires a single numeric identifier for address. As the packet climbs the tree, it has complete freedom to choose the ascending path irrespective of the destination address. The packet must *turn* at the height defined by the sub-tree that contains both the source and destination addresses. This is trivially calculated as the length of the common prefix shared between the switch and destination addresses. For t

TABLE I: BFT Topology Configurations for 16-PE design.

Topology	Rent p	Switch Config.	Num. of S/W		Bisection B/W
			t	pi	
`TREE`	0	`t-t-t-t`	15	0	1
`MESH0`	0.5	`pi-t-pi-t`	12	12	4
`MESH1`	0.5	`pi-pi-t-t`	12	16	4
`XBAR`	1	`pi-pi-pi-pi`	0	32	16

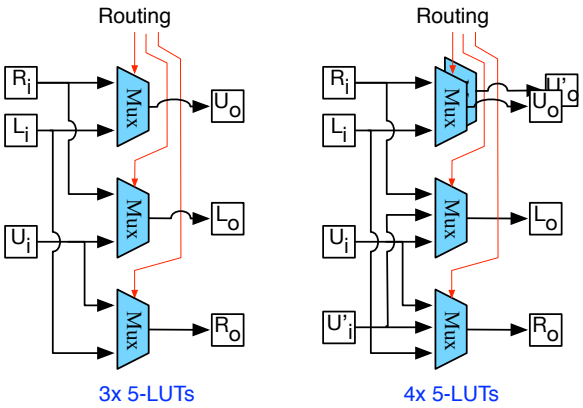


Fig. 2: Xilinx LUT mapping for t and pi switches using **Root Deflections** approach.

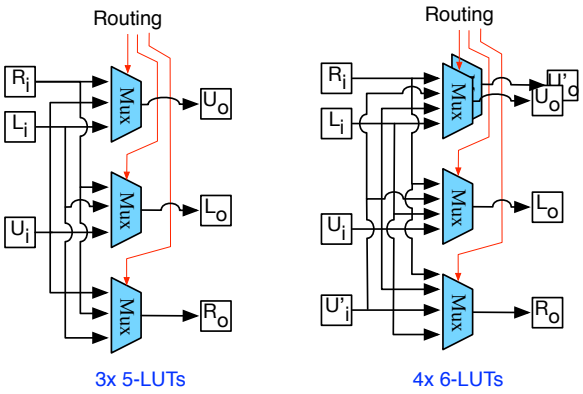


Fig. 3: Xilinx LUT mapping for t and pi switches using **Local Deflections** approach.

switches, there is only a single upward-bound port, and for pi switches we have a choice of two ports sending packets up the tree. Thus, as we climb multiple levels of pi switches, we get exponentially more choices on the uphill paths. The descending path is strictly defined by the destination address and no path freedom is available. The routing decision is made by simply extracting the ith bit corresponding to the ith level in the fat tree. This makes the routing function simpler than the mesh and can be implemented with effectively a single FF for uphill ports (toggles for fair distribution of bandwidth), and no LUTs/FFs for downhill decisions (directly use the ith wire as multiplexer select).

B. FPGA Mapping of Switch Multiplexers

In Figure 2, we show how to implement the switching crossbar in the BFT t and pi switches on a Xilinx FPGA. As we can see the t switch has three inputs (R, L from lower level, U from top) and three outputs (R, L to lower level, and U to top). This can be compactly fit in 3× 5-LUTs as each output port needs a simple 2-input multiplexer element. The pi switch, in contrast, has two ports going up. The switching for packets bound upwards is a simple choices between the R and L inputs from the lower level which can be mapped to 2-input multiplexer elements. However, downward bound packets can either be packets that are turning from the R or L inputs, of descending packets arriving at the U or U' inputs. This requires a 3-input multiplexer for the R and L output ports. For the t switch we can map the design to fractured Xilinx 5-LUTs. For the pi switch, only the upward-bound multiplexers can be efficiently mapped to fractured 5-LUTs, while the other two require 6-LUTs. In presence of contention, the traditional BFT implementation [4] buffers packets at the inputs. Our bufferless deflection router will send packets to an available output port in presence of contention.

C. Deflection Routing in BFTs

Deflection Routing is a long-studied topic in NoC design and is most popular in mesh [8] and torus [7] topologies. We show how to apply this idea to the BFT topology with suitable adaptations to the routing function and wiring of the NoC.

Root Deflections: The simplest implementation of deflection routing can be built using the switching structure shown in Figure 2. Instead of buffering packets, the deflection function simply routes the packet along the next available port. For upward-bound packets in a pi switch, this is naturally possible as part of the BFT routing function. For pi switches, since there are two ports going UP, it is sufficient to take either of those exits without compromising performance. In other cases, such as a t switch with a upward-bound packet, or any downward-bound packet in either t or pi switches, we have to deflect along an undesirable port. For upward-bound packets in a t switch, this will deflect a packet downwards prematurely, while for downward-bound packets, packets may be sent down the wrong sub-tree. To ensure packets get delivered to their destinations eventually we need physical loopback connection at (1) the **root** of the BFT tree, and (2) at the PE interfaces. In comparison, for a deflection-routed NoC like Hoplite, the loopback connections are naturally part of the torus topology. Our BFT approach is simple to implement and requires no modification of the t and pi switches apart from adding deflection rules in the routing function. However, we do need to sacrifice bandwidth at the root (the top-most level) of the BFT to implement the loopback for deflected packets as well as some storage costs at the PE interfaces to turn back deflected packets. We show FPGA costs in Table II.

Local Deflections: A different approach towards supporting deflections is to localize them within a level of the BFT with loopback. The switch structure to support this idea is shown in Figure 3. In this case, the deflected packed is bounced back along the direction of arrival at the contention switch. At the point of contention, the deflected packet will return back in the next cycle after turning around at the next switch. The routing function is modified to prefer deflected packets for turn-back. This also requires inserting a return path in the switch multiplexers but needs no extra wiring between the BFT levels. For the t switch this upgrades the 2-input multiplexers to 3-input ones while for the pi switch, we now need 4-input multiplexers. This increases the LUT-mapping cost of the t and pi switches (See Table II) due to non-fracturability of the multiplexers. However, we no longer require the top-level BFT

wiring to be sacrificed for loopbacks, but can instead be used to inject system-level traffic from high-bandwidth interfaces such as PCIe, DRAM, and Ethernet ports for distribution across the chip. Furthermore, as we show later in Section V, when compare to cheap Hoplite switches (which are equivalent in cost to the Root t switch), this overhead is recovered through lowered deflection penalties and consequently better throughputs at large system sizes.

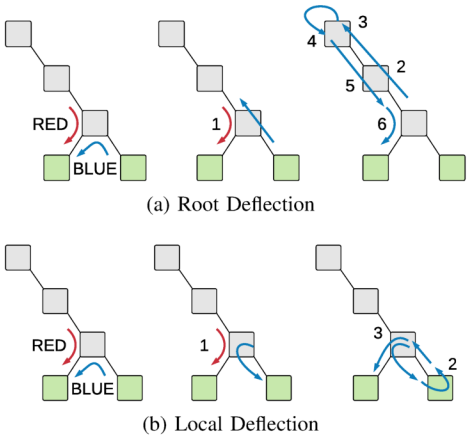
TABLE II: FPGA Costs of BFT Switches.

Router	LUTs	FFs	Clock (MHz)	Power (W)
Root t switch	59	109	500	0.247
Root pi switch	122	145	470	0.252
Local t switch	141	113	597	0.252
Local pi switch	218	150	430	0.255

We can better understand how the deflections work through a simple conflict example shown in Figure 4. Let us assume that a conflict arises, as shown for the L exit port in the lowermost switch, with both the blue and red packets trying to use this port, we have to deflect. For the **Root Deflection** design, the packet must traverse all the way to the root of the tree before its deflected down. It then gets a second attempt to take the correct path to its destination. Recall, the t and pi switches here do not have the internal multiplexer path to support support turning back at arbitrary level of the tree. As all deflected packets will reach the top before deflecting down, there's a system-wide impact of the bandwidth loss due to each deflected packet. In contrast, for the **Local Deflection** design, packets can turn back on the direction of arrival and reattempt routing along the correct path in the next cycle. This is possible because each multiplexer now supports an extra input requiring more FPGA LUTs. In this case, the packet only needs to traverse one extra level up the tree before turning back. Additionally, the bandwidth impact of the deflection is restricted primarily to the subtree being affected. Thus, the **Local Deflection** design offers lower latency packet delivery, and reduced impact of deflection on system-wide bandwidth when compared to the **Root Deflection** at the expense of extra hardware.

IV. Methodology

We describe the BFT NoC architecture using parametric RTL that can support generation of arbitrary-sized NoC topologies. In this paper we focus on system sizes between 2 and 512 PEs. Our RTL is flexible and can support any interleaving of t and pi switches. While it is possible to generate an 2^k possible configurations for BFTs with k levels, we only evaluate the four BFT configurations shown earlier in Figure 1 and Table I. The `TREE` configuration is built simply with t switches, while `XBAR` configuration uses only pi switches in all levels. These two NoCs represent the two extreme bandwidth scenarios. We also consider mesh-like topologies `MESH0` (alternating t and pi switches) and `MESH1` (pi-pi-t-t repeating pattern) with identical bisections



(a) Root Deflection

(b) Local Deflection

Fig. 4: Example of conflict at switch and deflection route. Grey boxes=switches, green boxes=PEs. Red and blue packets conflict on the L exit of the lowermost switch. In the **Root Deflection** case, the packets must traverse all the way to the top before turning back. For **Local Deflection** packet will turn in a single level of the BFT.

but different switch counts. These configurations allow us to compare performance with 2D FPGA overlay NoCs.

Simulation: We run extensive cycle-accurate Verilog simulations of different NoC configurations (number of PEs N, injection rates, traffic pattern, BFT structure) using `iverilog`. For simulation, we connect the NoC to PEs that implement NoC traffic under various synthetic patterns and programmable injection rates. We measure sustained rates (bandwidth), worst case latencies and source queueing delays. We evaluate various traffic patterns such as `RANDOM`, `LOCAL`, `BITREV`, and `TORNADO`. We modify these patterns to work with the linear BFT addressing where a single index is used for routing (unlike mesh and torus with X and Y fields are needed). The `TRANSPOSE` pattern, where X and Y fields are swapped, does not apply to the linear BFT PE arrangement.

Synthesis and Place and Route: We compile the various BFT RTL configurations using Vivado 2016.4 FPGA CAD tool targeting the XC7V485T FPGA. For synthesis, we use a dummy PE that avoids logic pruning during compilation. We assume the payload width for our data to be 32b with a variable number of address bits $\log N$.

When reporting resource costs of the NoCs, we consider LUTs and Wirelength. LUT costs are in terms of 6-LUTs used by the design as reported after Place and Route. For a BFT with N nodes, we consider the lowermost level of the NoC to have unit length wires. For upper stages of the network, we consider doubling the wirelength at each level. Thus the total wirelength can be expressed as $\sum_{i=0}^{\log N-1} \sum_{j=0}^{num\_sw[i]} (2^{i+1} \times type[i]?(2 : 1) + 2^i \times 2) \times B$ where B is the portwidth which is set to 32b for our designs. If the switch type at level i is a pi switch, we account for the cost of four output ports, while a ti switch would be charged the cost of three output ports. Upward-bound outputs pay twice the wirelength cost of downward-bound outputs. The sequence of t or pi configuration is determined by the Rent parameter p. This

also determines the number of switches $num\_sw[i]$ at a given level i. When comparing with 2D $\sqrt{N} \times \sqrt{N}$ NoC layouts for Hoplite, we assume two unit length wires between consecutive routers with an interleaved folded layout for high performance. Here the total cost of the wires is $N \times 2 \times 2 \times B$ with 2 output ports, and a length of 2 wire accounted for each port (again B=32b ports).

V. Results

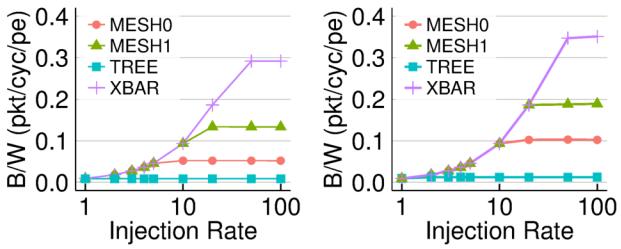
In this section, we evaluate the NoC metrics of the BFT topology such as throughput, latency, resource costs to understand when to use these architectures in a user application. In particular, we investigate the effectiveness of the Local approach vs. Root Deflections, and compare our NoC against the state-of-the-art Hoplite torus NoC. We consider uniform `RANDOM` and `LOCAL` traffic pattern (with locality diameter of 2 nodes).

A. Throughput

We first quantify the sustained throughputs achieved by the NoC under various injection rates for `RANDOM` traffic in Figure 5 at a system size of 256 PEs. We observe a higher throughput (gap of 10×) for the richer networks such as Crossbars over vanilla Binary Trees. This is expected as these richer networks have higher bisection bandwidths. We also observe a 30% better throughput for `MESH1` over `MESH0` even through both networks have identical bisection bandwidths. This was due to more switching capacity and richer communication bandwidth in the lower level networks for `MESH1`. As we will see later in Figure 12a, this represents a 2–5× improvement in sustained rates over Hoplite NoC without considering FPGA resource costs. When comparing the effect of supporting local loopbacks (Figure 5b) instead of root deflections (Figure 5a), we observe a 20–50% improvement in performance with the higher wins evident in the less bandwidth rich topologies. Further throughput wins of ≈30% are evident when `LOCAL` traffic pattern is routed on the tree as seen in Figure 6. For local loopback with `LOCAL` traffic, observed in Figure 6b, we see a drop in performance at 100% injection rates due to congestion caused by packets crossing sub-tree boundaries. In this scenario, locality extends just beyond the sub-tree and due to 100% injection rates, there are no empty slots for crossing over.

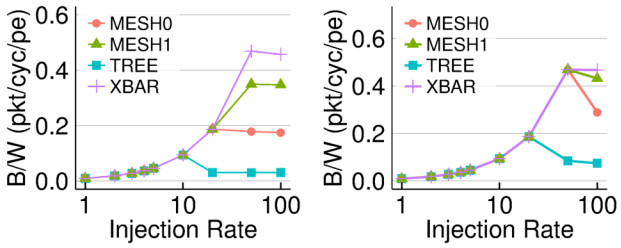
B. Impact of Loopback

Next, we investigate the effect of providing local loopback over root-based deflections in greater detail. In Figure 7, we compare the two alternatives for `RANDOM` traffic with a 50% injection rate (highly loaded NoC). At low injection rates below 15%, there is not much difference between the networks (Figure 5), and a designer should simply pick the cheapest NoC. The reader may recall, that the local loopback avoids long roundtrips to the root of the tree for deflections at additional expense. We observe that local loopback helps scalability by allowing sustained rates to be 20–40% better than the root deflection design. For both topologies, local



(a) Root Deflection (b) Local Deflection

Fig. 5: Sustained Rate trends for 256 PEs under `RANDOM` workload for various BFT NoC configurations.

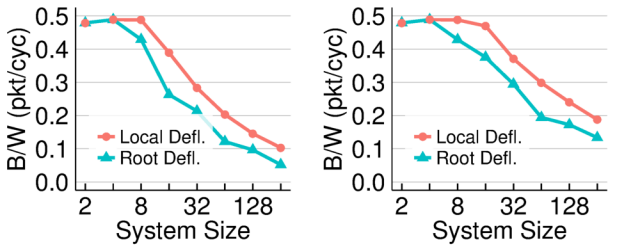


(a) Root Deflection (b) Local Deflection

Fig. 6: Sustained Rate trends for 256 PEs under `LOCAL` workload for various BFT NoC configurations.

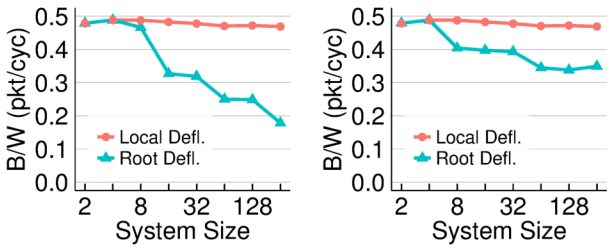
loopback starts to deliver better scalability for designs as small as 4 PEs. It is also clear that the `MESH1` topology offers better scalability than a `MESH0` NoC. When considering `LOCAL` traffic pattern with 50% injection rate and local deflections, illustrated in Figure 8, we observe that the sustained rates stay remarkably high and show very little degradation as system size is increased. This provides a strong evidence that (1) local traffic patterns map particularly well to tree-structured hierarchical networks, and (2) localizing deflections helps sustain the throughput trends.

We now plot the latency trends in Figure 9 with separate measurements for worst-case latency, and average source



(a) `MESH0` Topology (`pi-t-pi-t`) (b) `MESH1` Topology (`pi-pi-t-t`)

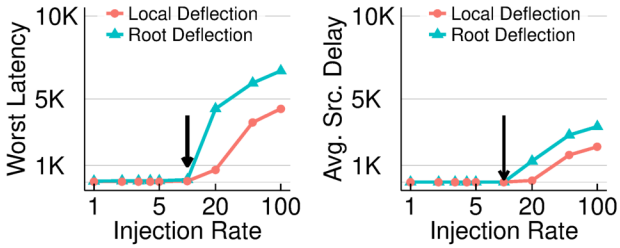
Fig. 7: Sustained Rate trends for various PE counts for the Mesh-like BFT topologies, `RANDOM` + 50% injection.



(a) MESH0 Topology (pi-t-pi-t)

(b) MESH1 Topology (pi-pi-t-t)

Fig. 8: Sustained Rate trends for various PE counts for the Mesh-like BFT topologies, LOCAL + 50% injection.

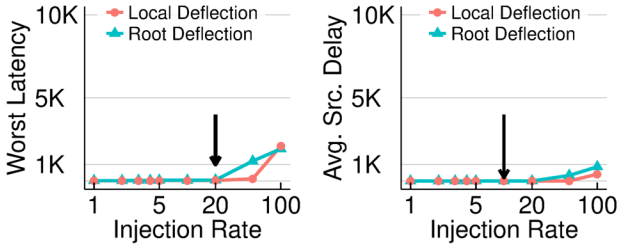


(a) Worst Case Latency

(b) Source Queueing Delay

Fig. 9: Latency breakdown for 256 PEs MESH1 BFT topology, RANDOM workload.

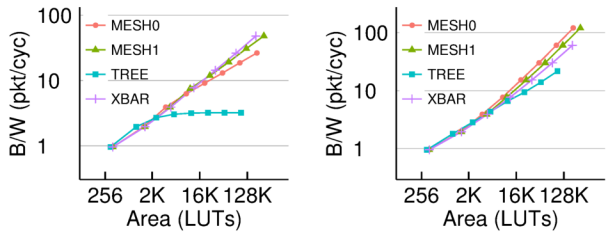
queueing delay (already included in worst-case latency) in the PE. As shown in Figure 9a, for the MESH1 topology with 256 PEs, we observe a performance advantage for local loopback design opening up above 10% injection rate for RANDOM workload. In this case, we see 50–60% higher worst case latency when using root deflections. We also see a consistency larger source queueing delays for the root deflection design (Figure 9b). We should expect this as the root deflection design can deflect packets in either direction (upwards, or downwards). Each packet deflected downwards blocks the PE from injecting its own packet and instead deflecting this packet back into the network. The source queueing delay can account



(a) Worst Case Latency

(b) Source Queueing Delay

Fig. 10: Latency breakdown for 256 PEs MESH1 BFT topology, LOCAL workload.



(a) RANDOM traffic

(b) LOCAL traffic

Fig. 11: Area-Throughput Tradeoffs across various BFT Topologies, 50% injection rate.

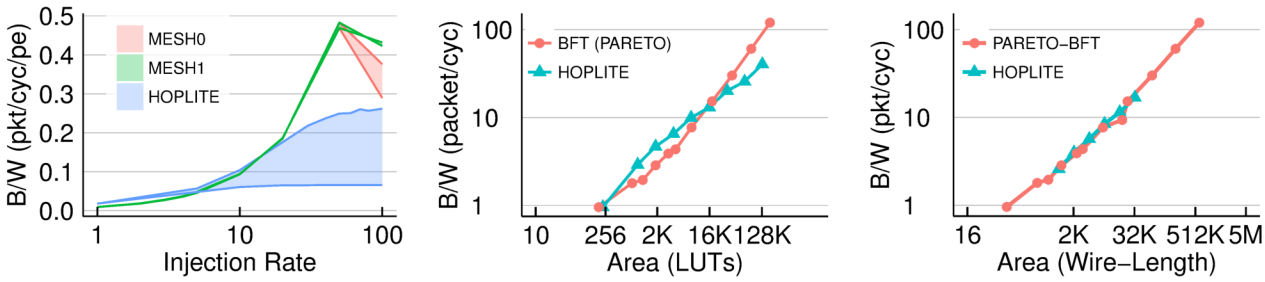
for >2× the total packet latency (3–5K average queueing delay vs. 4–7K worst case latencies at 100% injection rate). The results for LOCAL traffic patterns, shown in Figure 10, are again improved significantly.

C. Impact of Resource Costs

On a typical NoC topology, if an application wants more bandwidth, it can trade-off area for increased performance. In Figure 11a, we show the effect of two scenarios for 50% injection rate and RANDOM traffic with variable system size $2<N<512$. Here, we show how an developer may select a NoC based on cost/performance tradeoffs along with a selection crossovers marked with arrows. If a developer wants to route RANDOM traffic and can only afford to spend <2K LUTs for a NoC (<8 PEs), a TREE NoC is the best choice. The MESH0 is the best NoC at >2K and <10K LUTs (<8 PEs). At >10K LUTs, the richer MESH1 topology offers the best choices. The XBAR topology only makes sense if you have >64K LUTs (>64 PEs), and even then the performance advantage over a MESH1 NoC is marginal (1.1×). If the developer has an application with spatial locality, which is more likely, they should prefer MESH0 topology. In Figure 11b, we see conclusive evidence that richer topologies like MESH1 and XBAR simply occupy resources without delivering any performance improvements for LOCAL traffic pattern. The Figure 11 also highlights the danger of drawing conclusions exclusively from a uniform RANDOM workload as it leads to overprovisioning of resources in the NoC.

D. Comparing Hoplite

We now compare the cost and throughput trends of the BFT NoC against the state-of-the-art Hoplite FPGA NoC [7]. In Figure 12, we show absolute sustained throughputs, LUT-throughput and Wirelength-Throughput trends for a LOCAL workload. Figure 12a, shows the extent of possible sustained throughputs for the NoC under various injection rates at 16 PEs (lower end), to 256 PEs (higher end) for Hoplite and MESH0 and MESH1 BFT configurations that mimic the bisection bandwidth of the Mesh. Here the throughput advantage is 2–5× in favour of the BFTs at identical PE counts. For <8K LUTs, the Hoplite NoC offers better throughput/area than the BFT by 1.6× but at larger sizes >16K LUTs, the BFT offers



(a) Absolute Throughput between 16–256 PEs (comparisons show by arrows).

(b) LUT-Throughput 50% injection rate.

(c) Wirelength-Throughput (arbitrary units) 50% injection rate.

Fig. 12: Throughput Tradeoffs between the BFT NoCs and Hoplite. LOCAL workload.

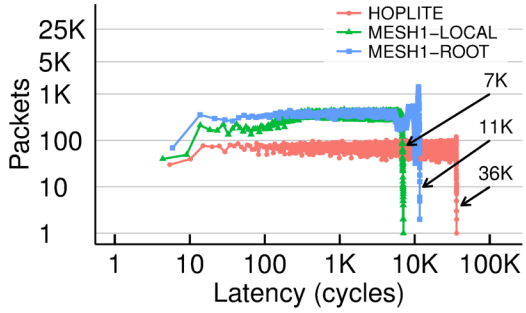
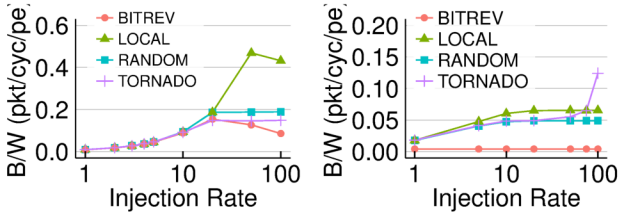


Fig. 13: Latency distribution for Hoplite vs. BFT MESH1 NoC for 256 PEs and RANDOM traffic+50% injection rate.



(a) BFT MESH1 NoC.

(b) Hoplite NoC.

Fig. 14: Sustained Throughputs across various NoC traffic patterns for 256 PEs (note y-axis extents).

superior throughput by factors as much as 1.5×. Here, we only show the *pareto-optimal* design points extracted from Figure 11b. These trends suggest the higher cost of the t and pi switches may deliver better absolute throughput (packets/cycle/PE), but the low cost implementation of the Hoplite NoC using fractured Xilinx 6-LUTs remains a formidable challenge at small system sizes. BFT-based topologies show better scalability at large system sizes above 16K LUTs. Not shown on the plot is the reduced FF cost of the BFT NoCs compared to the Hoplite NoC that closes the gap down to 1.3–1.5×.

We also plot the packet latency distribution for the competing NoCs in Figure 13. In this experiment, we route RANDOM traffic with 50% injection rate and 2K packets injected by each of the 256 PEs (512K packets total). We see that the Hoplite NoC has 3.2× larger worst-case latency than even the Root Deflection BFT MESH1 NoC and ≈5× higher latency over the Local Deflection approach. This result is in agreement with the throughput improvements possible for the BFT-based NoCs over Hoplite at large system sizes.

Finally, in Figure 14, we show the effect of routing various synthetic NoC patterns on Hoplite and the MESH1 NoC topology. Our previous plots focussed on RANDOM pattern as it does not unfairly advantage the BFT topology. In particular, the LOCAL pattern benefits from the hierarchical tree structure of the BFT particularly well and delivers >4× throughput boost compared to Hoplite (at the cost of larger t and pi switches). For other patterns, we see a ≈2× throughput boost across all injection rates for the BFT NoC over Hoplite. The BITREV traffic pattern, which is known to be tough to route, performs particularly poorly on the Hoplite NoC, but manages to scale well on the BFT up to 10% injection rate before suffering from bottlenecks.

VI. Discussion

In this section we discuss the relevance of our work in the context of other FPGA overlay NoCs (See Table III) and also explore a few FPGA implementation scenarios for the BFT.

Comparison with other NoCs: One of earliest FPGA NoC designs that used BFT topology is the decade-old Split-Merge design [4]. The switches in that design were composed of Split and Merge blocks with buffered inputs resulting in high resource cost per switch. Despite this, at the time, these designs were still the fastest and among the cheaper NoC designs of the day. The CMU Connect [2], and an improved Split-Merge architecture [3] are faster, and more suited for the newer FPGA architectures. The recent Hoplite [7] design drastically reduces resource cost while boosting performance of the NoC routers. It is able to do so by using bufferless deflection routing, and a careful LUT-mapping of the multiplexer elements in the switches. Our BFT design applies deflection routing to BFTs FPGA NoCs, and introduces a novel BFT-specific optimization (local loopbacks) to reduce the penalty of deflections. The idea of localizing conflicts in a NoC has been previously explored

in MinBD [9] in the context of Mesh topologies, and at the cost of extra wires, registers, and multiplexers to hold and process the local loopback packet. In contrast, our approach reuses existing links and registers of the BFT topology to route the loopback packets, but increases the cost of multiplexing in the switch in a manner similar to MinBD. In Table III, we show the FPGA router costs of a BFT FPGA design [4] and Hoplite NoC [7]. We easily beat the Split-Merge design mapped to V2-6000 switches by 2–3.5× in area and speed. The Hoplite NoC is 1.4–5× smaller than the largest pi switch, but comparable to the cheapest t switch in our repository (see Table II). The clock speeds of both designs are comparable. As we saw in Section V-D, despite this lower area cost, the Hoplite NoC is beat by the BFT NoCs at larger system sizes. Mesh routers such as the CMU Connect [2] and Penn Mesh-Topology Split-Merge [3] designs have been shown to be 20–30× larger than Hoplite. Thus, the improvements possible with the proposed BFT deflection-routed NoC are additive on top of previously established result [7]. The CMU Connect NoC generator can construct Fat Trees [10], but our design is 2× smaller and faster than their generator even when considering the most expensive pi switch with Local Deflection support. Our BFT generator can produce a rich set of NoCs, in a manner similar to the CONNECT NoC generator, with varying bandwidth capacity and costs while exploiting the configurability of the switching richness at each level of the NoC.

TABLE III: Comparing FPGA Costs of NoC routers.

Router	**Ref**	**FPGA Device**	**LUTs**	**FFs**	**Clock** (MHz)
Split-Merge t	[4]	Virtex-2 6000	486 (3.5×)	224 (2×)	200 (2.7×)
Split-Merge pi	[4]	Virtex-2 6000	820 (2.6×)	576 (3.8×)	200 (2.1×)
Hoplite	[7]	Virtex-6 LX240T	60 (0.2×)	100 (0.7×)	350 (1.2×)
CONNECT Fat Tree	[10]	Virtex-6 LX760	450[1] (2×)	N/A[2]	203 (2.1×)

[1]450 LUTs/router calculated from Table 1 of [10] – 20 routers in Fat Tree with 16 PEs, 32b links, 1.9% of LX760 used, LX760 has 474K LUTs.
[2]No FFs reported in [10].

Usage Scenarios: We consider two scenarios for interfacing the NoC with system-level traffic sources on the FPGA. In the first case, we can replace a few leaf PEs with PCIe, DRAM, Ethernet, or other forms of high-bandwidth system-level communication. We may need to replace multiple PEs to ensure all bandwidth from these interfaces can enter the NoC for chip-wide distribution. For the Root Deflection design, this is the only way to interface system-level traffic as the top-most level of the NoC needs to be wired back to itself for supporting deflections. Even with Hoplite NoC, or other mesh-based NoCs, we either need to provision the full link bandwidth with sufficient wiring capacity in the channel. Alternatively, the bandwidth may be distributed across multiple links at the expense of an equal number of PE injection slots. In the second case, we can imagine a better alternative for interfacing these system-level sources at the root of the NoC. This has two advantages: (1) the high-bandwidth traffic can be distributed across multiple top-level links, and (2) we do not have to sacrifice any PE bandwidth. This arrangement is only possible with the Local Deflections implementation as the top-level ports are freely available for communication.

VII. Conclusions

Butterfly Fat Trees (BFTs) modified to support deflection routing can outperform state-of-the-art FPGA overlay NoCs such as Hoplite by as much as 2–5× for uniform random traffic when considering sustained throughputs for highly loaded networks at identical PE counts and by as much as 1.5× when considering identical resource costs at large system sizes >16K LUTs. BFTs also deliver superior worst-case latency behavior improving it by ≈5× for 256 PEs with uniform random traffic. We modify the routing function to localize deflections with loopbacks to avoid the long deflection delays to the root of the NoC. This allows us to deliver 20–40% improvements in throughput and 50–60% better worst-case latencies over the root-based deflection designs when routing uniform random traffic above 15% injection rate. BFTs are multi-level networks that offer configurable bandwidth in each level, allowing an FPGA developer to tailor the NoC capability to application requirements and constraints. We show that the best BFT configuration for RANDOM workloads varies from a binary tree (p=0) for systems with <2K LUTs, mesh-equivalent BFTs (p=0.5) for systems <10K LUTs, and crossbars (p=1) if cost is not a constraint. For workloads with locality, the (p=0.67) BFT topology delivers robust performance at different system sizes.

References

[1] B. S. Landman and R. L. Russo, "On a Pin Versus Block Relationship For Partitions of Logic Graphs," *Computers, IEEE Transactions on*, no. 12, pp. 1469–1479, 1971.

[2] M. K. Papamichael and J. C. Hoe, "CONNECT: re-examining conventional wisdom for designing nocs in the context of FPGAs," in *the ACM/SIGDA international symposium*. New York, New York, USA: ACM Press, 2012, p. 37.

[3] Y. Huan and A. DeHon, "FPGA optimized packet-switched NoC using split and merge primitives," in *Field-Programmable Technology (FPT), 2012 International Conference on*, Dec. 2012, pp. 47–52.

[4] 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 *Proc. 14th IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2006, pp. 205–216.

[5] 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.

[6] N. G. Kapre, "Packet-switched on-chip FPGA overlay networks," Master's thesis, California Institute of Technology, 2006.

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

[8] T. Moscibroda, O. Mutlu, T. Moscibroda, and O. Mutlu, *A case for bufferless routing in on-chip networks*. New York, New York, USA: ACM, Jun. 2009, vol. 37.

[9] C. Fallin, G. Nazario, X. Yu, K. Chang, R. Ausavarungnirun, and O. Mutlu, "Minbd: Minimally-buffered deflection routing for energy-efficient interconnect," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, May 2012, pp. 1–10.

[10] M. K. Papamichael and J. C. Hoe, "The CONNECT Network-on-Chip generator," *Computer*, vol. 48, no. 12, pp. 72–79, 2015.