Hoplite: A Deflection-Routed Directional Torus NoC for FPGAs

NACHIKET KAPRE, University of Waterloo
JAN GRAY, Gray Research LLC

We can design an FPGA-optimized lightweight network-on-chip (NoC) router for flit-oriented packet-switched communication that is an order of magnitude smaller (in terms of LUTs and FFs) than state-of-the-art FPGA overlay routers available today. We present Hoplite, an efficient, lightweight, and fast FPGA overlay NoC that is designed to be small and compact by (1) using deflection routing instead of buffered switching to eliminate expensive FIFO buffers and (2) using a torus topology to reduce the cost of switch crossbar. Buffering and crossbar implementation complexities have traditionally limited speeds and imposed heavy resource costs in conventional FPGA overlay NoCs. We take care to exploit the fracturable lookup tables (LUT) organization of the FPGA to further improve the resource efficiency of mapping the expensive crossbar multiplexers. Hoplite can outperform classic, bidirectional, buffered mesh networks for single-flit-oriented FPGA applications by as much as 1.5× (best achievable throughputs for a 10×10 system) or 2.5× (allocating same amount of FPGA resources to both NoCs) for uniform random traffic. When compared to buffered mesh switches, FPGA-based deflection routers are ≈3.5× smaller (HLS-generated switch) and 2.5× faster (clock period) for 32b payloads. In a separate experiment, we hand-crafted an RTL version of our switch with location constraints that requires only 60 LUTs and 100 FFs per router and runs at 2.9ns. We conduct additional layout experiments on modern Xilinx and Altera FPGAs and demonstrate wide-channel chip-spanning layouts that run in excess of 300MHz while consuming 10–15% of overall chip resources. We also demonstrate a clustered RISC-V multiprocessor organization that uses Hoplite to help deliver the high processing throughputs of the FPGA architecture to user applications.

CCS Concepts: • **Networks** → **Network on chip**; • **Hardware** → **Programmable interconnect**; • **Computer systems organization** → *Reconfigurable computing;*

Additional Key Words and Phrases: FPGA overlays, unidirectional torus, deflection routing

ACM Reference Format:
Nachiket Kapre and Jan Gray. 2017. Hoplite: A deflection-routed directional torus NoC for FPGAs. ACM Trans. Reconfigurable Technol. Syst. 10, 2, Article 14 (March 2017), 24 pages.
DOI: http://dx.doi.org/10.1145/3027486

1. INTRODUCTION

It is an important and popular fact that the design and engineering of network,-on-chip (NoCs) in modern system,-on-chip (SoCs) and multiprocessing fabrics is critical for performance and energy efficiency. With the rising demand for accelerator building blocks and the variety of IP cores available, the use of an NoC-based communication fabric for assembling large designs quickly has never been more important.

The programmable FPGA fabric makes it possible to support *overlay* NoCs that are configured on top of configurable lookup tables (LUT) and routing resources. They are adaptable, customizable and tuneable but have generally suffered from high resource

Authors' addresses: N. Kapre, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo N2L 3G1, Canada; email: nachiket@uwaterloo.ca; J. Gray, Gray Research LLC, PO Box 3785, Bellevue, WA 98009-3785, USA; email: jsgray@acm.org, web: http://fpga.org.

DOI: http://dx.doi.org/10.1145/3027486

requirements when mapped to FPGAs. For example, the 32b CMU CONNECT router [Papamichael and Hoe 2012] takes up 1.5K LUTs@9.6ns while the 32b Penn Split-Merge router [Huan and DeHon 2012] occupies 1.7K LUTs@4.5 ns. While system-level interconnection tools like Altera Qsys [Altera 2011] provide a way to integrate IP cores in arbitrary topologies, the switching blocks are still too expensive (1.6K Adaptive Logic Modules (ALMs) per router for a fully connected 16-node design) and unsuitable for massively-parallel composition of computing cores. For context, a lightweight RISC-V processor mapped to an FPGA [Gray 2016] consumes 320 LUTs. This 4–5× imbalance in resource utilization of the processing and communication components has typically limited the performance of FPGA-based parallel overlays. The key culprit in this scenario is the inability to cheaply implement small FPGA onchip buffers and wide multi-bit crossbars while retaining high performance. In the full-custom or ASIC domain, NoC designs do not suffer the same drawbacks as custom SRAM buffers, and crossbar arrays can be provisioned directly on silicon without any intermediate configuration layer. Unlike traditional ASIC NoCs, support for virtual channels (VCs) and other exotic NoC infrastructure is too expensive to overlay on top of the FPGA. Under these circumstances, the performance limits and exorbitant costs of FPGA overlay NoCs can become a stumbling block in the wider adoption and integration of NoCs beyond small system sizes. Some recent academic studies have investigated the potential for hard NoCs [Abdelfattah and Betz 2012] that are essentially ASIC-style NoCs embedded within the FPGA fabric. However, it will take years before we see them in a shipped product if the business case is made in its favor. Furthermore, depending on the wiring budget allocated to the hard NoC, they may still be unable to perfectly satisfy the communication requirements of spatial applications. Given the reality of these constraints, high-performance low-cost NoCs can help fill a crucial gap in the landscape of FPGA-based overlays today by offering a competitive solution for building massively parallel processing fabrics.

In this article, we investigate the latency and throughput characteristics of constructing the Hoplite [Kapre and Gray 2015] overlay NoC using bufferless deflection-routed torus [Moscibroda et al. 2009] on top of a modern FPGA fabric. Deflection routing operates by sending incoming packets at a NoC router to available output ports even when the desired output port is unavailable. No packet buffering is permitted in the router. Thus, if a packet cannot route along its natural (fastest) direction of traversal, it gets deflected to a less desirable output resulting in a longer traversal. In exchange for this performance loss, we can build a simpler, cheaper router without buffers or complex arbitration policies. While the idea of deflection routing is nothing new, we show how to apply this to the FPGA substrate and cleverly exploit the LUT organization and mapping capabilities to best conserve available resources. Bufferless deflection routers such as the CMU BLESS [Cai et al. 2015] design are smaller than the CMU Connect and Penn Split-Merge designs but still occupy 1K LUTs@13.2ns as they were not designed with the FPGA organization in mind. Deflection routing, when implemented specifically to exploit the characteristics of the FPGA substrate, greatly simplifies the engineering of overlay networks. In particular, the use of unidirectional torus instead of a bidirectional network enables simpler switching and control logic in the design. This is a better match to the LUT organization of the FPGA enabling efficient implementation of the NoC multiplexers. The use of deflection routing elegantly handles conflicts within the NoC without resorting to complex handshake-based control and FIFO-based designs that add complexity to the router arbitration logic. This reduction in complexity simplifies the circuit design thereby enabling faster clock frequency of the NoC fabric. Hoplite was first devised to efficiently interconnect hundreds of soft processors in the Phalanx system [Gray 2014] but is broadly applicable to other roles.

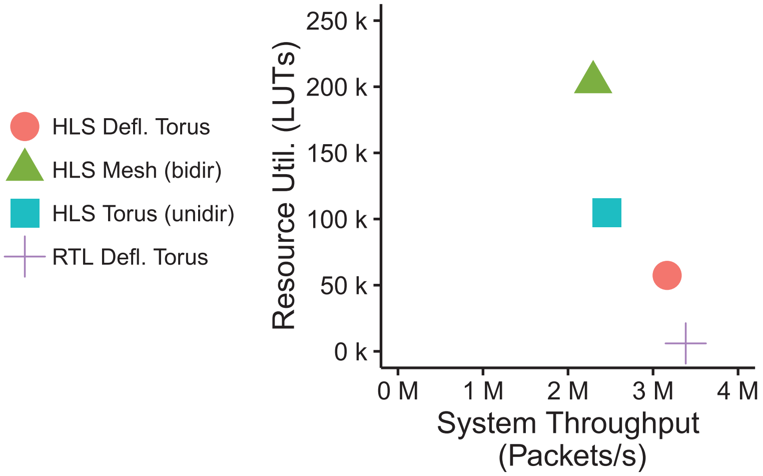


Fig. 1. Area-throughput tradeoffs for various switches in a 10×10 NoC (Virtex-6 LX240T FPGA) under random traffic and offered injection rate of 0.5 packets/cycle/PE. Also showing results for hand-crafted RTL implementation of Hoplite. HLS, high-level synthesis.

In Figure 1, we preview our preliminary results for the various switches we developed for an 10×10 NoC. Typically, we expect the area-time engineering tradeoffs to improve performance at increased cost. However, we observe the opposite behavior in this experiment. Even when evaluating the simple effect of changing topologies from a mesh to a torus (excluding FPGA-optimized Hoplite), we observe throughput improvements with large reductions in LUT costs. For instance, the two-dimensional (2D) bidirectional mesh requires ≈200K LUTs and delivers a throughput of ≈2M packets/s. In contrast, the 2D unidirectional torus requires ≈100K LUTs while slightly improving NoC throughput. Thus, the mere choice of a directional 2D torus alone helps lower FPGA implementation cost by simplifying the switching crossbar. Now, when we further optimize and select deflection routing over buffered switching on the torus, we reduce resource utilization to ≈50K LUTs while boosting throughput to ≈3M packets/s. This second benefit is achieved by eliminating FIFO queuing costs and a faster clock frequency. Finally, when we consider FPGA-optimized RTL mapping, the total cost drops to ≈6K LUTs at slightly improved throughput. This reduction is due to the elimination of overheads of high-level synthesis. All effects considered together, the FPGA-optimized deflection-router based design is able to support 30% higher bandwidth for uniform random traffic while also delivering a 3× reduction in FPGA resource utilization at 100 processing elements ((PE) like a soft processor, or a communicating IP block).

We make the following key contributions in this article:

—FPGA-focused design and characterization of buffered mesh, buffered torus, and bufferless deflection router architectures on modern Xilinx and Altera FPGAs. Initial comparison based on Vivado High-Level Synthesis (for Xilinx FPGAs), to enable fair comparison across various design styles, along with an optimized RTL mapping and floorplanning toolflow for both Xilinx and Altera FPGAs.

—Performance evaluation and quantification of the NoC under various statistical traffic patterns. Analysis of engineering tradeoffs in fairness and latency distribution when using deflection routing for various workloads.

—Demonstration of the efficiency and effectiveness of Hoplite NoCs in constructing GRVI-Phalanx massively parallel processor cluster arrays.

We extend the original Hoplite [Kapre and Gray 2015] work and present new results in the following areas:

—We provide commentary on platform-specific implementation considerations for mapping Hoplite routers to both Xilinx and Altera FPGAs. The clustered LUT architecture of these two FPGA vendors is unique and motivates a specialized implementation for both cases.
—We show how to map a RISC-V multiprocessor accelerator using the Hoplite NoC as an example of a real-world case study.
—We conduct extensive floorplanning experiments for large chip-spanning NoCs of different sizes and different interface widths to push the FPGA fabric to the limit.
—We attempt to understand the worst-case latency trends for the base Hoplite architecture to motivate future designs.

2. BACKGROUND

2.1. Context

Modern computing fabrics including multi-core CPUs, SIMD GPUs, and heterogeneous embedded SoCs have all adopted some form of time-shared networking resource for exchanging data and control. These networks are used to support cache coherency traffic or explicit, user-controlled DMA data transfers between IP blocks. These scenarios cover application communication requirements that are dynamic and unknown until runtime. Unlike these computing systems, FPGAs have long supported *statically configured* routing resources that are programmed and managed offline during compile time. While this structure is ideal for circuit-style dataflow computations, there is still a demand for supporting traffic generated by computing overlays (e.g., Vector, VLIW, Dataflow) or between IP cores using an AXI-compatible bus protocol. In particular, we are interested in supporting massively parallel, customized soft-processor arrays programmed on top of the FPGA fabric. We consider traffic patterns [Abad et al. 2012] where we generate a large number of packets that needs to be independently routed to dynamically determined destination information. To support such dynamic workloads, we need packet-switched overlay designs where each packet (or flit) is routed based on address information that is bundled with the data (payload).

2.2. Related Work

While we still use them today, bus-based shared networks were common in the resource starved silicon-poor era of the late 1990s–2000s. As wiring delays overwhelmed gate delays and the effects of Rent's rule [Landman and Russo 1971] manifested in system-level communication requirements, it was no longer adequate to rely purely on busses alone. Shared, switched networks that route packets instead of wires [Dally and Towles 2001] became increasingly important. ASIC-based NoC designs have enjoyed the ability to introduce new performance-enhancing features such as virtual channels and exotic flow-control strategies, as their silicon implementation costs are fairly modest. Few FPGA-based NoC router designs such as the CMU CONNECT [Papamichael and Hoe 2012] have attempted to replicate this model on top of FPGAs and have reported slow and large designs. However, the cost of supporting virtual channels on FPGAs in the style of CMU CONNECT is high as each VC buffer and associated control adds to circuit complexity. While the CONNECT framework allows construction of virtual-channel free designs, and FPGA-amenable wire-rich higher-radix topologies, they ultimately still need expensive FIFO-based flow control. The Penn Split-Merge [Huan and DeHon 2012; Kapre et al. 2006] architecture throws out the ASIC-inspired design methodology in favor of simpler, VC-free, handshake-based, pipelining-friendly FPGA switches. However, both these designs still spend 30–40% of their resources on crossbar switching

Table I. Comparing FPGA-Based NoC Routers (32b Payloads, Xilinx Virtex-6 LX240T, Otherwise Indicated)

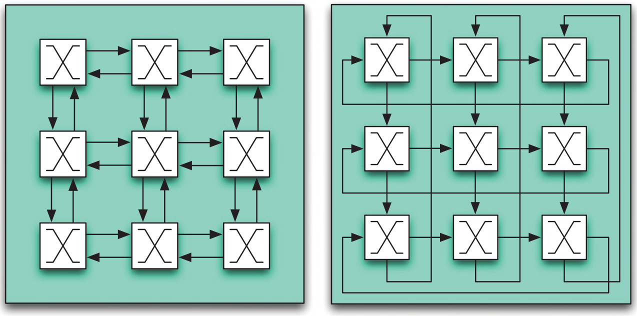
Router	**LUTs**	**FFs**	**Cycle** (ns)
BLESS (without buffers) [Cai et al. 2015][1]	1,090	335	13.2
CONNECT 2VCs [Papamichael and Hoe 2012]	1,562	635	9.6
Split-Merge DOR [Huan and DeHon 2012; Kapre et al. 2006]	1,785	541	4.5
Altera Qsys [Altera 2011][2]	1,673	—	3.1
Hoplite NoC written in Vivado HLS (This paper)			
Mesh	2,035	1,669	6.8
Torus	1,046	949	4.8
Deflection Torus	576	570	3.1
Hoplite RTL (This paper)			
Deflection Torus	60	100	2.9

[1] FPGA used for BLESS router mapping is an older Virtex-2 Pro XC2VP70-FF1704 with 4-LUT architecture.
[2] Derived from Table 1 of Altera [2011] for a fully connected 16-node system. Stratix IV C2 speed grade part. Interface width not indicated.

and 20–40% on buffering requirements while operating between 90 and 200MHz. The Split-Merge router can be run faster up to 310MHz at the cost of additional pipelining per hop, which directly affects worst-case latency. For statically known workloads, we may instead use time-multiplexed NoCs [Kapre et al. 2006] that store the pre-computed routing decisions in lookup tables at each router output port. This has been shown to reduce resource requirements by 2× or more while also operating faster due to simpler, cleaner multiplexing logic. However, this design style does require *a priori* static knowledge of the communication workload and an offline scheduling step that may not always be feasible. Low-cost routers [Kim 2009] and bufferless deflection routers were proposed in Moscibroda et al. [2009] as a way to address the rising buffer costs (area, delay, power) in ASIC-based NoCs for multiprocessing workloads with multi-flit packets. This was refuted in Michelogiannakis et al. [2010], where energy benefits were found to be minimal and latency and bandwidth benefits of the buffered designs were superior. In Cai et al. [2015], an FPGA implementation of the BLESS bufferless deflection router is presented that is cheaper than the CMU Connect and Penn Split-Merge designs. BLESS uses a 2D bidirectional mesh topology that requires wider crossbar input sizes than a directional torus. This poorly matched the FPGA LUT organization, resulting in an implementation that still requires 1K LUTs running at 13.2ns clock period. In this article, we investigate the potential for implementing deflection routers organized in a directional torus for single-flit workloads. We summarize key resource utilization results for our proposed design in Table I and contrast it against existing FPGA overlay NoC routers.

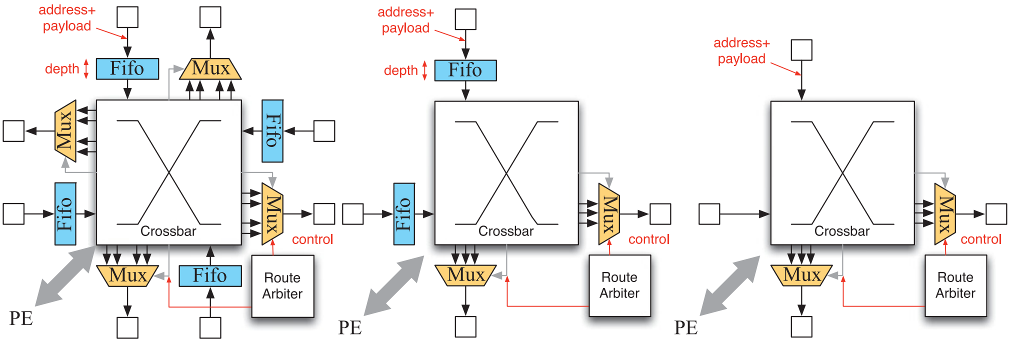
2.3. Network-on-Chip Design

While we can organize the switches in various topologies with CMU Connect and Altera Qsys tools, we only consider designing regular 2D layouts for this work. In particular, we choose to evaluate overlay NoCs connected in a 2D layout using mesh and torus topologies as shown in Figure 2. Data are routed over the network in a packet-switched style. While generally, each packet can be composed of multiple flits, we consider finer-grained load-store-style traffic generated in a multiprocessing fabric. In this scenario, each packet is a single flit and carries two fields: (1) address of destination and (2) data payload. To avoid deadlock during routing, we can implement various routing algorithms in the switch such as Dimension-Ordered Routing (DOR) and West-Side First. For simplicity, we support DOR routing, as it is cheap and straightforward



(a) 2D Bidirectional Mesh (b) 2D Directional Torus

Fig. 2. Topology of NoCs considered in this study: A bidirectional mesh and a directional torus.



(a) Bidirectional Mesh (b) Directional Torus (c) Deflection Torus

Fig. 3. Switch organizations evaluated in this study. Bidirectional buffered mesh and directional buffered torus needs FIFO buffers, and wires in each dimension, while directional torus eliminates buffers as well.

to implement on the FPGA logic similar to the Penn and CMU routers. Furthermore, we do not address livelock scenarios that are possible in deflection routing.

3. DEFLECTION ROUTED TORUS NOC

In this article, we explore the architecture tradeoffs in the use and optimization of FPGA NoCs with network traffic generated from statistical sources. Specifically, we investigate the impact of bufferless routing on FPGA NoC implementation cost and on performance in terms of latency and bandwidth.

In Figure 3, we show the high-level internal organization of three styles of switches that could be used to build an NoC for a 2D mesh of PEs. A classic buffered 2D Mesh switch supports DOR routing and packet traversals in all four directions of the mesh (and the PE). The mesh switch is organized into three principal building blocks that contribute to cost of the switch: (1) crossbar, (2) route arbitration, and (3) buffering and multiplexing of IOs. The implementation complexity of the crossbar is $O(N^2)$, where N is the number of IOs. For the 2D bidirectional mesh, $N = 5$ (five inputs and five outputs, loopback self-edge not supported thereby depopulating the crossbar by 1) and is typically implemented using one Virtex-6 6:1 LUTs (MUXes) per bit per output port. We also need to consider any associated pipelining registers and FIFOs for performance. The route arbiter implements DOR routing algorithm and handles conflicts emerging from the packets being routed. The rest of the resource requirements are due to the input FIFOs and bypass MUXing and registers along with output registers. These FIFOs

Table II. Relative Resource Utilization (LUTs and FFs) of Various Switch Blocks in Vivado HLS

	Crossbar	(%)	Arbiter	(%)	Total	(%)
Mesh (LUTs)	860	42	280	14	2035	100
Torus (LUTs)	286	27	64	6	1046	100
Defl. Torus (LUTs)	215	37	130	22	576	100
Mesh (FFs)	389	23	97	6	1669	100
Torus (FFs)	223	23	31	3	949	100
Defl. Torus (FFs)	205	35	79	13	579	100

are implemented using SRL blocks and registers with bypass paths to reduce queueing delay within the switch. This alone is sufficiently complex to implement cheaply on the FPGA, and we eschew the idea of Virtual Channels and complex credit-based flow control to keep hardware simple. We tabulate these requirements for a 32b payload, 16b address switch in the Table II. We obtain these number from compiling synthesizable C++ descriptions of the different NoC routers in Vivado high-level synthesis (HLS). These offer a substantial speedup over Verilog-based designs for functional verification on various workloads and enable a fair comparison across the different design styles.

The use of a directional network such as a torus is one way to lower implementation cost specifically in the crossbar block. A unidirectional buffered torus switch only accepts packets from two dimensions (and the PE) thereby reducing the crossbar complexity as we now have $N = 3$ (one connection for the PE and two connections for vertical and horizontal lanes).

Buffering cost can be eliminated by adopting a deflection routing technique [Moscibroda et al. 2009] that misroutes packets in the presence of a conflict. In buffered switches, FIFOs are used to hold incoming packets when the desired outgoing port is not available, that is, another packet is using the port in that cycle. In contrast, in deflection-routed networks, we intentionally misroute packets (knowing that deadlock is not possible in bufferless networks) along available ports if the desired port is not available to simplify circuit implementation cost. As we have identical numbers of incoming and outgoing links in the switch, we can always guarantee an outgoing slot for incoming packets. For the outgoing PE port, we misroute the packet back into the PE if the switch is occupied, thereby creating implicit backpressure. We can observe the reductions in resource costs in Table II. The 2D buffered mesh requires roughly 3.5× more LUTs and 3× more FFs than the deflection routed torus. The buffered torus switch is only marginally larger than the deflection routed torus requiring 1.8× more LUTs and 1.6× more FFs. When considering switch throughputs, the deflection routed torus runs roughly 2.1× faster than the buffered mesh switch and 1.5× faster than the buffered torus. Thus, the simpler and leaner switch design of the deflection routed torus requires fewer resources and runs faster than the alternative buffered switches. The route arbitration for the deflection torus is indeed marginally slower (by 0.1–0.2 ns) due to the more complex routing logic for handling all deflection cases, but overall clock frequency is still faster.

When comparing the efficiency of a buffered network to that of a bufferless network, we must consider the relative impact of *congestion*. In a buffered network, this manifests as *waiting* time in the buffers, while, in a deflection torus, congested packets are *misrouted*. While implementation of the misrouting policy is substantially simpler on the FPGA (see Table II), packets may now take multiple trips in the NoC due to misrouting at conflicted switches before reaching their destination. While this may seem unfair, as packets may re-traverse the NoC again covering longer distances, the penalty of waiting time as well as leaner, faster deflection-routed FPGA switches will make this an interesting architecture comparison. Thus, in the final evaluation, we will observe a composite effect of waiting time in buffered networks with the slower NoC design competing with the faster, simpler deflection router with its associated misrouting overheads.

4. FPGA-AWARE HOPLITE IMPLEMENTATION

With the high-level context of understanding deflection torus NoCs against other topologies in Section 3, we now show how to map Hoplite to modern FPGA fabrics. For this, we directly use a Verilog RTL implementation instead of the Vivado HLS switch shown earlier. Since the switching crossbar occupies a large fraction of total switch area, it is important that we first understand how this structure maps to the FPGA LUT organization.

4.1. Switch Datapath Optimization

The optimized switch datapath is the culmination of a series of architecture simplifications and FPGA-specific technology mapping optimizations (Figure 4) that reduce the switch area of a router with w-bit links from $10w$ 6-LUTs or more to just w 6-LUTs that is, one LUT per router per bit of link width. Each one-bit slice of a Hoplite router's 3×2 switch and its registers is technology mapped into a single Xilinx 6-LUT or Altera ALM, with an FF→wire→LUT→FF critical path.

—**Use a directional torus.** The Hoplite switch datapath is partially inspired by the dimension sliced router [Kim 2009] but is more austere. The unidirectional torus reduces the switch crossbar from 5×5 to 3×3 (Figure 4(a) and Figure 4(b)). Here, we delete NO and EO (north out and east out) output links. The remaining outputs SO, EO, O (relabeled Y, X, O) use simpler 3:1 MUXes that are a better match for 6-LUT FPGAs. This saves at least $2w$ LUTs.

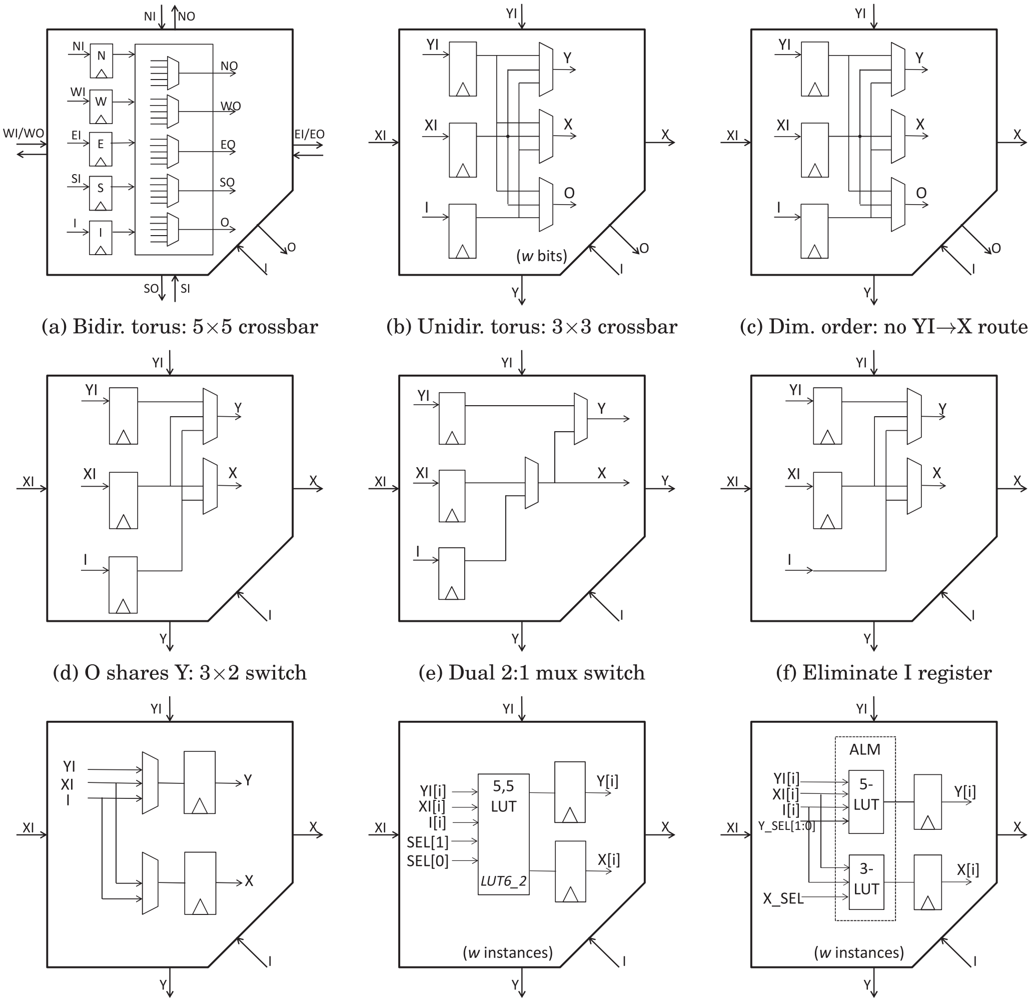
—**Use X→Y dimension-order routing.** We can further save resources by restricting the possible turns in the switch to DOR. Thus, at router (x1,y0), any YI input message has destination (x1,y1) with matching x1, and therefore never routes to the X output. We can elide the YI input to the X output MUX (Figure 4(c)) and simplify the X MUX to a 2:1 w-bit MUX.

—**Share output links.** In the lifecycle of a message, it enters the NoC at some router (x0,y0). It traverses some routers and X links to (x1,y0). Perhaps it deflects and traverses more X links, back around to (x1,y0). Eventually it routes on Y to (x1,y1) and exits the NoC. Along that path, it uses an X or Y output every cycle but only uses an O PE output once. Since the PE output port is infrequently used, that is, inessential, it may be elided by occasionally borrowing another output link, further simplifying the switch to 3-inputs×2-outputs. Therefore, we can delete the O PE output MUX and use (share) the Y output instead. (A separate output O_v signals that the Y output is a valid output message to the PE.) This saves about w LUTs (Figure 4(d)).

—**Factor MUXes.** It is possible to factor the 3:1 and 2:1 MUXes into two w-bit 2:1 MUXes (Figure 4(e)). Both Altera and Xilinx fracturable 6-LUT FPGAs can implement two 2:1 MUXes per LUT. This switch requires just $2\ w/2 = w$ LUTs but incurs two LUT delays and does not allow some routes (e.g., I→Y while XI→X). Better tech mappings follow.

—**Eliminate I register.** The switch's (Figure 4(d)) PE input I register is inessential and can be elided—a PE can include one of its own if necessary (Figure 4(f)).

At this point, the switch uses two w-bit registers and two w-bit MUXes. But if this design is presented to FPGA implementation tools, it may still use up to twice as many resources as necessary. To understand why requires a brief review of Xilinx [Xilinx Inc.



(a) Bidir. torus: 5×5 crossbar (b) Unidir. torus: 3×3 crossbar (c) Dim. order: no YI→X route

(d) O shares Y: 3×2 switch (e) Dual 2:1 mux switch (f) Eliminate I register

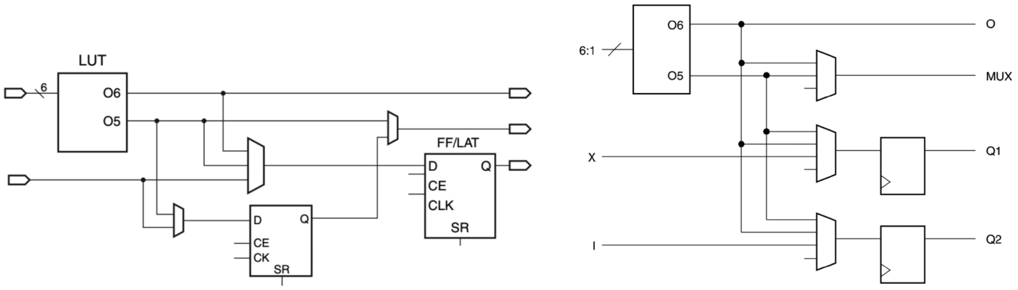
(g) Switch precedes registers (h) Xilinx 5,5-LUT tech mapping (i) Altera ALM tech mapping

Fig. 4. Hoplite switch datapath simplifications and FPGA technology mappings.

2015] and Altera [Altera Corp. 2015] logic cell architecture. Modern FPGAs provide programmable logic cells with 6-input lookup tables (6-LUTs) followed by two to four D flip-flops. Different FPGA families have different logic cell interconnect constraints.

4.2. Xilinx-specific Optimizations

Xilinx 6-LUTs can be subdivided into two 5-LUTs that compute two functions of the *same* five inputs. In the Virtex-6/7, these outputs O6 and O5 may be optionally registered in two FFs (Figure 5(a)). The only way to pack two 5-LUTs *and* two D-FFs per logic cell is to register the LUT outputs of the same logic cell. In contrast, in the UltraScale family, the two flip-flops may be used independently of the 5,5-LUT (Figure 5(b)). It is still best to route the FF inputs from the adjacent 5,5-LUT using dedicated intra-logic cell routing as it is faster, conserves inter-logic cell and inter-logic-cell-cluster interconnect, and (presumably) saves energy. Therefore, continuing with the switch elaboration:



(a) 7-series 6-LUT with *dependent* D-FFs

(b) UltraScale LUT, two D-FFs

Fig. 5. Xilinx LUT/FF technology mapping considerations. Schematics derived from Xilinx Inc. [2016a, 2016b].

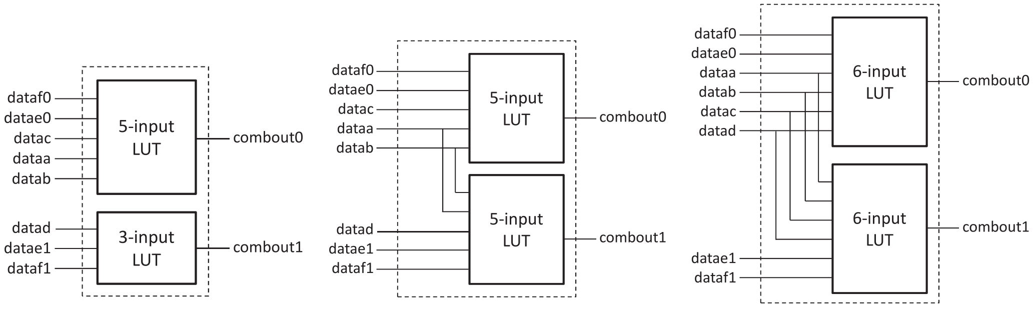
Table III. Xilinx 3-input, 2-output Switch: $SEL_{1:0}$ Selects One of Four *Implemented* Transfer Functions. The Five Transfer Functions Marked – Are Not Implemented

	$SEL_{1:0}$	Y	X	Purpose
1	00	I→Y	I→X	Y ring ingress, simpler SEL
2	–	I→Y	XI→X	(XI→X plus Y ring ingress)
3	–	I→Y	YI→X	(Not dimension-order routing)
4	–	XI→Y	I→X	(Ingress *and* egress)
5	01	XI→Y	XI→X	Fanout for multicast
6	–	XI→Y	YI→X	(Not dimension-order routing)
7	10	YI→Y	I→X	YI→Y plus X ring ingress
8	11	YI→Y	XI→X	Route X and Y ring messages past each other
9	–	YI→Y	YI→X	(Not dimension-order routing)

—**Register the LUT outputs.** Replace input registers XI, YI with output registers X, Y. These D-FFs map into the same logic cells as their input MUX LUTs (Figure 4(g)).

—**Xilinx 5,5-LUT technology mapping.** Each Xilinx 6-LUT in 5,5-LUT mode can compute a pair of output functions Y_next_i and X_next_i, both a function of five common inputs YI_i, XI_i, I_i, and $SEL_{1:0}$ (Figure 4(h)). Table III details nine possible transfer functions; each cycle $SEL_{1:0}$ can select one of a fixed subset of four, but which four of nine? Three functions with YI→X (3, 6, and 9) are unnecessary under dimension-order routing. Of the remaining six, the base (default) Hoplite router switch for Xilinx implements 1, 5, 7, and 8. The rationale is as follows: transfer functions 7 and 8 are used most often, routing X and Y ring traffic past each other and for X ring message ingress. Function 5 enables XI→Y routing as well as message fanout for multicast delivery. Function 1 provides I→Y routing and enables a remarkable simplification of the router switch arbiter, described next. (Functions 2 and 4 are beneficial in some scenarios, but we can only pick four.) Overall, this technology mapping achieves an area of one Xilinx 6-LUT per router per bit of link width.

—**Use an empty (zero LUTs) router switch arbiter.** The default ("Base") Hoplite router *for Xilinx* implements basic dimension order routing, routing on X before Y, and with a static input prioritization, enabling simple switch arbitration logic:
—X_next ← XI if XI valid, else I.
—Y_next ← YI if YI valid, else X_next.
Given the transfer functions and SEL encodings in Table III, SEL is just the concatenation of {YI.v, XI.v}, that is, Y.v of the prior router on the Y ring and X.v of the prior router on the X ring. In other words, zero LUTs—and zero LUT delays—are required to determine the switch's output multiplexer selects from its input message links. It should be noted, however, that we still need some decoding logic to generate



(a) ALM as 5-LUT, 3-LUT (b) ALM as dual 3:1 mux (c) ALM as 4×2 crossbar

Fig. 6. Altera LUT/FF technology mapping considerations.

valid signals for each outgoing port, *i.e.* X.v, Y.v and O_v. But it is still valid to claim that the arbitration itself consumes zero LUTs.

An example illustrates router switch and NoC operation. Assume a 3×3 Hoplite NoC that is idle (empty). A message is sent from PE at (0,0) to the adjacent PE at (1,0). At time 0, the NoC is idle. Each of the nine routers' switches have negated valid outputs X.v and Y.v. The PE at (0,0) asserts its message on I and asserts I.v (message valid). At router (0,0), the $\mathrm{SEL}_{1:0}$ input, that is, router (0,2)'s Y.v, concatenated with router (2,0)'s X.v, is 00. SEL = 00 selects transfer function 1, so the message on router (0,0)'s PE input I transfers to its X and Y output registers. DOR logic compares the router's coordinates (0,0) with the message destination (1,0) and asserts X.v and negates Y.v, propagating the valid message to router (1,0). At time 1, at router (1,0), the message is received on XI, with XI.v asserted, and YI.v (i.e., router (1,2) Y.v) negated. SEL = 01 selects transfer function 5, transferring XI to its output registers X and Y. Router (1,0)'s coordinates match the message destination, so DOR logic negates X.v *and* Y.v, and asserts O_v, so the PE at (1,0) receives the message on its router's Y output.

4.3. Altera-specific Optimizations

The Altera ALM provides input inputs into its 6-LUT, which can also implement a 5-LUT plus 3-LUT (Figure 6(a)), two 5-LUTs that share two inputs (a dual 3:1 MUX) (Figure 6(b)), or two 6-LUT functions that share four inputs (a 4×2 partial crossbar) (Figure 6(c)). This fracturing differs from the Xilinx LUT organization and hence merits a new strategy for embedding the Hoplite switch crossbar.

—**Altera fracturable ALM technology mapping.** The 3:1 Y MUX requires a 5-LUT. The 2:1 X MUX requires a 3-LUT. Both will pack into a single Altera ALM (Figure 6(a)). The two LUT outputs can be routed to their D-FFs using intra-ALM interconnect (Figure 4(i)). This achieves an area of one ALM per router per bit of link width. And since an ALM can implement dual 3:1 MUXes (Figure 6(b)), Altera implementations can also efficiently implement routing algorithms beyond dimension order routing, that is, with both XI→Y and YI→X turns.

—**Use Altera DFFEAS D-FF load MUXes(?)** In principle, it is possible to implement a remarkable *two* bits of switch (4 outputs) per ALM, using DFFEAS mode flip-flops. Each of an ALM's four D-FFs has a free load MUX: "d = sload ? sdata : datain;." It is possible to map two bits of "X_next = X_SEL ? XI : I;" to an ALM's ALUTs and two bits of "Y_next = Y_SEL ? YI : X_next;" to its D-FF load MUXes. However, in practice, attempts to scale out this mapping trick to a large switch fail. When using the Quartus Prime Standard 2015.1 Fitter, targeting Stratix V and Arria 10, our mapping experiments only manage to pack 12–13bits (24–26 outputs) of switch per 10-ALM LAB instead of the desired 20bits/LAB. This may be due to limited/sparse interconnect resources entering the LAB or Fitter limitations.

4.4. Switch Optimality

The lower bound on area of a *non-time-multiplexed* 2D torus router datapath, with unidirectional w-bit links, is the minimum logic necessary to drive the $2w$ nets of the w-bit X- and Y-dimension output links. Both X- and Y-dimension outputs require some combinational function of input nets. A modern 6-LUT/ALM FPGA can implement up to two logic function outputs per LUT (Altera DFFEAS mode notwithstanding). Therefore the lower bound on 6-LUTs per w-bit router switch area is $2w/2 = w$ LUTs. Similarly, the lower bound on LUT delays through a 2D router switch that selects message outputs from message inputs is one LUT delay. The present Hoplite router switch design achieves these lower bounds.

4.5. Floorplanning

Floorplanning is a geometry problem that requires assigning different rectangular portions of the physical chip to different logical portions of the design. A NoC is used to interconnect many PEs situated across the die of an SoC. The SoC floorplan may be determined automatically by the FPGA implementation tools, or it may be induced by the explicit placement of the PEs and (or) NoC routers by means of a constraint file generated by a floorplanning constraints script. The script parameters include target FPGA tool, die region, link width, torus dimensions, whether to fold (interleave) the physical arrangement of routers to bound worst-case physical link lengths, the numbers of link pipeline registers to insert to mitigate long wire delays, and whether to pack router switch LUTs in a tight block. Depending on the implementation tool, the script generates UCF (user constraints file), XDC (Xilinx design constraints), or QSF (Quartus settings file) constraints.

RLOCs (Relative Location): Using Xilinx ISE (e.g., 6 and 7 Series), the router RTL may be configured to generate a relationally placed macro (RPM). An RPM is a translation-invariant layout that precisely specifies the placement of logic components within a block. These constraints can be relocated to any portion of the FPGA chip by simply specifying an offset thereby yielding translation invariance. Routers use LUT6_2 (6-input LUT) and FDCE (Register) instances placed by RLOC and RLOC_ORIGIN constraints (relative locations) into tight rectangular regions.

PBLOCKs (Placement Block): Translation invariance is increasingly harder to guarantee on modern FPGAs due to the irregular nature of embedded hard blocks in FPGA silicon. Thus, using newer Xilinx ISE or Vivado tools, router LUTs may also be constrained to a rectangular region called a PBLOCK. A PBLOCK constraint is not translation invariant and is locked to a specific position on the chip. For our router, we constrain the logic to such a rectangular region and let the CAD tools manage low-level placement decisions within the region.

LogicLock: Similarly, using Altera Quartus Prime, router ALMs may be constrained to rectangular LogicLock regions, one per router, as tight rectangles of LABs.

4.6. Example of NoC Resource Usage and Performance

Given the shallow (1 LUT) combinational logic tree per router stage, in large systems the dominant (>80%) contributor to the NoC clock period critical path is the long wire delay from one router site to another. Accordingly, the floorplanning tool is also needed to evaluate representative FPGA resource use and best case clock periods for *die-spanning* NoCs. A test SoC design is configured with a Hoplite NoC of various dimensions and link widths and floorplan layouts. Each Hoplite router *PE* test core is

Table IV. Example Area and Performance of Various Widths and Layouts of 6×4 Hoplite NoCs in a Xilinx KU040-2. Bold Entries Are Shown in Figure 7. Latency is Average No-Load Latency from Source to Destination Routers (All-Pairs)

Width (bits)	Flat/ Fold	Period (ns)	LUTs (K)	FFs (K)	%LUTs	Link BW (Gbps)	Bis.BW (Gbps)	Latency (ns)
64	fold	3.0	1.9	3.5	0.8	21	171	17
64	xyy	2.3	1.9	4.4	0.8	28	223	16
128	fold	3.0	3.4	6.5	1.4	43	341	17
128	xyy	2.4	3.4	8.4	1.4	53	427	17
256	fold	3.1	6.5	12.7	2.7	83	661	17
256	xyy	2.5	6.5	16.3	2.7	102	819	18
512	fold	3.1	12.6	25.0	5.3	165	1321	17
512	xyy	2.5	12.6	32.2	5.3	205	1638	18
1024	fold	3.3	24.9	49.5	10.4	310	2482	18
1024	xyy	2.6	24.9	64.0	10.4	394	3151	18

just a minimal area circuit that swizzles the bits of an output message and offers it back as the next input message to its router.

Table IV is a parameter sweep of this test SoC, implemented in a Xilinx Kintex UltraScale KU040-2 FPGA by Vivado 2015.4. In each case, an $NX = 4$ column × $NY = 6$ row Hoplite NoC is implemented with varying data payload link widths (64b to 1024b), flat or folded torus topologies, and with/without pipelining of long wires. For example, a folded 6×4 64b-wide NoC has a clock period of 3ns, uses <2000 LUTs, that is, <1% of the device LUTs, and achieves a link bandwidth of 21Gbps, a NoC bisection bandwidth of 171 Gbps, and an average no-load latency from the source router to the destination router of 17ns. At the other extreme, a 1024-bit-wide non-folded torus NoC runs at 2.6ns (385MHz), has a link bandwidth of almost 400Gbps, 3Tbps of bisection bandwidth, using about 10% of the LUTs of this mid-sized FPGA.

The “fold” entries in Table IV physically interleave routers so the length of any inter-router link is at most twice the inter-router distance. The “‘xyy” entries are flat tori, not folded. These routers are laid out in increasing coordinate order. All links from (x,y) to (x+1,y) or (x,y+1) are relatively short; however, links from (NX-1,y) to (0,y) or from (x,NY-1) to (x,0) cross the width or height of the die. Since these long nets have wire delays of 5ns or more, the NoC and the floorplan script are configured with pipeline registers. The notation “xyy” indicates there is one pipeline register in the X ring links from (NX-1,y) to (0,y) and two pipeline registers in the Y ring links from (x,NY-1) to (x,0). Therefore, the diameter of such “xyy” NoC’s X rings is $NX + 1 = 5$ and that of Y rings is $NY+2 = 8$. The average no-load latency column data incorporate the additional cycles of latency introduced by these registers.

Figure 7 shows die plots for 3 of the 10 test NoCs, with 64b, 256b, and 1024b link widths. In particular, in the 1024b-wide router designs, the additional x and y,y link pipeline register flip-flops are apparent. In such ultra-wide NoC links, there is routing congestion in the routers and in the rows and columns of logic they span. In practice, such routers are probably best floorplanned using sparse “router locale” regions.

The Hoplite NoC architecture seems well matched to FPGAs with interconnect fabric flip-flops such as the forthcoming Altera Stratix 10’s HyperFlex registers [Hutton 2015]. The NoC’s bufferless feed-forward design, simple flow control, and absence of resets and clock enables on link output and pipeline registers enable a straightforward technology mapping to interconnect flip-flops. By introducing one or two pipeline registers between each router, a Hoplite NoC might run at full device f_{MAX} of 1GHz or more. This should double link bandwidth and bisection bandwidth per ALM, with little impact (for better or worse) on average message delivery latency.

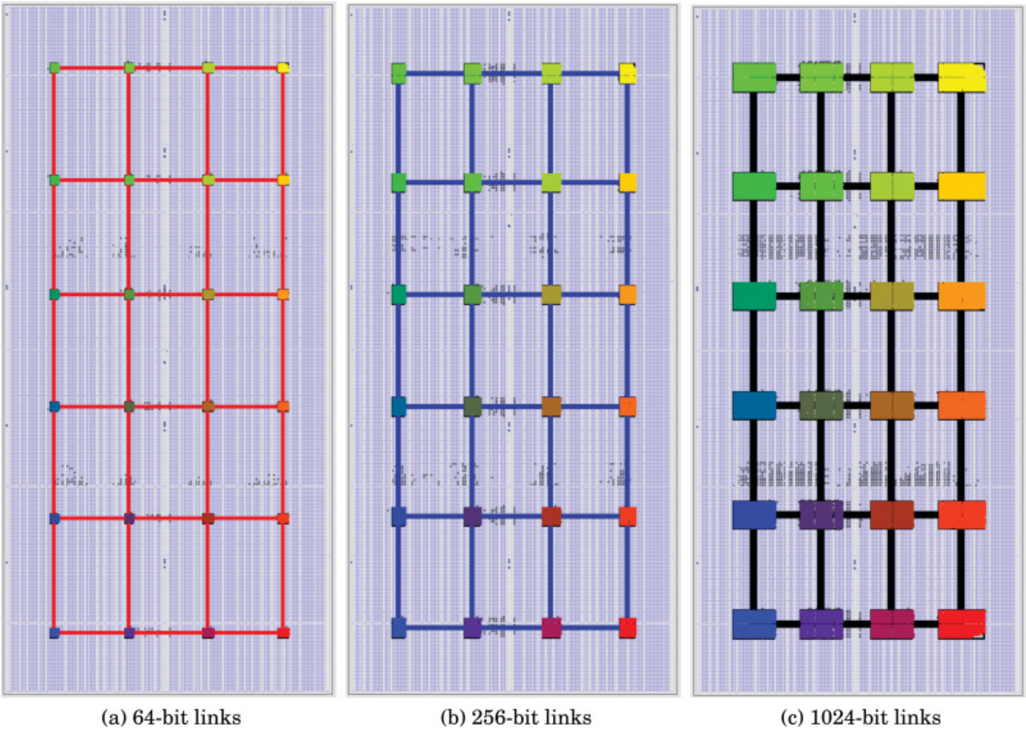

(a) 64-bit links (b) 256-bit links (c) 1024-bit links

Fig. 7. Example 6×4 Hoplite NoCs with 64b, 256b, 1024b links, not folded, with one X and two Y sets of wraparound link pipeline registers, Xilinx KU040-2 target.

4.7. A Hoplite NoC in Practice: GRVI Phalanx

A wide Hoplite NoC lies at the heart of the *work-in-progress* GRVI Phalanx FPGA accelerator framework. GRVI [Gray 2016] (Gray Research RISC-V RV32I) is a RISC-V [Asanović and Patterson 2014] soft processor. Phalanx is a parallel processor and accelerator array framework. In a Phalanx design, groups of processors and accelerators form shared memory clusters, and clusters are connected with each other and with I/O and memory devices by a Hoplite NoC with 300-bit links.

GRVI is an FPGA-efficient implementation of the RISC-V RV32I instruction set. It has an austere two- or three-stage pipelined 32-bit RISC microarchitecture, with register file, operand MUXes, ALU, result MUX, load/store, and program counter logic. It uses about 320 6-LUTs and runs at 300–375MHz in Kintex UltraScale -2 FPGAs.

A typical GRVI cluster (Figure 8) has eight GRVI PEs that share 12 BRAMs. Four BRAMs are used as small 4KB kernel program instruction memories (IRAMs). Each pair of processors share one IRAM. Eight BRAMs form a 32KB cluster shared memory (CRAM) with twelve 32-bit wide ports. Four ports provide a four-way banked address-interleaved memory for PEs. Each cycle, up to four accesses may be made on the four ports via an 8↔4 memory port interconnect. The remaining eight CRAM ports provide an 8-way banked interleaved memory for accelerator(s) and also form a single 256-bit wide port to send or receive 32byte messages, per cycle, to any NoC destination, via the cluster's Hoplite router.

To send a message, one or more PEs prepare a 32byte message in CRAM, and then one PE stores the global address of the message destination to the memory-mapped NoC interface "NOC ITF." This unit loads, and sends a 32byte message to the specified NoC

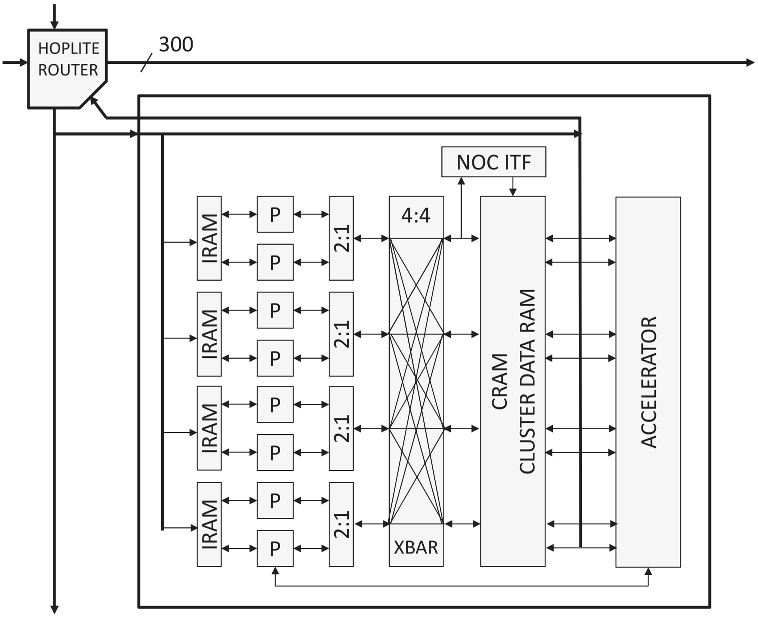


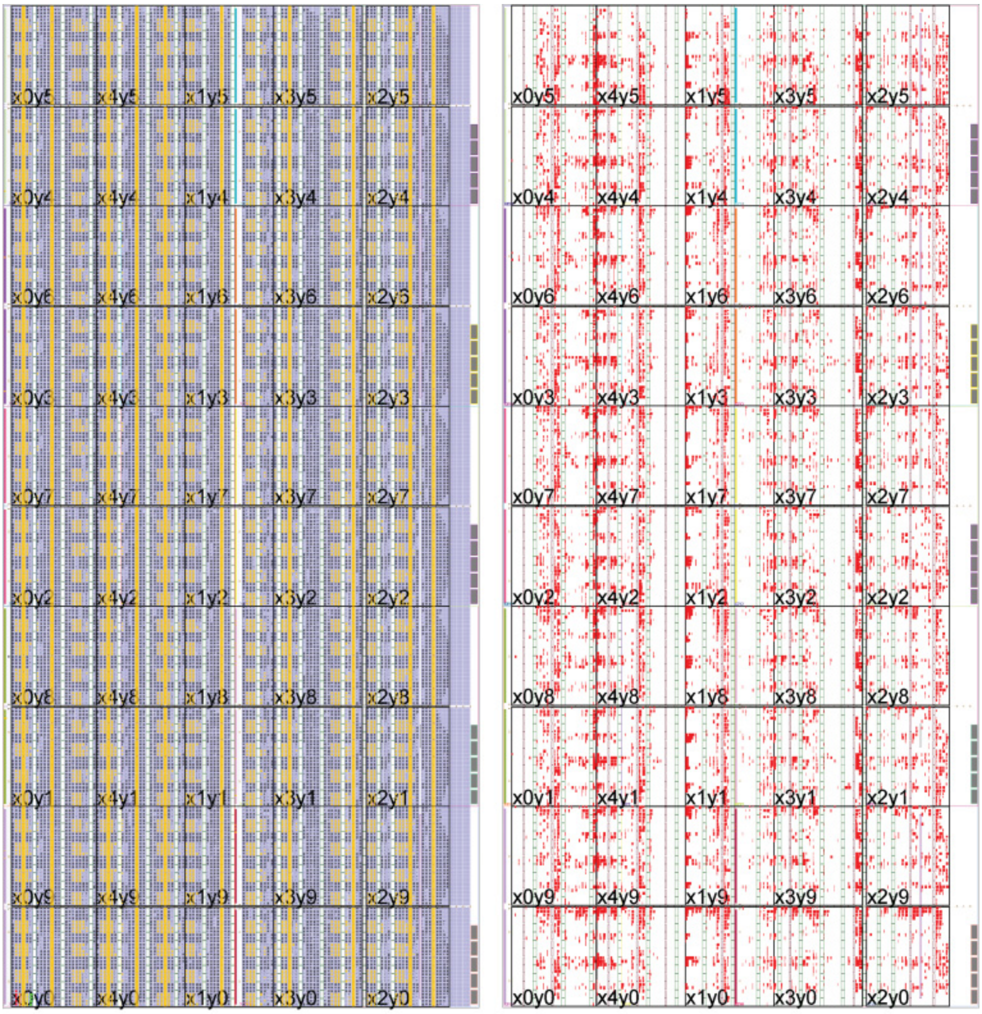
Fig. 8. GRVI cluster tile: one 300b Hoplite router, eight GRVI PEs, RAMs, accelerator(s).

PE via its Hoplite router. If the destination is another GRVI cluster, then the arriving message is immediately written into that cluster's CRAM and/or its accelerator(s). (Accelerators may also use the router to send/receive messages.) NoC message send also enables fast local memcpy and memset. Aligned data may be copied at 32bytes per two cycles by sending a series of 32byte messages from a cluster, via its router, to itself. Eventually, NoC messaging will be used to store/load a 32byte line to DRAM, to send/receive an Ethernet packet (as a series of messages) to/from an Ethernet NIC, and to send/receive data to/from AXI4 (Advanced Extensible Interface) endpoints.

Figure 9(a) is a floorplanned 400 GRVI Phalanx implemented in a Kintex UltraScale KU040. It has 10 rows by 5 columns of clusters (i.e., on a 10×5 Hoplite NoC); each cluster with eight PEs sharing 32KB of CRAM. It uses 74% of the device's LUTs (about 178,000 LUTs) and 100% of its BRAMs (600 BRAMs). The folded sparse 300-bit-wide Hoplite NoC (Figure 9(b)) uses 6% of the device's LUTs (just 40 LUTs per PE). In aggregate, the 400 PEs have a peak throughput of about 100,000 MIPS. Total bandwidth into the CRAMs is 600GB/s.

This design runs at 250MHz. To minimize resource use, there are no elastic buffers or FIFOs in the design *or in its NoC interfaces*. Instead, NoC ingress flow control of message sends is manifest as wait states (pipeline holds) in the PE(s) attempting to send messages. Back pressure from a router, through its local arbitration network, to each PE's pipeline clock enable, is a critical path in the design, and *currently* it caps the clock period at about 300MHz (smaller less congested designs) and 250MHz (die spanning SOCs). At 250MHz, the NoC has a bisection bandwidth of 700Gb/s. Depending on workload, power is 12–17W (30–43mW/processor), measured with SYSMON (Xilinx System Monitor core).

Listing 1 is a Verilog RTL snippet to build a GRVI Phalanx, that is, to instantiate the NoC and NX×NY array of clusters and interconnect clusters to their router ports. (It employs 'XY, and so on, macros to mitigate Verilog's lack of 2D array ports.) A suitable floorplan generator (not shown) produces a constraints file that floorplans the clusters and NoC routers into a die-spanning 10×5 array of tiles.



(a) 400 Processor GRVI Phalanx (b) LUTs used by the folded torus NoC

Fig. 9. GRVI Phalanx: a 10×5 array of 300b-wide Hoplite routers and 8 PE clusters.

5. RESULTS

We stress-test our NoC topologies and switches under various statistical traffic patterns and evaluate throughput, average/worst-case latency scenarios, and the impact of FPGA implementation area on performance. Statistically generated workloads are commonly used within the NoC community [Abad et al. 2012]. We develop processors that generate traffic under the following models: (1) uniform random, (2) uniform random but locality-aware traffic (within an *rlimit*), (3) bit-reverse, (4) tornado, and (5) transpose. These workloads stress the network under both bandwidth and worst/average latency scenarios. We model injection rate as the rate at which packets are available to enter the network (measured as packets per cycle per PE). We generate NoC performance statistics by running our statistical workloads for 32K cycles and consider *offered rates* (or offered throughput) between 0.025 (2.5% of time spent attempting sending packets) and 1.0 (100%). We measure *sustained rates* (or sustained

```
wire`XY    i_rdy; // client input accepted this cycle
wire`XY    o_v;   // client output valid this cycle
wire`MsgXY i;     // client input channels
wire`MsgXY o;     // client output channels

NOC #(.MCAST(MCAST), .NX(NX), .NY(NY), .D_W(D_W), .X_W(X_W), .Y_W(Y_W), ...)
  noc(.clk, .rst_in, .ce, .i_rdy, .i, .o_v, .o);

genvar x, y;
generate
  for (y = 0; y < NY; y = y + 1) begin : ys
    for (x = 0; x < NX; x = x + 1) begin : xs
      Clu #(.MCAST(MCAST), .NX(NX), .NY(NY), .X(x), .Y(y),
            .D_W(D_W), .X_W(X_W), .Y_W(Y_W), ...)
        c(.clk, ...,
          .no_v(o_v[`xy(x,y)]), .no(o[`mxy(x,y)]),
          .ni_rdy(i_rdy[`xy(x,y)]), .ni(i[`mxy(x,y)]));
    end
  end
endgenerate
```

Listing 1. Verilog RTL to generate a GRVI Phalanx and its NoC

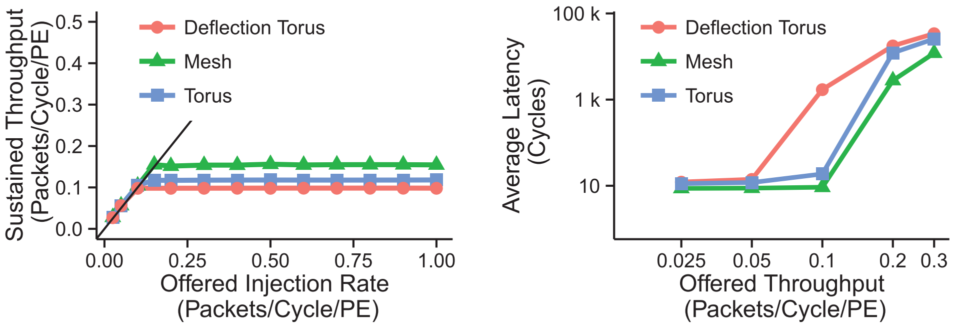
throughput), which are the rates actually possible due to blocking and congestion within the NoC. Consequently, these will be less than the offered rates. We also measure *average latency* along with *worst-case latency* of the workloads by tracking the latency of each packet while including source queueing delay in our measurement (i.e., time spent at the source PE between attempts to enter NoC and actual entry).

We do not have optimized RTL implementations of the buffered NoCs. Hence, we use the placement-and-routed FPGA resource results from our HLS-based switches for all NoCs to enable a fair *apples-to-apples* comparison. We generate RTL for our various switch configurations using synthesizable C++ descriptions of the switch with Vivado High-Level Synthesis. For Hoplite, we use a C++ description of Hoplite shown in Figure 4(c). We also evaluated other configurations from Figure 4 and observed only 1–2% loss in performance, so the performance results presented here are representative (only implementation costs will vary across Figure 4). We modularize the switch unit into (1) crossbar, (2) router arbiter, and (3) input buffering components when generating the switch units. We synthesize the designs using Vivado HLS (C to RTL) and Vivado 2013.4 (RTL to bitstream). We run cycle-accurate simulations of the NoC in C++ and develop cycle-accurate models of the traffic generators in the PEs.

5.1. Throughput Tests

Under randomly generated communication workloads (uniform random), the NoC is able to sustain throughput only up to a certain input offered rate (≈0.15) as shown in Figure 10(a). Note that the metrics are independent of implementation costs, that is, measurement in terms of cycle counts and number of processors. This saturation behavior is expected due to congestion effects in the NoC at high traffic loads. The deflection torus saturates at a peak sustained rate of around 0.1, the buffered torus at 0.12, and the buffered mesh at around 0.15 for a 10×10 system. This gap is the result of the switching elements in the deflection torus and the buffered torus having half the bandwidth of a mesh switch. This limited freedom constraints the paths that packets can take resulting in performance saturation at lower sustained rates. Furthermore, buffering allows the torus to sustain marginally superior bandwidth compared to the deflection torus due the cost of deflections.

In Figure 10(b), we measure the average latency of each packet traversal on our NoCs as a function of sustained throughput at various PE counts. The deflection torus has generally higher average routing times per packet (2× longer delay over mesh) are



(a) Sustained vs. Offered Throughput

(b) Average Latency vs. Sustained Throughput

Fig. 10. Sustained rates and average latency trends for 10×10 PEs, each point is an *offered injection rate* for uniform RANDOM traffic.

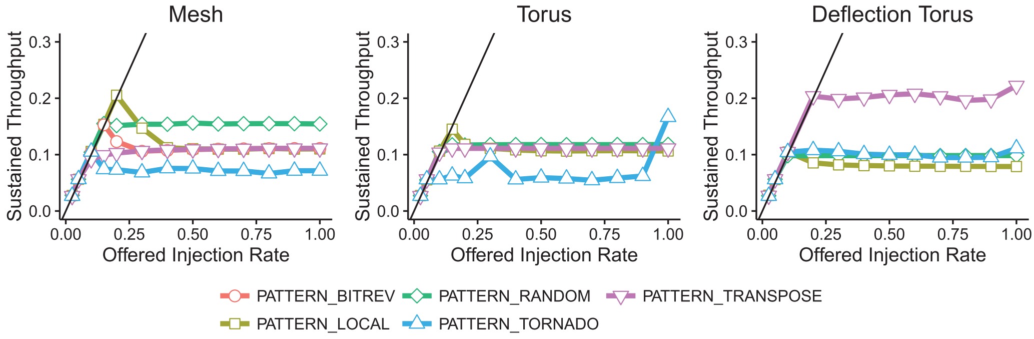
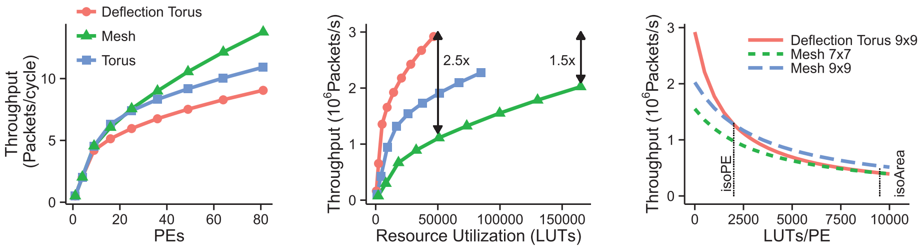


Fig. 11. Sustained vs. offered throughput (packets/cycle/PE) across various traffic patterns. (10×10 PEs, each point is an *offered injection rate*).

again in agreement with the bandwidth results in Figure 10(a). Additionally, we observe a premature degradation in performance at an injection rate of 0.1 for the deflection torus resulting in longer average latencies. This is because the effect of deflections starts to hurt traffic sooner than the effect of buffering delays in the other NoCs. This is expected as minimally buffered deflection routers [Fallin et al. 2012] have been known to provide better performance than deflection routing when implementation of lightweight buffering is not too expensive. Additionally, above an injection rate of 0.2 we observe all NoCs show similar average latency trends. This suggests the cost of waiting in buffers is starting to close the gap with the penalty of deflections in the NoC. Overall, deflections still stay more expensive for latency than buffered routing. Thus, when considering only PE counts, absolute cycles and uniform random traffic patterns, the 2D buffered mesh emerges as a clear winner on both fronts: bandwidth and average latency.

Apart from uniform random pattern, we evaluated the NoCs across other commonly used synthetic routing patterns [Abad et al. 2012] as shown in Figure 11. In this case, the LOCALITY-based pattern runs somewhat well on the 2D mesh up to an offered injection rate of ≈0.2. We even see peaking behavior in sustained throughput before congestion effects degrade performance. The shape of this trend is a well-known characteristic of shared interconnect systems such as Ethernet (CSMA/Aloha [Buchholz 1992]). However, for the deflection torus, the directionality of the routing damages



(a) Throughput vs. PEs

(b) Throughput vs. Area (Only NoC)

(c) Throughput vs. Area/PE

Fig. 12. Evaluating NoC throughput (uniform random traffic, offered rate of 0.5, each point represents a PE config.).

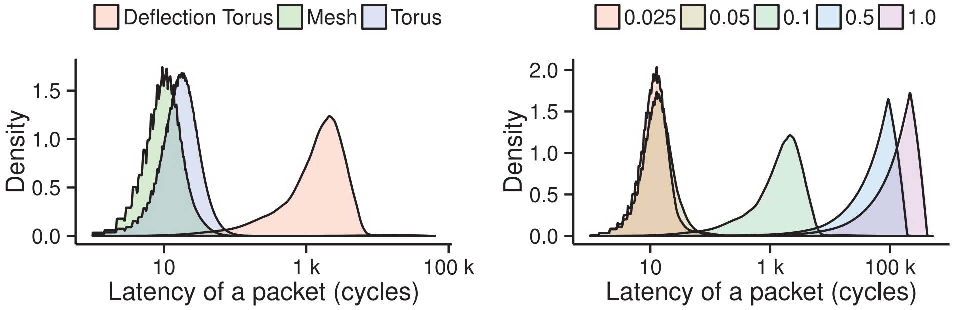
the potential of locality-aware traffic significantly to a sustained rate of ≈0.1. The TORNADO, BITREV, and TRANSPOSE patterns generally route poorly on all topologies with the torus-based networks performing marginally worse than the mesh. The TORNADO pattern exhibits noisy behavior for the torus-buffered NoC due to simulation effects and the alignment of injection times for the packets being unfavorable. At a 100% injection rate, we observe an spike in the throughput of the TORNADO pattern on the torus NoC due to injections being perfectly aligned resulting in fewer conflicts in the network. The TRANSPOSE pattern routes particularly well on the deflection torus due to limited congestion in the NoC. These results further suggest that 2D buffered meshes are superior to torus-based NoCs and deflection routing designs. This seems to counter our original claims of the goodness of the deflection-routed unidirectional torus-based Hoplite NoC. *What is going on here?*

5.2. Area Utilization Considerations

The previous NoC bandwidth and latency characterization may raise the question whether the smaller, faster deflection routed NoCs are worthy of consideration at all. When resource utilization costs are ignored, it is fair to suggest that the higher-bandwidth 2D mesh-based NoCs will outperform their directional torus-based cousins as shown in Figure 12(a). However, when mapped to a real FPGA, the specific implementation costs for mapping the PEs and switches will play a significant role in determining the best NoC.

In Figure 12(b), we consider the real physical implementation costs (both area and clock frequency) of the switches and the PEs. For a fixed NoC area budget allocation to all NoCs, the deflection torus now delivers the highest throughput rather than the 2D buffered mesh. This is because the larger bidirectional, buffered mesh consumes more LUTs and thereby is able to accommodate a smaller sized NoC in the same physical area when compared to the deflection torus. Apart from accommodating larger NoC dimensions, the deflection torus router also runs at a faster clock frequency than the 2D buffered mesh router. Thus, the smaller, faster Hoplite switches compensate for the larger switching capability of the 2D mesh router.

Another key consideration is the relative size of the PE when compared to the size of the switch. We setup an experiment where PE sizes are variable and cover realistic scenarios ranging from small lightweight integer soft processors (1–2K LUTs) to large spatial floating-point dataflow engines (9–19K LUTs). At the smallest theoretical PE size (0 LUTs) the NoC frequency and the PE frequency are considered to be identical. As we increase the PE size, we heuristically degrade the circuit frequency by roughly a nanosecond for every 500 LUT increase in area of PE. We expect that for larger PEs, we can afford to pay the larger costs of the higher-bandwidth bidirectional mesh switches



(a) Hoplite. vs Other NoCs

(b) Effect of Injection Rate, Hoplite

Fig. 13. Density histograms of packet latency for a 10×10 NoC routing uniform random traffic, offered rate = 0.1).

for higher throughput. In this scenario, the larger PE is likely to dictate clock frequency and the resource cost of the switch may matter less against that of the PE itself.

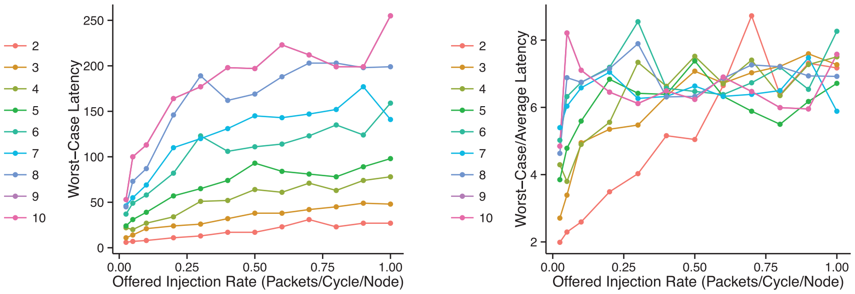
Iso-Area case: When identical FPGA resources are available, a 9×9 deflection torus design matches the cost of a 7×7 mesh design. Due to lower circuit frequency of the mesh, this translates into a lower throughput (measured in packets/s) for the mesh. As we increase PE size, the overall frequency will be dictated by the PE than the switch. In this scenario, the mesh closes the performance gap with the deflection torus at PE sizes that are 9.5K LUTs/PE (crossover marked `iso-Area` in Figure 12(c)). For context, this size roughly corresponds to the resource requirement of a double-precision multiply-add block with associated control logic.

Iso-PE case: We can discount the physical cost of the NoC and assume that the 9×9 NoC of both torus and mesh topologies will require identical resources. This scenario is roughly equivalent to hardening the NoC footprint in both cases and making the NoC costs irrelevant to the comparison. If we now compare a deflection torus with a mesh, then we can conclude that the deflection torus is only superior to the mesh for small PE sizes below 2K LUTs (crossover marked `iso-PE` in Figure 12(c)). Again, for context, a 2K LUT design is closer to a lightweight soft-processor capable of executing a basic ISA.

This suggests that we should prefer deflection torus networks for PEs <2K LUTs (soft processors) when NoCs have been hardened and <9.5K LUTs (spatial floating-point engines) when we must pay for the NoC in soft logic resources.

5.3. Impact on Fairness

A key consideration for latency-critical workloads, is the fairness of the routing algorithm. Fairness directly affects the packet routing latencies, both in the average and worse-case scenarios. This metric is separate from the sustained throughputs shown earlier and is important for real-time applications. For the NoCs considered in this article, we show the the latency histogram of the routed workload in Figure 13(a). Here, we observe that the average packet latency of the mesh and torus is better than that of the deflection torus. This data is for a `RANDOM` workload routed for a fixed duration of 32K cycles (with an offered rate of 0.1 this translates to roughly 3.2K packets per PE). The buffered mesh is able to restrict the worst-case packet latency to ≈29K cycles for an offered rate = 0.1 with random traffic on a 10×10 system. In contrast, the torus and deflection torus allow longer worst case routing delays of ≈30K cycles and ≈33K cycles, respectively, under identical workload conditions. The unfairness for the deflection torus is particularly pronounced and agrees with the average latency trends shown



(a) Impact of Injection Rate on Absolute Worst-Case Latency

(b) Impact of Injection Rate on Ratio of Worst to Average Case Latency

Fig. 14. Understand worst-case latency trends for various N×N configurations (colors represent different N). `RANDOM` workload of 1K packets.

earlier in Figure 10(b). This gap is narrower for smaller injection rate and sufficiently higher injection rates due to underutilization and saturation of the NoC resources, respectively. We illustrate this effect in Figure 13(b). This suggests that routing delays due to misrouted packets ($O(\sqrt{N})$ along a ring) are smaller than the overheads of holding congested packets in FIFO buffers (waiting time). When factoring in physical operating frequency and higher offered rates, this gap will be even worse for the mesh. However, when considering fully featured NoC routers such as CMU Connect with virtual-channel support, the worst-case routing latencies will likely be lower than the ones we report here for the buffered mesh.

We further analyze the effect of injection rates on worst-case deflection costs and its relation to the average case latencies in Figure 14 for a fixed-size `RANDOM` workload of 1K packets per PE. Evaluation on a fixed-size workload reflects a real-world multiprocessor acceleration scenario such as sparse matrix vector multiplication (SpMV). Parallel SpMV computations generate a fixed-sized communication trace tied to the size and sparsity pattern of the matrix. As expected, the cost of worst-case latency increases *linearly* as we increase NoC size N (by as much as ≈20× when comparing 2×2 NoC against the 10×10 NoC). Furthermore, aggressive injection rates have a clear ≈5× effect on worst-case latency at 10×10 NoC size. When comparing worst-case latencies to average case latencies in Figure 14(b), we see a saturation at 6–8× at sufficiently large injection rates. The sustained rates in most configurations saturate above injection rates of 0.2 as observed earlier in Figure 10(a). This explains why the latency trends will be correspondingly affected. There is a smaller deviation from the average latency at low injection rates that would be realistic in most scenarios. However, the steep slopes at large NoC sizes suggest a quick descent into a large deviation from the average-case behavior.

5.4. Qualitative Comparison of Hoplite NoCs to Buffered Virtual Channel 2D Torus Routers

To put our work in context with prior work, we qualitatively compare Hoplite to other NoCs that use buffered virtual-channel based routers.

—**Multi-flit packets.** Routers that process packets spread across multiple flits or perform message segmentation and reassembly of flits, can transmit and receive packets of arbitrary bit widths. Hoplite routers only ever send or receive an atomic message (single flit packets) in one cycle.

- **Virtual channels.** Routers with virtual channels can transmit several classes of traffic, and blocking on one message class does not impede delivery of other message classes. In Hoplite, this can only be achieved with separate physical channels.
- **In-order delivery.** Some prior work routers may be configured to deliver messages in order. Base Hoplite with simple deflecting dimension order router does not guarantee in-order delivery.
- **Fairness.** Some buffered routers, particularly with credit flow control, can achieve relatively fair message ingress, transmission, and delivery. Hoplite's use of simple local flow control on message ingress means an upstream PE that tends to generate a large number of packets can flood its X ring with traffic that may arbitrarily delay ingress of packets from other PEs on the same ring. Similarly Y ring packets from a busy upstream PE may delay delivery of packets sent by PEs on other X rings whose packets cannot enter the necessary Y ring.
- **High NoC utilization.** Compared to Base Hoplite, buffered router NoCs achieve higher sustained rates at high injection rates. As offered traffic (injection rate) increase past a certain threshold, which is traffic and NoC size dependent, Hoplite NoCs see high deflection rates, leading to high message delivery latencies in the NoC itself and, of course, longer queueing at message ingress.

Many of these shortcomings can be mitigated, and since, for a given NoC size and router link width, an FPGA-optimized Hoplite NoC uses several orders of magnitude fewer LUTs, and has lower latency traversing each router, depending on workload, a Hoplite NoC solution may be far superior to prior FPGA soft NoCs.

In particular,

- **Configurability.** Since Hoplite is implemented in an FPGA, its routing function is configurable and specializable to the given workload. One example of this is the ability to reconfigure the routing function for certain regions of the NoC to throttle PE injection rates based on some system-level fairness policy for packet injection.
- **Simplicity of PE interface.** Prior routers impose several hardships on the PE. If the message width is greater than the flit/link width w, then the PE must segment and reassemble packets into/from flits. If multiple incoming flits arrive in an interleaved fashion, then the PE must provide sufficient RAM to buffer and reassemble those partially received packets. For credit flow control routers, PEs may also need to maintain per-VC credit counters for NoC input flits and perhaps even per-VC buffers for NoC output flits.
- **Multi-channel NoCs.** Since Hoplite routers are so frugal with LUTs, it becomes possible to create a system with multiple NoCs. For example, if multiple PEs sent DRAM read requests (64b), then DRAM write requests (576b), and receive DRAM read responses (576b), it is practical to instantiate a 64b NoC alongside a 576b NoC. This has two benefits. It separates traffic into two classes, so DRAM read responses do not block on DRAM read requests, and (by virtue of the two NoCs) it doubles the effective message delivery rate.
- **Improving fairness.** Although this is the subject of future research, there are several simple ways to improve the fairness of message ingress. When too many offered packets are not accepted, or when X→Y turn deflection rates exceed a threshold, either PEs or the routers themselves may throttle chatty ingress PEs to enable other PEs to sent packets at some guaranteed throughput. A simple but not ideal way to accomplish this is to limit the ingress rate at any PE to some threshold determined by simulation to not push the NoC overall into unfair or highly deflecting operating regimes. A more sophisticated approach is to employ a different routing function that enables X router ingress and Y router turns to be scheduled so, for a certain traffic

cadence, clock cycles, or other functions of time, a certain traffic flow from sources to destinations are fair, and deflection-free.

6. CONCLUSIONS

We show how to design Hoplite, an FPGA-friendly NoC using the deflection routing on a unidirectional torus, in a manner that is 3.5× smaller (occupying ≈500 LUTs/switch for HLS-generated design) than a conventional 2D Mesh NoC while running ≈2× faster (≈300MHz). Across a range of statistical workloads while consuming constant area, the deflection routed torus can sustain a 2.5× throughput advantage over the 2D bidirectional, buffered mesh despite requiring half as many wires. We also note that waiting time in buffers of 2D mesh actually delivers packets with a worst-case latency that is marginally better than deflection routed torus due to multiple misroutes. Overall, we expect simpler, cheaper, lighter-weight NoC fabrics to be more amenable for supporting massively parallel FPGA overlays and other custom compute datapaths where switched communication is required. For example, our hand-crafted RTL version of the switch with RLOC constraints for layout occupies 60 LUTs, 100 FFs and runs at 2.9ns.

REFERENCES

P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J A Gregorio. 2012. TOPAZ: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *Proceedings of the 2012 6th IEEE/ACM International Symposium on Networks on Chip (NoCS)*. 99–106.

M. S. Abdelfattah and V. Betz. 2012. Design tradeoffs for hard and soft FPGA-based networks-on-chip. In *Proceedings of the 2012 International Conference on Field-Programmable Technology (FPT)*. 95–103.

Altera. 2011. Applying the Benefits of Network on a Chip Architecture to FPGA System Design. Altera White Paper. (Apr. 2011). Retrieved from https://www.altera.com/en_US/pdfs/literature/wp/wp-01149-noc-qsys.pdf.

Altera Corp. 2015. Arria 10 Core Fabric and General Purpose I/Os Handbook. Retrieved May 2015 from https://www.altera.com/en_US/pdfs/literature/hb/arria-10/a10_handbook.pdf.

Krste Asanović and David Patterson. 2014. Instruction sets should be free: the case for RISC-V. Technical Report No. UCB/EECS-2014-146. (Aug. 2014).

Buchholz. 1992. Comments on CSMA. *IEEE* 802, 11 (1992), 802–11.

Y. Cai, K. Mai, and O. Mutlu. 2015. Comparative evaluation of FPGA and ASIC implementations of bufferless and buffered routing algorithms for on-chip networks. In *Proceedings of the 16th International Symposium on Quality Electronic Design*. 475–484. DOI:http://dx.doi.org/10.1109/ISQED.2015.7085472

W. J. Dally and B. Towles. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the Design Automation Conference, 2001*. 684–689.

C. Fallin, G. Nazario, X. Yu, K. Chang, R. Ausavarungnirun, and O. Mutlu. 2012. MinBD: Minimally-buffered deflection routing for energy-efficient interconnect. In *2012 Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*. 1–10. DOI:http://dx.doi.org/10.1109/NOCS.2012.8

J. Gray. 2014. Keynote 3 2014; The past and future of FPGA soft processors. In *Proceedings of the 2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. 1–1. DOI:http://dx.doi.org/10.1109/ReConFig.2014.7032481

J. Gray. 2016. GRVI Phalanx: A massively parallel RISC-V FPGA accelerator accelerator. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 17–20. DOI:10.1109/FCCM.2016.12

Yutian Huan and A. DeHon. 2012. FPGA optimized packet-switched NoC using split and merge primitives. In *Proceedings of the 2012 International Conference on Field-Programmable Technology (FPT)*. 47–52.

Mike Hutton. 2015. Understanding How the New HyperFlex Architecture Enables Next-Generation High-Performance Systems. Altera White Paper. Retrieved April 2015 from https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01231-understanding-how-hyperflex-architecture-enables-high-performance-systems.pdf.

N. Kapre and J. Gray. 2015. Hoplite: Building austere overlay NoCs for FPGAs. In *Proceedings of the 2015 25th International Conference on Field Programmable Logic and Applications (FPL)*. 1–8. DOI:http://dx.doi.org/10.1109/FPL.2015.7293956

Nachiket Kapre, Nikil Mehta, Michael deLorimier, Raphael Rubin, Henry Barnor, Michael J. Wilson, Michael Wrighton, and Andre DeHon. 2006. Packet switched vs. time multiplexed FPGA overlay networks. In *Proceedings of the 14th IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE, 205–216.

John Kim. 2009. Low-cost router microarchitecture for on-chip networks. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 255–266.

B. S. Landman and Roy L. Russo. 1971. On a pin versus block relationship for partitions of logic graphs. *IEEE Transactions on Computers* 12 (1971), 1469–1479.

G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis. 2010. Evaluating bufferless flow control for on-chip networks. In *Proceedings of the 2010 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)* (2010), 9–16.

Thomas Moscibroda, Onur Mutlu, Thomas Moscibroda, and Onur Mutlu. 2009. *A Case for Bufferless Routing in On-chip Networks*. Vol. 37. ACM, New York, NY.

Michael K. Papamichael and James C. Hoe. 2012. CONNECT: Re-examining conventional wisdom for designing nocs in the context of FPGAs. In *Proceedings of the ACM/SIGDA International Symposium*. ACM Press, New York, NY, 37.

Xilinx Inc. 2015. 7 Series FPGAs Configurable Logic Block User Guide. Retrieved February 2015 from http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB .pdf.

Xilinx Inc. 2016a. *7 Series FPGAs Configurable Logic Block User Guide UG474*. Technical Report. Xilinx Inc.

Xilinx Inc. 2016b. *UltraScale Architecture Configurable Logic Block User Guide UG574*. Technical Report. Xilinx Inc.

Received April 2016; revised November 2016; accepted December 2016