Hoplite: Building Austere Overlay NoCs for FPGAs

Nachiket Kapre
Nanyang Technological University
50 Nanyang Avenue, Singapore
nachiket@ieee.org

Jan Gray
Gray Research LLC
Bellevue, WA
jsgray@acm.org

***Abstract*—Customized unidirectional, bufferless, deflection-routed torus networks 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 FPGA resources to both NoCs) for uniform random traffic. We present Hoplite, an efficient, lightweight, fast FPGA overlay NoC that is designed to be small and compact by (1) eliminating input buffers, and (2) reducing the cost of switch crossbar that have traditionally limited speeds and imposed heavy resource costs in conventional FPGA overlay NoCs. We implement bufferless deflection routing cheaply, requiring the generation of only output multiplexer controls and no backpressure handshakes. Additionally, we use directional channels that help reduce crossbar cost by restricting the number of inputs to the crossbar to three instead of four. 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 a prototype RTL version of our switch with RLOCS that requires only 60 LUTs and 100 FFs per router and runs at 2.9 ns.**

I. Introduction

It is an important and popular fact that the design and engineering of 2D Mesh-based NoCs in modern SoCs and multi-processing fabrics is critical for performance and energy efficiency. For instance, we have traditionally assumed that 2D bidirectional meshes are the most efficient network instead of the third most efficient[1]. 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 is possible to support *overlay* NoCs that are configured on top of these programmable LUT and routing resources. They are adaptable, customizable and tuneable, but have generally suffered from high resource requirements when mapped to FPGAs. For example, the 32b CMU CONNECT router [11] takes up 1.5K LUTs@9.6 ns while the 32b Penn Split-Merge router [6] occupies 1.7K LUTs@4.5 ns. The key culprits in these designs is the inability to cheaply implement small FPGA onchip buffers and 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 VCs (virtual channels) 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. Some recent academic studies have investigated the potential for hard NoCs [2] 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.

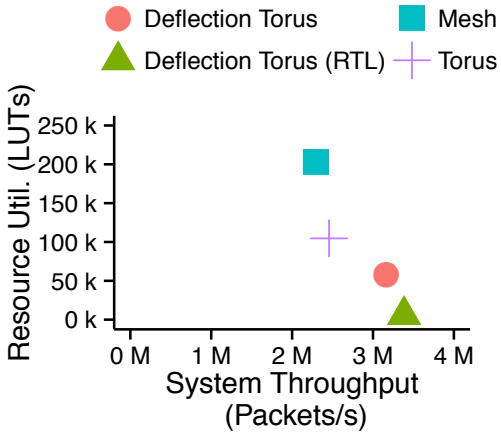


Fig. 1: Area-Throughput Tradeoffs for various switches in a 10×10 NoC (Virtex-6 LX240T FPGA) under random traffic and offered injection rate of 0.5 packets/cycle/PE. Showing Hoplite hand-crafted RTL implementation results in addition to switches generated from HLS tools.

In this paper, we investigate the latency and throughput characteristics of constructing overlay NoCs (which we call Hoplite) using bufferless deflection-routed torus [10] on top of a modern FPGA fabric. This design style simplifies the engineering of FPGA overlay networks in two fundamental ways: (1) The use of unidirectional torus instead of a bidirectional network enables simpler switching and control logic in the design. (2) The use of deflection routing elegantly handles

[1] adapted from *The Hitchhiker's Guide to the Galaxy* by Douglas Adams

conflicts within the NoC without resorting to handshake-based designs that add complexity to the router arbitration logic. This reduction in complexity, simplifies the circuit design sufficiently thereby enabling faster clock frequency of the NoC fabric. Hoplite was first devised to efficiently interconnect hundreds of soft processors in the Phalanx system [5], but is broadly applicable to other NoC roles. In Fig. 1, we preview our preliminary results for the various switches we developed for an 10×10 NoC. Despite the reduction in available wires, the 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 PEs (PE is a Processing Element, like a soft processor, or a communicating IP block).

We make the following key contributions in this paper:

- FPGA-focussed design and characterization of buffered mesh, buffered torus and bufferless deflection router architectures on the Xilinx ML605 platform.
- Quantification of the engineering tradeoffs in fairness and latency distribution when using deflection routing for various workloads.
- Performance evaluation and quantification of the NoC under various statistical traffic patterns.

II. Background

A. 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 [1] 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).

B. Related Work

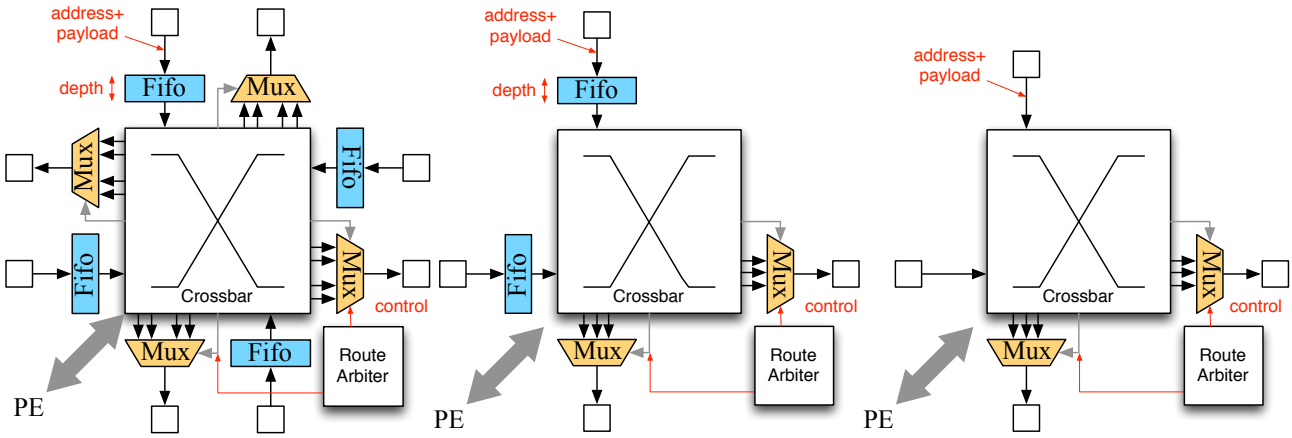
While we still use them today, bus-based shared networks were common in the resource starved silicon-poor era of the late 1990s-2000s. As wiring delays overwhelmed gate delays and the effects of Rent's rule manifested in system-level communication requirements, it was no longer adequate to rely purely on busses alone. Shared, switched networks that route packets instead of wires [4] 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 [11] 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. The Penn Split-Merge [6], [7] architectures throws out the ASIC-inspired design methodology in favor of a simpler, VC-free, handshake-based, pipelining-friendly FPGA switches. However, both these designs still spend 30–40% of their resources on crossbar switching and 20–40% on buffering requirements while operating between 90–200 MHz. The Split-Merge router can be run faster up to 310 MHz 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 [7] 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 [8] and bufferless deflection routers were proposed in [10] as a way to address the rising buffer costs (area, delay, power) in ASIC-based NoCs for multi-processing workloads with multi-flit packets. This was refuted in [9] where energy benefits were found to be minimal and latency and bandwidth benefits were superior for the buffered designs. We investigate the potential for custom deflection routers for specific application scenarios for flit-level routing in the context of FPGAs where buffers and crossbars are expensive and slow. We summarize key resource utilization results for our proposed design in Table I and contrast it against existing FPGA overlay NoC routers.

TABLE I: Comparing FPGA-based NoC routers (32b payloads, Xilinx Virtex-6 LX240T).

Router	LUTs	FFs	Cycle (ns)
CONNECT 2VCs	1562	635	9.6
Split-Merge DOR	1785	541	4.5
Hoplite NoC written in Vivado HLS (This paper)			
Mesh	2035	1669	6.8
Torus	1046	949	4.8
Deflection Torus	576	570	3.1
Hoplite Prototype Hand-Crafted RTL (This paper)			
Deflection Torus	60	100	2.9

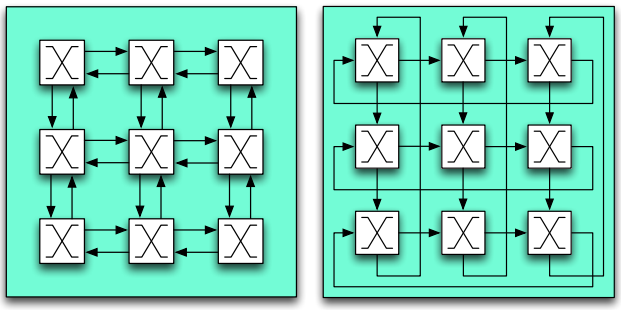
C. Network-on-Chip Design

While we can organize the switches in various topologies, we consider designing overlay NoCs on the FPGA in a 2D layout using mesh and torus topologies as shown in Fig. 2.



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

Fig. 3: Switch Organizations evaluated in this study. Bidirectional Buffered Mesh needs FIFO buffers, and wires in each dimension, Directional Buffered Torus doubles bisection bandwidth but still requires buffering, while Directional torus eliminates buffers as well.



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

Fig. 2: Topology of NoCs considered in this study – A Bidirectional Mesh and a Directional Torus.

Data is routed over the network in a packet-switched style. While generally, each packet can composed of multiple flits, we consider finer-grained load-store style traffic generated in a multi-processing 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), West-Side First (WSF). For simplicity, we support DOR routing as it is cheap and straightforward to implement on the FPGA logic similar to the Penn and CMU routers.

III. Deflection Routed Torus NoC

In this paper, 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 Fig. 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 processing elements (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, (3) buffering and multiplexing of IOs. The implementation complexity of the crossbar is $O(N^2)$ where N is number of IOs. For the 2D bidirectional mesh, N=4 (five inputs and four outputs, one connection for the PE, four connections for each of the four directions in the mesh, 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 and 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 are implemented using SRL blocks and registers with bypass paths to reduce queueing delay within the switch. This alone is sufficiently complex to preclude 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.

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, two connections for vertical and horizontal lanes). We tabulate these reductions in Table II.

Buffering cost can be eliminated by adopting a deflection

routing technique [10] that mis-routes packets in presence of a conflict. In buffered switches, FIFOs are used to hold incoming packets when the desired outgoing port is not available *i.e.* another packet is using the port in that cycle. In contrast, in deflection-routed networks, we intentionally mis-route 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 number of incoming and outgoing links in the switch, we can always guarantee an outgoing slot for incoming packets. For the outgoing PE port, we mis-route 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 *mis-routed*. While implementation of the mis-routing policy is substantially simpler on the FPGA (see Table II), packets may now take multiple trips in the NoC due to mis-routing 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 mis-routing overheads.

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

IV. Methodology

1) Switch RTL for simulation: We generate RTL for our various switch configurations using synthesizable C++ descriptions of the switch with Vivado High-Level Synthesis. 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 (See Section IV-3).

2) Switch RTL for synthesis: Despite HLS optimizations, a bottom-up RTL description of the switch still delivers significantly better results. In Fig. 5, we show a prototype proof-of-concept 10×10 RTL deflection torus NoC implemented on a Virtex-6 LX240T (ML605). This functionally-verified Verilog design consumes 60 6-LUTs and 100 FFs per router, for a total of 6,150 6-LUTs and 14,800 FFs (including test logic) while operating in excess of 300 MHz when densely floorplanned. This is less than 4% logic utilization of the device. The RTL switch provides several improvements over the HLS version:

- It employs a 3-input *2-output* switch instead of the 3-output switch of the HLS version. An output packet for the client PE appears on (and shares) the Y output. Under load this may cause additional packet deflections when valid input packets XI and YI vie to exit on the shared Y output link.
- The partial crossbar switch achieves two link output bits per 6-LUT, using fracturable LUTs.
- The datapath is technology mapped and floorplanned using `LUT_MAP` and `RLOC` constraints.
- For a 10×10 NoC, 4b X and 4b Y addressing bits suffice and reduce DOR arbiter complexity. The packet payload is intentionally widened from 32b to 40b to keep (comparable) 48b wide packets and links and preserve wiring requirements in the mapping.

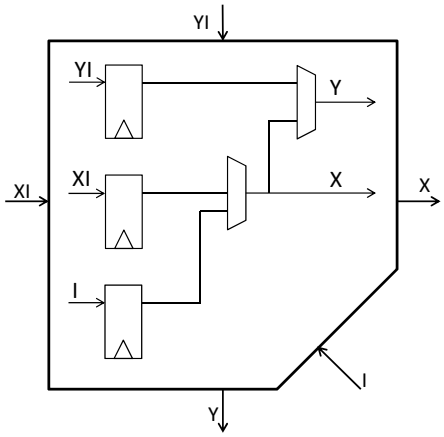


Fig. 4: Xilinx partial crossbar switch using dual 2-1 muxes.

The RTL is parameterized to target Altera or Xilinx FPGAs. Modern Altera FPGAs [3], by virtue of their fracturable 8-input ALM (Adaptive Logic Module), can implement a one-bit wide slice of a 3-2 crossbar (dual 3-1 muxes) per ALM. Modern Xilinx FPGAs [12] provide (less flexible) fracturable 6-LUTs which may be used as a pair of 5-input LUTs, so long as the two LUTs take the same five input signals. Therefore an 8-bit-wide 2-1 multiplexer (e.g. with a common select line) can be implemented in a four 6-LUT "slice". Two such 2-1 muxes can be cascaded to build a partial crossbar from XI, YI, I inputs to X, Y outputs (Figure 4). Thus eight 6-LUTs in total can switch 8 X-outputs and 8 Y-outputs. This partial crossbar datapath does not implement every transfer function of the three inputs, but it provides a useful subset: YI→Y and XI→X, or XI→Y, or YI→Y and I→X, or I→Y.

```
(* KEEP_HIERARCHY="true" *)
module m2x48(input sel,
             input [47:0] a,
             input [47:0] b,
             output [47:0] o);
  (* RLOC="X0Y0" *)
  m2x8 m0(.sel(sel),
          .a(a[7:0]),
          .b(b[7:0]),
          .o(o[7:0]));
  ...
  (* RLOC="X0Y5" *)
  m2x8 m5(.sel(sel),
          .a(a[47:40]),
          .b(b[47:40]),
          .o(o[47:40]));
endmodule

(* KEEP_HIERARCHY="true" *)
module m2x8(input sel,
            input [7:0] a,
            input [7:0] b,
            output [7:0] o);
  genvar i;
  generate for (i=0; i<8; i=i+1) begin : ms
    (* RLOC="X0Y0" *)
    m2 m(.sel(sel),
         .a(a[i]),
         .b(b[i]),
         .o(o[i]));
  end endgenerate
endmodule

(* LUT_MAP="yes" *)
module m2(input sel,
            input a,
            input b,
            output o);
   assign o = (~sel&a) | (sel & b);
endmodule
```

Listing 1: 48-bit 2-1 mux RPM – a column of 6 slices

Xilinx hierarchical RPM (relationally placed macro) implementation is not well known. Listing 1 presents the Verilog source of a 48-bit-wide floorplanned 2-1 mux RPM used in

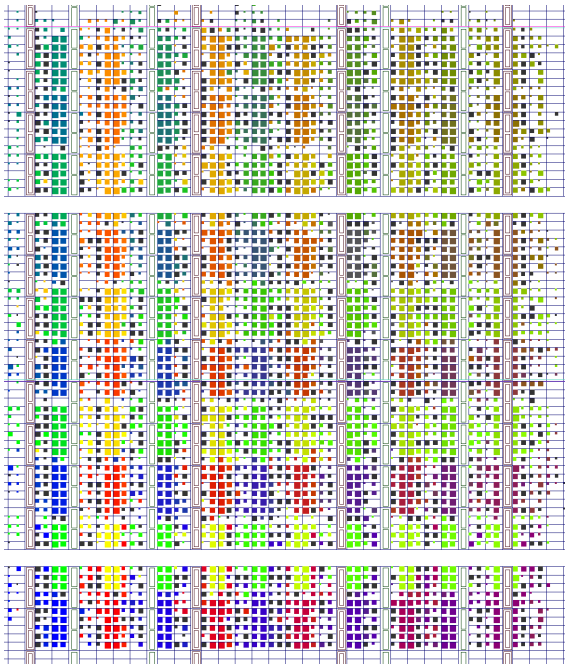


Fig. 5: Folded torus layout of 10×10 prototype Hoplite NoC on a small portion (or corner) of LX240T (4% LUTs-FFs, 2.9ns clock). Here, each color represents one router in the 2D folded torus.

Figure 5. The `LUT_MAP` constraint on `m2` forces it to be mapped to a LUT; the `RLOC="X0Y0"` constraint causes eight `m2`s to be implemented in the same slice of four 6-LUTs; and `RLOC` constraints on the six instances `m0`..`m5` of `m2x8` cause them to be placed one above another in a column of six slices.

The design composes two `RLOC`'d `m2x48s` into a Hoplite router switch RPM, and composes 10×10 Hoplite routers into a 2D torus NOC of placed router RPMs. In Figure 5, the overall floorplan of the routers has been interleaved, or folded, to keep all link wires relatively short. Otherwise there could be many long or die-crossing wires, for example in X links from (9,y) to (0,y) or in Y links from (x,9) to (x,0).

Since we do not have equivalent hand-optimized RTL for buffered switches, we use the HLS results for a fairer relative comparison (making the deflection router look much worse than is possible in RTL).

3) Workloads: We evaluate our NoC under various statistically generated workloads commonly used within the NoC community [1]. 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 of 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

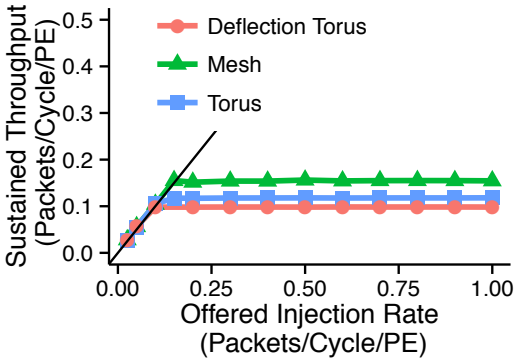


Fig. 6: Sustained vs. Offered Throughput in the NoCs (10×10 PEs, each point is an *offered injection rate*).

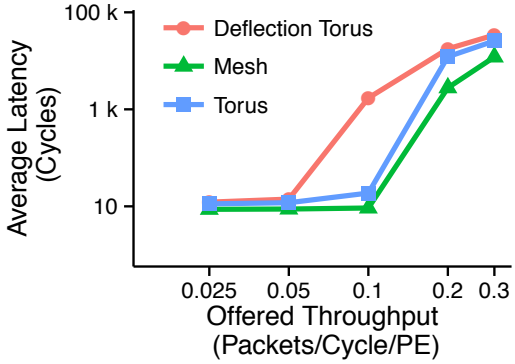


Fig. 7: Average Latency vs. Sustained Throughput Plots (10×10 PEs, each point is an *offered injection rate*).

time spent attempting sending packets) and 1.0 (100%). We measure *sustained rates* (or sustained 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 attempt to enter NoC and actual entry).

V. Results

We now 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 (HLS switches) on performance.

A. 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 Fig. 6. This expected behavior 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 deflection torus and the buffered torus having half the bisection bandwidth of the mesh. Furthermore, buffering allows the torus to sustain marginally superior bandwidth compared to the deflection torus due the cost of mis-routing.

In Fig. 7, 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 again in agreement with the bandwidth results in Fig. 6. Additionally, we observer an earlier degradation in performance at an injection rate of 0.1 for the deflection torus resulting in longer average latencies. 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 [1] as shown in Fig. 8. In this case, the `LOCALITY`-based pattern runs somewhat well on the 2D mesh up to an offered injection rate of ≈0.2. However, for the deflection torus, the directionality of the routing damages 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.

B. 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. When PE costs and switch 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 Fig. 9a. However, when mapped to a real FPGA, the specific implementation costs will plan a significant role. In Fig. 9b, we consider the real physical implementation costs (both area and clock frequency) of the switches and the PEs. We use the placement-and-routed resource results from our HLS-based switch to retain consistency and continuity across designs. For a fixed NoC area budget, the deflection torus delivers the highest throughput in our comparisons. 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. Additionally the faster clock frequency of the deflection torus further improves performance. Thus, the smaller, faster switches compensating for the larger switching capacity of the 2D mesh router. Another key consideration is the relative size of the PE when compared to the size of the switch. As we increase the size of PEs, we (heuristically) consider a drop in the circuit frequency of roughly a nanosecond for every 500 LUT increase in area of PE. Our PE sizes cover realistic scenarios ranging from small lightweight integer soft processors to large spatial floating-point dataflow engines. We

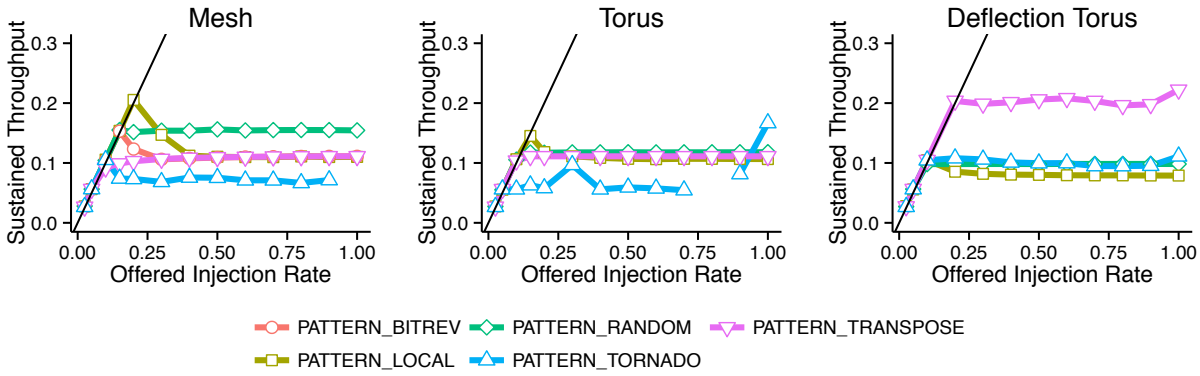
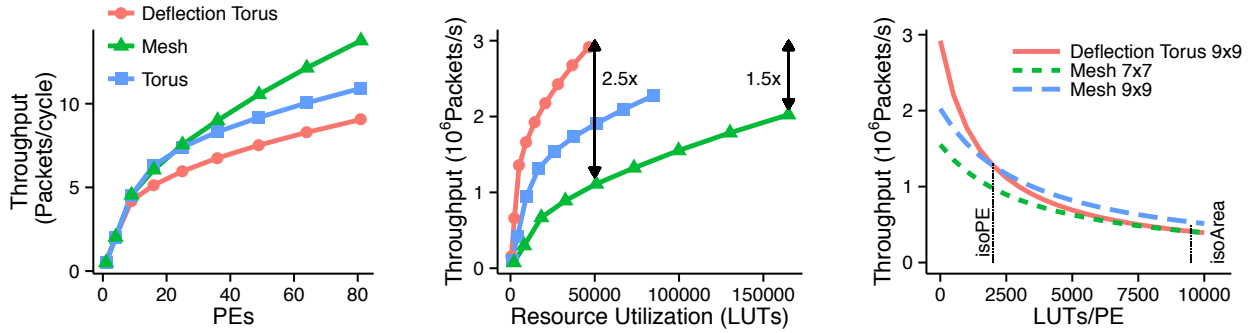


Fig. 8: Sustained vs. Offered Throughput (packets/cycle/PE) across various traffic patterns (10×10 PEs, each point is an *offered injection rate*).



(a) Throughput vs. PEs (b) Throughput vs. Area (Only NoC) (c) Throughput vs. Area/PE

Fig. 9: Evaluating NoC Throughput (Uniform random traffic, offered rate of 0.5, each point represents a PE config.).

expect that for larger PEs, we can afford to pay the larger costs of the higher-bandwidth bidirectional mesh switches for higher throughput (packets/cycle). 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. Surprisingly, even in this scenario, the mesh only starts to close the gap with the deflection torus at PE sizes that are 9.5K LUTs/PE. For context, this size roughly corresponds to the resource requirement of a double-precision multiple-add block with associated control logic. A 9×9 deflection torus design matches the performance (throughput) of a 7×7 mesh design using equal FPGA resources (marked `iso-Area` in Fig. 9c). If we discount the physical cost of the NoC which is somewhat analogous to embedding a hard NoC, and compare a 9x9 deflection torus and mesh designs, we can conclude that the deflection torus is only superior to the mesh for small PE sizes below 2K LUTs (marked `iso-PE` in Fig. 9c). 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 for a hard NoC design (soft processors) and <9.5K LUTs for an overlay NoC (spatial floating-point engines).

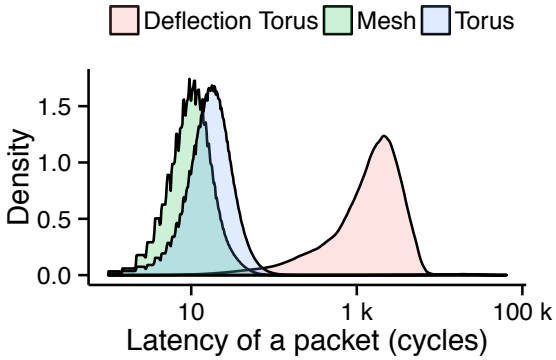


Fig. 10: Density Histogram of Packet Latency for a 10×10 NoC routing uniform random traffic, offered rate=0.1).

C. Impact on Fairness

A key consideration for latency-critical workloads, is the fairness of the routing algorithm. As we observe in the latency histogram shown in Fig. 10, the average latency of the mesh and torus is better than the deflection torus. Furthermore, the

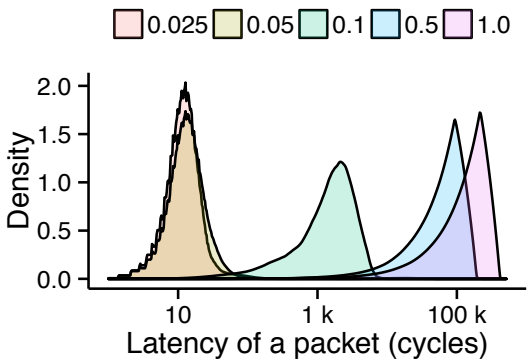


Fig. 11: Density Histogram of Packet Latency for a 10×10 Deflection Torus NoC routing uniform random traffic under various offered injection rates.

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 much longer worst case routing delays of ≈30K cycles and ≈33K cycles respectively under identical workload conditions. The unfairness for the deflection torus at an injection rate of 0.1 is particularly pronounced and agrees with the average latency trends shown earlier in Fig. 7. This gap is not this large for smaller injection rate and sufficiently higher injection rates due to underutilization and overutilization of the NoC resources respectively. We illustrate this effect in Fig. 11.

D. Summary

We can summarize key findings of our experiments here:

- **Identical PE counts and Identical Clock Period**: If physical costs and implementation frequency of the PE and the switch are ignored, the 2D buffered bidirectional mesh network with 2× bisection bandwidth is the clear winner. This finding is confirmed when measuring sustained rates (Fig. 6) as well as average latency (Fig. 7).
- **Identical Resources and Observed Clock Period**: When physical mapping costs and circuit frequency are considered, the deflection torus network is as much as 3× faster at identical area usage of 50K LUTs (Fig. 9).
- **Statistical Patterns**: When we stress-test the NoCs with hard-to-route patterns (`TORNADO`, `BITREV`, and `TRANSPOSE`), the gap between the NoCs is not particularly significant. For `LOCAL` traffic patterns, the Mesh deliver better performance due to higher switching capacity at identical PE counts but still gets beat by the deflection torus when considering area (Fig. 8).
- **Fairness**: The effect of waiting time in buffers is only marginally better than the effect of mis-routing on deflection routed network (Fig. 10).

VI. 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 (≈300 MHz). 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 mis-routes. 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

[1] P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J. A. Gregorio. TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers. *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 99–106, 2012.

[2] M. S. Abdelfattah and V. Betz. Design tradeoffs for hard and soft FPGA-based Networks-on-Chip. In *Field-Programmable Technology (FPT), 2012 International Conference on*, pages 95–103, 2012.

[3] Altera Corp. Arria 10 Core Fabric and General Purpose I/Os Handbook, May 2015.

[4] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689, 2001.

[5] J. Gray. Keynote 3 2014; the past and future of fpga soft processors. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–1, Dec 2014.

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

[7] 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*, pages 205–216. IEEE, 2006.

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

[9] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis. Evaluating Bufferless Flow Control for On-chip Networks. *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 9–16, 2010.

[10] T. Moscibroda, O. Mutlu, T. Moscibroda, and O. Mutlu. *A case for bufferless routing in on-chip networks*, volume 37. ACM, New York, New York, USA, June 2009.

[11] 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*, page 37, New York, New York, USA, 2012. ACM Press.

[12] Xilinx Inc. 7 Series FPGAs Configurable Logic Block User Guide, February 2015.