# Implementing FPGA overlay NoCs using the Xilinx UltraScale memory cascades

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

*Abstract*—
**We can enhance the performance and efficiency of deflection-routed FPGA overlay NoCs by exploiting the cascading feature of the Xilinx UltraScale BlockRAMs. This allows us to (1) harden the multiplexers in the NoC switch crossbars, and (2) efficiently add buffering support to deflection-routing. While buffering is not required for correct operation of a deflection routed NoC, it can boost network throughputs for large system sizes under heavy load and allow functional support for fixed-length, multi-flit NoC traffic. Since the multiplexer controls of the cascaded RAMs can be driven from user-logic, the NoC routing function can be implementing in LUTs while the data is steered across the dedicated cascade multiplexers and links. Thus, our approach uses hard resources in the BlockRAM architecture to absorb the bulk of the cost of a NoC in the form of crossbar multiplexing, as well as packet queuing. For the XCVU9P UltraScale+ FPGA, we show how to map the 72b Hoplite NoC router at a cost of 3 FIFO blocks, 64 LUTs, and 40 FFs per switch while operating at ≈727 MHz (400 MHz in 60×12 grid). This reduces LUT count by 1.4× and FF cost by 2× over a pure LUT-based implementation while also being 1.2× faster. For uniform `RANDOM` traffic, we boost throughputs of a 16×16 NoC by 50–60%, reduce worst-case packet latency by ≈40%, and lower energy use by 10–40% over classic bufferless deflection-routing at injection rates of 15–20% and higher with 16-deep buffers. When compared to hard NoC router designs, our BRAM-based soft NoC also closes the area gap to under a factor of two instead of the 20–23× gap claimed in earlier studies.**

## I. Introduction

FPGA-based overlay NoCs (networks-on-chip) have a long and rich history of design evolution that has improved throughputs, latencies, and resource requirements. These NoCs, called soft NoCs or overlay NoCs, are implemented on top of an FPGA fabric using programmable resources such as LUTs, FFs, and general purpose interconnect. Some NoCs such as [13], [8] are specifically tailored for FPGA embodiment so as to exploit the wire-rich FPGA substrate with high-arity topologies [13], or to deliver high-performance routers through modularity and localization of control [8]. More recent NoCs such as Hoplite [10] are substantially smaller than these earlier designs by compromising certain NoC performance metrics through the use of bufferless deflection routing. The high resource costs or poor performance of soft NoCs makes it attractive to consider hard NoC routing infrastructure in FPGAs [1]. Unlike soft NoCs that use existing FPGA resources, hard NoCs use dedicated silicon resources to implement specific NoC routers as well as using dedicated

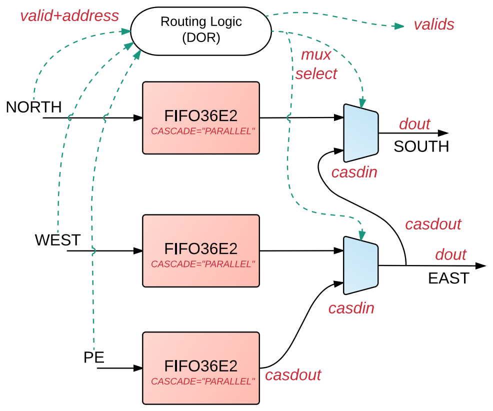


Fig. 1: Embedding Hoplite NoC router functionality within the **PARALLEL** FIFO cascade mode of the Xilinx UltraScale FPGA.

interconnect links between the routers that does not interfere with the programmable resources of the FPGA. While the performance, and efficiency advantages of a hard NoC are evident, the engineering cost of adding the NoC fabric to the FPGA and associated CAD tools may be years away from actual product deployment. Additionally, the architecture of the hard NoC may be inflexible for varied FPGA workloads. If a user application needs NoC functionality today, or is unlikely to benefit from a fixed configuration of a hard NoC in the future, we need to satisfy these user requirements by improving the cost and performance of a soft NoC. In this paper, we consider the use of UltraScale BlockRAM cascades (1) to reduce soft logic resource requirements of the Hoplite NoC, (2) to demonstrate improvements in throughput through the use of buffering instead of deflections when possible, and (3) to close the gap with clean-slate hard NoC routers by exploiting existing hard resources in the FPGA. We show a high-level diagram of our proposed router in Figure 1.

**Use of cascade resources**: We consider the use of RAM cascades in the Xilinx UltraScale FPGAs to support a portion of NoC functionality using dedicated resources. Xilinx UltraScale FPGAs [3] use a novel architecture tailored to the new 20 nm manufacturing technology. Among the various changes to the FPGA organization, it is now possible to cascade BlockRAM resources to construct larger RAM structures and FIFOs from small 18 Kb and 36 Kb blocks using specialized programmable structures. This is supported with the help of fast vertical cascade links which are similar to carry chains between LUTs and cascading links between DSP48 blocks. The RAM tile internally provides dedicated multiplexers to steer data busses to adjacent RAM blocks. Importantly, the multiplexer selection controls and read-write logic is exposed to user logic. *How does the generation of these controls affect the implementation cost of the deflection routing function?*

**Avoiding deflections**: We also consider the use of FPGA BRAMs for buffering to reduce the overheads of deflections in Hoplite. Hoplite [10] is small, lean design that requires a single Xilinx 6-LUT for each bit of the NoC link along with a few extra LUTs for route decoding. However, the penalty of deflections can be high and result in lower throughputs at large system sizes and under heavy load. A known technique to overcome this limitation is to use buffering within the NoC to absorb conflicting packets and to reduce the deflection counts in the network [7]. For the Xilinx UltraScale FPGA, when using the BRAM cascade, the RAMs can be configured as FIFOs with hardened control logic to serve precisely this role. *How much buffering is adequate to balance the positive improvement in throughput with any negative impact on packet latency due to head-of-line blocking [6] which is a known problem with buffered NoC routers?*

We enumerate the important contributions of this paper:

- We modify the Hoplite NoC router to use dedicated hard BlockRAM resources and associated cascading features on the Xilinx UltraScale FPGA. This requires redesigning the routing function to account for the unique pipeline structure of the cascade. We also generate FIFO access controls.
- We quantify the impact of FIFO buffering on the NoC links to hold packets in conflict instead of deflecting them along long round-trips in the NoC lanes. We conduct a performance analysis of the NoC under various synthetic workloads, system sizes, and injection rates.
- We map Hoplite NoC configurations spanning the complete chip fabric on the XCVU09P UltraScale+ FPGA card using XDC location constraints for the BRAMs alone.

## II. Hoplite Review

In this section, we describe the features of the Hoplite router implemented using fracturable Xilinx 6-LUTs. Abundant details of the Hoplite NoC switch are available in [10], [9], [4], [5], but we summarize important characteristics here.

Hoplite [10] is a lightweight, FPGA-optimized NoC that uses bufferless deflection routing, unidirectional torus topology, and port sharing to reduce the resource costs of a NoC in a manner well-suited to an FPGA implementation. Hoplite has been shown to outperform classic buffered FPGA NoCs such as CMU Connect [13] and the Penn Split-Merge [8] on performance by 1.5–2.5× (under certain conditions). It also lowers the resource needs by 20–30× be eliminating buffering and complex controls. When using deflection routing, all arriving packets must be routed to some outbound destinations. Buffering is not mandatory as all arriving packets are guaranteed to be processed, sometimes along an undesirable

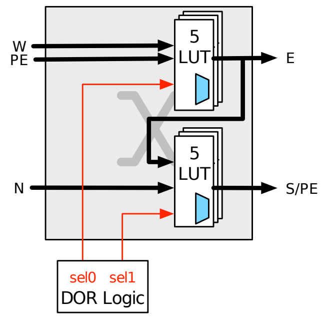


Fig. 2: Hoplite NoC switch design optimized for fracturable dual 5-LUT Xilinx FPGA CLB organization.

deflection. The performance penalty of deflection is the price we pay for a low-cost FPGA implementation.

In Figure 2, we show a resource-optimized version of the Hoplite switch that uses internal multiplexer chaining to better fit the Xilinx fracturable LUT CLB organization. In this design, we have two NoC inputs from the torus (North, and West) and two NoC outputs (South, and East). We also have a separate PE (Processing Element) connection that is allowed to inject new packets and remove packets from the network. The connection to the PE can be back-pressured due to contention within the network. As Hoplite uses simple Dimension-Ordered Routing (DOR), packets are allowed to turn from the X-dimension to the Y-dimension but not vice versa *i.e.* packets arriving on the North input cannot turn East. This observation makes it possible to use a multiplexer cascade as shown in Figure 2 instead of a full internal switching crossbar that is typically required in NoC switches. Under this arrangement, the first multiplexer level chooses a packet between the West input and the PE input. This may either exit along the East output or be routed to the South exit through the next multiplexer stage. This second multiplexer stage arbitrates between the North input and this previous multiplexer output. The output of this multiplexer is destined for the South exit, or an exit to the PE. This port sharing is disambiguated with appropriate valid signals. A limitation of this approach is that the first multiplexer stage can potentially block packets from the PE whenever packets are present on the WEST input. Additionally, output sharing for the South and PE exits can also cause performance loss. We can eliminate this bottleneck by supporting a full crossbar without the multiplexer cascade (doubling of LUT cost), and by providing separate output for both South and PE exist (additional bank of LUTs required, tripling cost of original design). For many designs, the simple output shared, fractured LUT implementation is often adequate.

When mapping the Hoplite router to hardened FPGA resources we look for circuit structures that can embed the functionality of the design in Figure 2. In Hoplite-DSP [4], it was possible to support full crossbar operation of the Hoplite router inside a DSP48 block at the cost of doubling the internal DSP operating frequency. Instead, for this paper, we investigate the UltraScale RAM cascade configuration which (1) naturally permits the fractured design, (2) does not steal

DSP resources from the user, and (3) is able to use the BRAM FIFOs for packet buffering to reduce deflections.

## III. NoC Design with UltraScale Cascades

In this section, we show how to configure the UltraScale BlockRAM cascades to support Hoplite NoC operation.

### A. Cascade Modes

As described in UG573 [14], UltraScale memory blocks (18Kb or 38Kb blocks) can be internally cascaded to construct complex RAM structures without needing programmable logic or routing resources. For the BlockRAMs, the data ports can be cascaded either serially or in parallel to build a variety of memory configurations. This configurability can be used to configure deeper memories (larger than 18/36Kb) or systolic arrangements as required by the user design. When configured as FIFOs, cascading is useful to construct deeper FIFOs or to combine data from multiple FIFOs into a single output stream. It is this Parallel FIFO cascade mode that is of relevance to the Hoplite NoC design.

As shown in Figure 3, we use three RAM blocks configured as FIFOs to implement the Hoplite NoC switch with the `CASCADE_ORDER` attribute set to **PARALLEL**.

- The lowermost FIFO is connected to the PE and the existing handshake signals are connected to the FIFO write enable, and FIFO full flags. The FIFO is useful to absorb packets coming from the PE that are unable to enter the NoC due to network congestion. The CASDO output is the Cascade Output port that is directly wired to the CASDI input of the middle FIFO block.
- The FIFO in the middle is connected to the WEST input from the NoC and also uses the CASDI (Cascade Input) connection from the lower FIFO. The multiplexer select is driven by the DOR (Dimension Ordered Routing) logic to determine which packet is forwarded to the output EAST port. The multiplexer (shown in blue) is the first level of multiplexing shown earlier in Figure 2.
- The topmost FIFO is connected to the NORTH input from the NoC, as well as the CASDI (Cascade Input) connection from the middle FIFO. As before, the DOR logic drives the multiplexer selection. The multiplexer (shown in blue) is the second level of multiplexing shown in Figure 2.

### B. Adapting Routing Function for Buffering

Deflection-routed NoCs have to route each incoming packet to available output ports. For an FPGA implementation scenario no buffering is provided to (1) reduce distributed RAM costs when implementing on top of FPGAs and (2) to simplify the handshake signaling to a single valid per link. Buffered switches require complex flow control that is expensive on FPGA fabrics and often unnecessary for multiprocessor message-passing workloads with other mechanisms to achieve synchronization and ordering.

Hoplite uses Dimension Ordered Routing (DOR) where packets are allowed to change from X to Y dimension but not vice versa to avoid deadlock. If the desired output port is

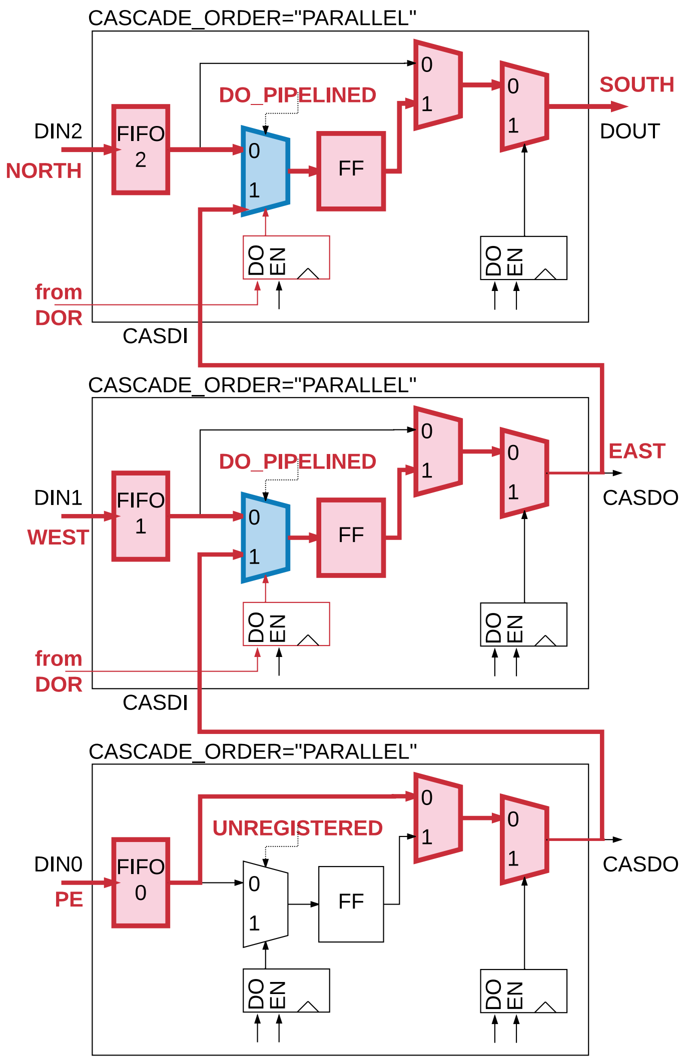


Fig. 3: FIFO cascade **PARALLEL** mode shown overlaid with (1) ports labeled to match Hoplite router, and (2) data flow within the cascade highlighted to match router.

not available, then the incoming packet must deflect along an undesired output port. There is no flow control, and all arriving packets must be forwarded to some output port. However, the penalty of deflections limits throughputs (packets/cycle) particularly at high injection rates and large system sizes. It is possible to use buffering along the links to reduce the number of deflections in the network. When there is a conflict at an output port, one of the conflicting packets can be buffered in the FIFOs instead of being penalized with a deflection. The DOR routing function needs to be adapted to generate the FIFO read/write signals to account for this condition. When the FIFOs are full and the desired output ports are still unavailable, we have to resort to conventional deflection routing until the congestion clears. Even when using FIFOs, no flow control information is allowed to cross the switch boundary *i.e.* a switch cannot communicate FIFO full/empty status to connected routers. This simplifies the logic design of the arbiter and avoids long-distance flow control signals on the FPGA. In the original Hoplite design, the PE can already

TABLE I: Hoplite+Buffering Routing function. Added support for queuing of deflection-vulnerable packets.

| **Inputs** (arriving packet)[1] | | | **Outputs** (mux select) | | **FIFOs** (read enable) | | |
|---|---|---|---|---|---|---|---|
| **N** | **W** | **PE** | **S/PE** | **E** | **N** | **W** | **PE** |
| N→S/PE | None | None | N | 0 | RD | 0 | 0 |
| None | W→E | None | 0 | W | 0 | RD | 0 |
| None | W→S/PE | None | W | W | 0 | RD | 0 |
| None | None | PE→E | 0 | PE | 0 | 0 | RD |
| None | None | PE→S | PE | PE | 0 | 0 | RD |
| N→S/PE | W→E | x | N | PE | RD | RD | 0 |
| N→S/PE | None | PE→E | N | PE | RD | 0 | RD |
| None | W→E/S/PE | PE→E/S | W? | W | 0 | RD | 0 |

[1]Preferred turn directions are computed on arriving packets rather than those at the head of the FIFO as the cascaded FIFOs do not provide peek behavior at FIFO read head.

be stalled when output ports are busy. For our FIFO-based design, we can locally export the FIFO full state to the packet generation logic within the PE to achieve identical behavior with no logic cost. Deep FIFOs can absorb multiple packets and limit deflection penalties, but they can block downstream packets (head-of-line blocking [6]) and affect packet latencies as well. Through appropriate sizing of the NoC buffers, we can aim to deliver balanced performance.

The UltraScale FIFO cascades do not expose internal read-side data ports to user logic directly. They have to pass through the cascade multiplexers and registers (in DO_PIPELINED mode) before they are visible to user logic. However, routing decisions must be made on packets at the head of the FIFO, even if these signals are not exposed to user logic (See Figure 3). Hence, we are required to pre-compute decisions at the time the packet is committed to the FIFO. These decision bits need to be stored in distributed RAMs that mimic the packet delays in the FIFO. The routing function then generates FIFO read controls (RDEN), and multiplexer select (CASDINMUX) signals. The cascade pipeline structure also forces different input to output delays for different packet paths, which must be accounted for when generating the control signals for the FIFOs. This makes the DOR logic more complicated to implement than the original Hoplite implementation. We show the modified routing decision matrix in Table I. Here, the pre-computation logic will indicate the preferred routing direction of the packets when they are entering the FIFO. The post-computation logic will determine exact routing direction (multiplexer control) after taking into account conflict conditions and FIFO full states. The sequential multiplexer ordering assigns the highest priority to packets arriving along the NORTH direction first, WEST direction next, and the PE direction the last. As indicated earlier, if NORTH and WEST FIFOs are approaching FULL state, the router falls back to deflection routing mode until they return to non-full state.

### C. Store-Forward Routing

Deflection routing typically works with single-flit packets. It is possible to support multi-flit packets while maintaining

TABLE II: Queueing capacity of packets from FPGA interfaces for deflection-routed store-forward scenario.

| **Interface** | **Burst Description** | **FIFO36E2** (72b-flit) Length | Fill |
|---|---|---|---|
| PCIe x16 | TLP of 512 bytes | 57 | 11% |
| DDR4 DRAM | 64 byte cache-line | 8 | 1.5% |
| | 72-b interface | 1 | 0.1% |
| 1Gbps Ethernet | 9000 byte jumbo packet | 1000 | 190% |
| | 1522 byte regular packet | 170 | 33% |

compatibility with deflection routing under certain conditions with store-forward routing. For store-forward routing (also used in the old DimeTalk [12] FPGA NoC router), network traffic must consist of fixed pre-determined packet lengths that are injected and switched at the interval of packet length. This allows the packet to be fully stored at each switch hop before traversal. This helps use the spare BRAM capacity in the NoC router, reduce addressing overhead per packet, and enable support for FPGA system-level traffic.

This can be achieved as follows:

- We can statically configure physically separate networks for distributing data with unique packet lengths within FPGA regions as desired.
- Alternatively, the NoC can be dynamically reprogrammed at runtime to support a particular packet length tailored to an active interface.
- Some combination of spatio-temporal partitioning combining above two properties.

In Table II, we enumerate the PCIe, DRAM, and Ethernet packet sizes under a burst-oriented mode and observe lengths between 8–1000 flits long for 72b flits.

## IV. Results

We run cycle-accurate simulations of the RTL using iverilog (git hash 063ae77 github.com/steveicarus/iverilog.git). In our simulations we route fixed-sized workloads 1024 packets/PE to completion, allowing sufficient warm-up time in the NoC, when calculating throughput and latency metrics. We sweep various buffer depths (4–256), system sizes. We use cycle-accurate Verilog models for the FIFO18E2/FIFO36E2 and SRLC32E blocks from Vivado 2016.3 distribution. ($2\times2$–$16\times16$), traffic patterns (`LOCAL`, `RANDOM`, `TORNADO`, `BITREV`, `TRANSPOSE`), and injection rates (1–100% in multiple increments).

### A. FPGA Realization

We compile the RTL to Xilinx UltraScale FPGAs using Vivado 2016.3 edition. A single instance of our 72b switch takes 64 LUTs and 40 FFs along with three FIFO32E2 blocks while operating at 727 MHz. This is $\approx1.4\times$ smaller in LUT count and $2\times$ smaller in FF cost when compared to LUT-based 72b Hoplite. The somewhat high LUT cost of the RAM-based Hoplite is due to the SRL-based FIFOs to delay matching the direction decoded signals. This is a limitation of the cascaded FIFO organization that prevents peeking the FIFO read heads to determine desired turn directions. It may be possible to

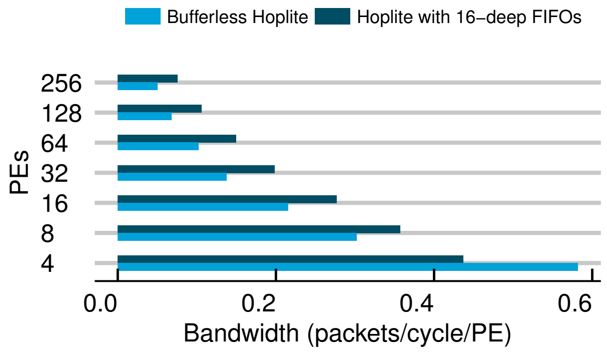


Fig. 4: Sustained bandwidth for 100% injection rate for uniform `RANDOM` traffic across various NoC configurations (PE sizes) with 16-deep FIFOs.

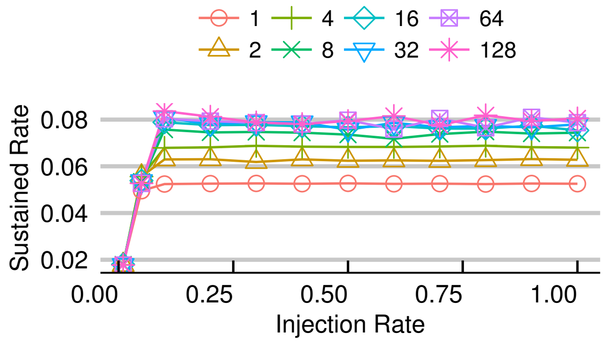


Fig. 5: Impact of buffer depth (colors+shapes) on throughput at various injection rates for 16×16 NoC under uniform `RANDOM` traffic.

reduce the SRL costs if we had peeking ability on the raw FIFO outputs rather than the cascaded outputs.

We generate layout for the VU9P UltraScale+ FPGA chip used in the Amazon AWS F1 Developer Preview platform. This FPGA has 2160 RAMB36 blocks organized as 180×12 grid of RAM blocks. This gives us a maximum NoC configuration size of 60×12 = 720 switches assuming 3 RAMs allocated per switch. Under this arrangement, we use 100% of the BRAM resources for the buffered NoC while leaving the denser UltraRAM for user design. The complete NoC takes up 49.6K LUTs (4%) and 68K FFs (3%) including pipelining costs between routers and dummy PE packet injection logic while operating at 385 MHz. This system-level frequency is roughly 1.2× faster than equivalent LUT-based Hoplite implementations. Smaller NoC system sizes that do not use all Block RAM resources are, of course, possible. When generating layout, we take care to avoid straddling the three BRAM switch across clock region and SLR (super logic region) boundaries. While it is possible to use vertical cascades for routing NoC traffic between the switches, we do not use this feature in our design. This is because the multiplexer ordering forces the NoC links to be generated in a non-cascadable order (East exit first, South/PE exit next). If the vertical UltraScale RAM cascades were redesigned to allow static bypassing of certain RAM stages, this can be simplified.

## B. *Bandwidth Trends*

In Figure 4, we show how the presence of buffering (16-deep FIFOs) improves the sustained throughput by ≈50-60% over pure bufferless solution. In this plot, the sustained throughput is measured per PE, and we observe a reduction in this figure as we increase the number of PEs as expected due to increased network congestion. The total communication throughput (multiplied by the number of PEs) increases as we would expect with parallelism. The quantum of benefit is visible only above 8 processors. This is due to the longer latency of packet propagation through the FIFOs which dominates overall runtime. At larger system sizes >8 PEs, buffering starts to deliver measurable improvements.

## C. *Impact of Buffering*

In Figure 5, we show the impact of buffering depth of performance (sustained throughput) at various injection rates for a 256 processor NoC. When the network is lightly loaded <8–10% injection rate, the bufferless network (depth of 1 is equivalent to bufferless routing with input registers) is able to match the performance of the buffered networks. However, as we increase injection rates, the performance starts to separate and at >16-deep buffers the network starts to saturate. We observe the stabilized sustained rates increase from 0.05 (5%) for bufferless routers to 0.08 (8%), an almost 60% increase in throughput. We achieve diminishing returns above 16-deep FIFOs due to the routing constraints of the arbitration function. There is limited freedom in steering packets under contention as the BRAM multiplexer cascades only flow in one direction. This restriction prevents packets arriving along the NORTH dimension to steer EAST, and also forces the EAST port to get blocked for (1) all packet injections from the PE, and (2) all FIFO reads from WEST FIFO. Despite this limitation, we are still able to deliver 50–60% improved throughput with shallow 16-deep FIFOs. As explained in Section III-C, the deeper FIFOs also permit transmission of fixed-length packets with multiple flits (keeping packets per BRAM to 8–16 as desired) which is useful for suitably aligned DRAM transactions, PCIe bursts, and Ethernet frames.

## D. *Effect of FIFO sizing*

To better understand the effect of FIFO sizing, we plot the FIFO blocking rates which are a ratio of number of FIFO full events as a function of total cycle count in Figure 6. As we can see, the PE port is blocked most often as it has the lowest priority in the multiplexer cascade. Additionally, this is the only port in the NoC that has flow control and allows the FIFO fullness to stall the packet generation logic in the PE. The West and North FIFOs are blocked considerably less often as they have a higher priority. The full status flags on these FIFOs result in the router resorting to regular deflection routing and there is no further increase in the occupancies of those FIFOs. Fortunately, this fallback to deflection mode is

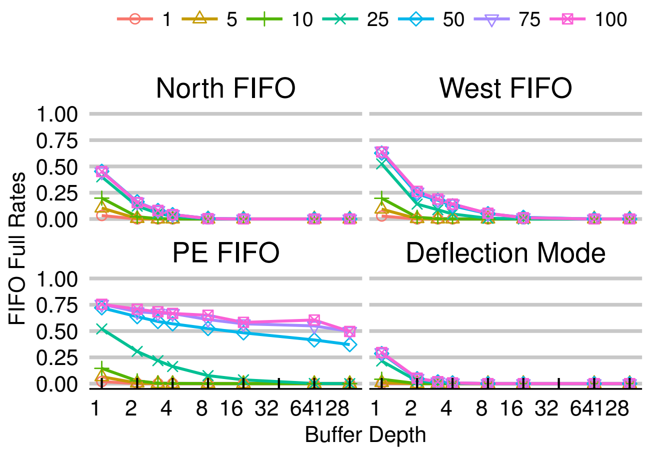


Fig. 6: FIFO Full rates for 16×16 NoC at various buffer depths ($x$-axis) and injection rates (colors + shapes).

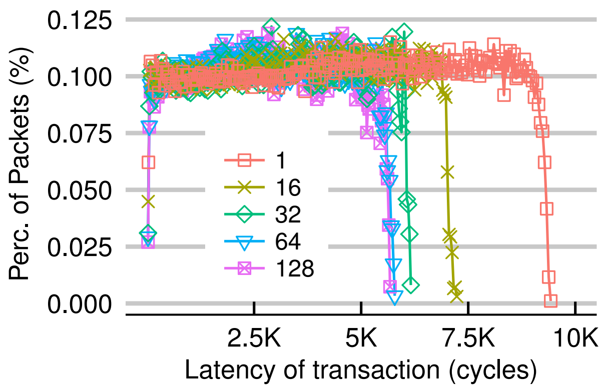


Fig. 7: Improvement in packet latencies due to buffering for 16×16 NoC configuration, uniform RANDOM traffic, and 100% injection rate for various buffer depths.

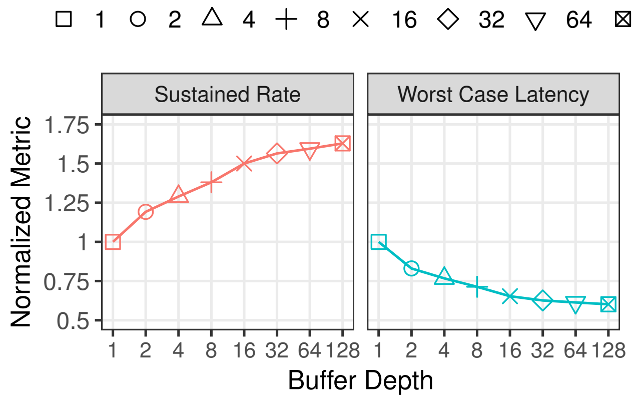


Fig. 8: Evaluating tradeoffs between normalized bandwidth (left) and normalized worst-case latency (right) for 16×16 NoC configuration, uniform RANDOM traffic, and 100% injection rate for various buffer depths.

very rare and happens <25% of the time for very shallow buffer depths of <4, and practically never for larger than >4 buffer depths. Among the switch FIFOs, the North port has the higher priority in the cascade and is blocked slightly less often than the WEST FIFO. For both these NoC directional FIFOs, we observe negligibly rates of FIFOs going full above FIFO depths of 16–32. Thus the FIFOs are able to absorb the network traffic at these sizes. Under these conditions, larger buffers do not contribute to throughputs, but improve energy efficiency by avoiding unnecessary deflections to reduce energy (Section IV-F), and also reduce worst-case latency (Section IV-E). We note, however, that the 512-deep FIFO36E2 and FIFO18E2 blocks can still be used for transmitting larger multi-flit packets (Section III-C).

### E. Worst-Case Latency Trends

In Figure 7, we observe the improvement in latency distribution due to FIFO buffering for a 16×16 NoC routing uniform RANDOM traffic under 100% injection rate. As we observe, packets now spend less time in the NoC, but spend most of it waiting in FIFOs rather than deflecting. In this particular instance, we observe an almost 40% decrease in the worst-case packet latency when using buffered routing with 128-deep buffers. This latency improvement stays steady even at lower injection rates. Above a depth of 64, the reduction in worst-case latency slows down and the benefits saturate.

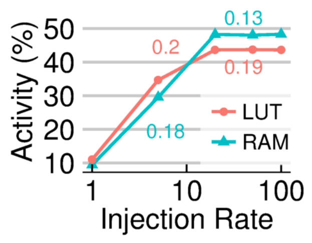
In Figure 8, we show the impact of buffering depth on the bandwidth (left plot) and worst-case latency (right plot) at 256 PEs and 100% activity rate. A buffer depth of 16–32 delivers a good balance between improved throughputs and worst-case latency improvement. Above a depth of 32, the improvements in bandwidth and worst-case latency saturate and deliver diminishing returns.

### F. Power

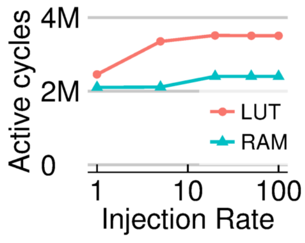
As shown in Table III, the use of BlockRAM resources increases the dynamic power use of the switch block by almost 2.2× over a LUT-based implementation for injection rates <75%. This is to be expected as the buffers consume physical area and power. At high activity rates ≥75, the RAM-based routers are more marginally more efficient. The multiplexers and the buffers together account for 90% of the total power. As shown in Figure 9a, the reduction in activity rate in the RAM-based routers is visible for injection rates below 10%. At higher rates, the faster completion time of the NoC workload results in 5–8% higher activity rate. At low injection rates, the FIFOs easily absorb deflections, but the resulting reduction in NoC activity is low resulting in only 10% energy improvement. At high injection rates, FIFOs avoid deflections to a greater extent to deliver ≈45% reduction in NoC activity but a higher activity rate due to faster throughput. Despite this, the slower NoC workload completion time for the LUT-based NoC results in an energy improvement of ≈40% in favor of the BRAM-based router.

TABLE III: Power Usage for Buffered and Bufferless routers (Vivado Power Analysis. 50% static probability).

| **Kind** | **Total** 12.5% (W) | **Static** (W) | **Dynamic (diff. activity %)** (mW) 5% | 10% | 20% | 50% | 100% |
|---|---|---|---|---|---|---|---|
| Buffered | 2.523 | 2.441 | 69 | 71 | 74 | 82 | 94 |
| Bufferless | 2.478 | 2.440 | 34 | 39 | 49 | 78 | 127 |



(a) Activity Rate and Energy



(b) Active Cycles (1M packet simulation)

Fig. 9: NoC Activity for 16×16 NoC with `RANDOM` traffic. Energy annotation (mW) calc. at 400 MHz freq.

## G. Impact of Traffic Patterns

Finally, in Figure 10, we show the effect of varying injection rates at different synthetic traffic patterns at 256 PEs and a buffer size of 16. As we increase injection rates, the throughputs start to saturate. The availability of buffering boosts throughputs for the `RANDOM`, `LOCAL`, `TORNADO`, and to lesser extend the `TRANSPOSE` patterns. However, buffering does not significantly affect `BITREV` patterns which is known to be adversarial and not reflective of real-world behavior. For the realistic scenarios, we observe throughputs boosted by 50–60% (≈8% vs ≈5% rates also seen earlier in Figure 5 for `RANDOM` traffic).

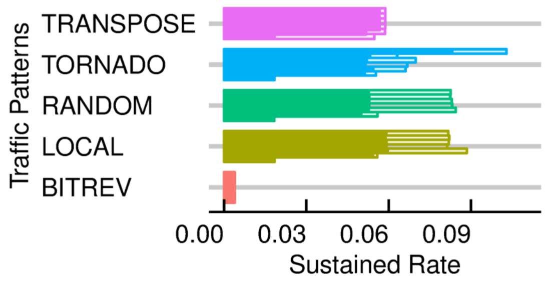


Fig. 10: Comparing the effect of buffering on NoC traffic patterns on 16×16 NoC, with varying injection rates. Translucent:16-deep buffer, Filled: Bufferless.

## V. Discussion

### A. Buffering in Deflection Routing

The idea of introducing buffering in deflection routed networks has been around for decades, and applied in the context of high-performance computing systems and large-scale optical networks [6]. In the context of ASIC networks-on-chip, Mutlu [7] shows how to use input buffering (FIFOs), or lower-cost shared buffering per NoC switch to improve the performance of deflection routed networks. Unlike this design, we exploit the unique cascade structure of FPGA UltraScale RAM blocks to support buffered deflection routing on FPGAs.

### B. FPGA-based NoCs

One of the early studies in FPGA-based overlay NoCs that uses BlockRAMs is the **DimeTalk** [12] router. The design used the memory for storing routing tables instead of packet buffering and it was large ≈432 Virtex-2 slices per 32b router. The **UBC BRS** [11] router also uses FPGA BlockRAMs to effectively share the buffering requirements of multiple input ports with virtual channels. However, due to bandwidth limits of a single BlockRAM, the sharing compromises the throughput of the router by 15–50%. The **CMU Connect** [13] design presents an FPGA-friendly NoC generator that supports various topologies and features such as virtual channels. The **Penn Split-Merge** [8] router presents modular NoC design blocks that can be used to compose larger, arbitrary FPGA NoCs using FPGA-friendly design that eschews the idea of virtual channels and uses localized flow control within the block. Our prior work on leaner switch design with **Hoplite** [10] shows how to reduce resource costs for packet-switched single-flit routers by almost an order of magnitude over CMU Connect and Penn Split-Merge designs. In addition to reducing implementation cost, the use of BRAMs in this paper boosts throughput, lowers latency and improves energy efficiency of the routed workload. **Marathon** [9] overcomes the latency and ordering limitations of Hoplite by using static scheduling of packet injection while still using the lean Hoplite design for the NoC hardware itself. However, such knowledge of packet injection times may not be available. In contrast, the design presented here does not require any a priori knowledge of the communication workload. **DSP48-based Hoplite** [4] lowers LUT costs by using the embedded multiplexers within the DSP48 block to support the internal NoC switch crossbars. While this approach saves LUT+FF costs, it steals valuable DSP48 resources away from the user design. Within the DSP48 block, the design only uses the 48b multiplexers and discards the adder and multiplier resources. To compound matters further, the user must multi-pump the DSP48 block at twice the user design frequency to retain cycle-accurate behavior at the NoC interfaces. In this paper, we explore the use of the UltraScale BlockRAM cascades to support NoC functionality without these workarounds. These RAMs are configured as FIFOs without multi-pumping to support packet queuing functionality and this extra resource cost is converted into a performance improvement. **Multi-level Hoplite** [5] shows how to overcome worst-case packet latencies by creating hierarchical multi-level networks at the expense of additional wiring resources. In this paper, we do not require additional wiring resources but use RAM blocks to enhance router performance.

### C. Resource Comparison with Hard Routers

We evaluate silicon costs of our router on a 65 nm TSMC process based on resource models published in [2] and report

TABLE IV: Comparing BRAM-based router with ASIC hard router (65 nm TSMC + Stratix-III baseline).

| **Datawidth** | **Hard Router** | | **BRAM-based Router** | | **Ratio** |
|---|---|---|---|---|---|
| | $mm^2$ | LABs | $mm^2$ | LABs | |
| 36-bit | 0.23 | 10.9 | 0.26 | 12 | 1.3× |
| 72-bit | 0.3 | 14 | 0.48 | 21.8 | 1.8× |

our measurements in Figure IV. From [2], the cost of a 9kbit RAM is 0.0635 $mm^2$ while that of a 144kbit RAM is 0.5897 $mm^2$. From this data, assuming a linear fit, we estimate an 18kbit RAM to need 0.073 $mm^2$ (1.15× scaling factor) while a 36kbit RAM to need 0.146 $mm^2$ (2.3× scaling factor).

**18kb FIFO + 36b mode**: For our router, we use 3 18kbit Block RAMs which cost 3×0.073 $mm^2$=0.219 $mm^2$ of physical area. This is equivalent to 9.9 Stratix-III LABs (1 LAB = 0.0221 $mm^2$). For our FPGA implementation, both the BRAMs and the crossbar (implemented as a cascaded multiplexer) are hard blocks. We use programmable logic resources to implement the decoder itself which accounts for ≈64 6-LUTs which roughly equivalent to 6 Stratix-III LABs (each LAB = 10 ALMs or LUTs). Thus, we estimate our 36b router occupied 11.9 LABs (0.219 + 6×0.0221 = 0.35 $mm^2$) worth of silicon area on the FPGA die with the buffers accounting for 62% of router area. This is much larger than the ≈50% requirement for input module seen in Figure 10 from [2]. Furthermore, the total area of a 36b hard router with the 10-deep 2-VC design from [2] is 0.23 $mm^2$ which is roughly equivalent to 10.9 LABs. In contrast, our 36b router is 1.3× larger ≈15 LABs but offers 256-deep×36b input FIFOs which are 10× larger than the 10-deep×2-VC FIFOs available in the reference hard router.

**36kb FIFO + 72b mode**: If we use the 36kbit Block RAMs instead, we get twice the capacity (36kb vs 18kb) and also twice the datawidth (72b vs. 36b). This increases the total cost of our BRAM-based router to 3×0.146 $mm^2$=0.438 $mm^2$. This is equivalent to 0.438 $mm^2$/0.0221 $mm^2$ = 19.8 LABs yielding a total router cost of 25.8 LABs (silicon area of 0.438 + 6×0.0221 = 0.57 $mm^2$). For a 72b hard router, we project a total cost of approximately 0.20 $mm^2$ (input/output buffers and crossbar from Figure 10 in [2]) + 2.25×(0.009 $mm^2$ + 272×13$\mu m^2$ + 260×120$\mu m^2$) (wiring and fabric port costs) = 0.3 $mm^2$. Here, we scale wiring costs and fabric port costs by 2.25× when going from 32b to 72b width. Thus, we estimate the total cost of the hard router to be roughly 14 LABs. In this case, the BRAM-based router is ≈1.8×. We attribute this increase to the differential increase in router components when increasing data width of the links. While the input module and crossbar costs will increase linearly with data width, they account for <65% of total router area. Thus the overall router area does not increase as aggressively as our BRAM-based router, where the buffer costs are a large percentage (76%, $\frac{0.438}{0.57}$) of total router area.

Thus, our BRAM-based router improves throughput, latency, and energy use of the deflection-route FPGA soft NoC while closing the area gap with the hard NoC router to within 1.3–1.8× (less than a factor of 2). This is significantly smaller than the 20–23× area gap claimed in [2].

## VI. Conclusions

We show how to exploit Xilinx UltraScale BRAM cascade structures to implement buffered deflection-routed NoC on FPGAs. While buffering is not strictly required for correct implementation of deflection routing, it is useful for boosting performance. For 16×16 NoCs, we deliver throughput improvements of 50–60%, worst-case latency reduction of ≈40%, and energy reduction of 10–40% for uniform `RANDOM` workloads while using only 16-deep buffers. Our router occupies 64 LUTs, 40 FFs, and 3 RAM blocks per instance while operating at 727 MHz per instance. This is 1.5–2× smaller than the equivalent LUT-based implementation of Hoplite. On the VCU9P UltraScale+ FPGA board, we can fit the largest 60×12 NoC configuration that operates at 400 MHz while requiring less than 4% of LUT/FF resources on the chip. Our design also significantly closes the physical area gap over hard FPGA NoCs from the previously claimed 20–23× to under a factor of two.

*Acknowledgements*: The authors would like to thank Jan Gray for providing access to Hoplite RTL source code.

## References

[1] 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.

[2] M. S. Abdelfattah and V. Betz. Networks-on-Chip for FPGAs: Hard, Soft or Mixed? *ACM Trans. Reconfigurable Technol. Syst.*, 7(3):20:1–20:22, Sept. 2014.

[3] S. Chandrakar, D. Gaitonde, and T. Bauer. Enhancements in UltraScale CLB architecture. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pages 108–116, New York, NY, USA, 2015. ACM.

[4] K. H. B. Chethan and N. Kapre. Hoplite-DSP: Harnessing the Xilinx DSP48 multiplexers to efficiently support nocs on fpgas. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–10, Aug 2016.

[5] K. H. B. Chethan, A. Shubham, and N. Kapre. Deflection routing for multi-level FPGA Overlay NoCs. In *2016 International Conference on Field Programmable Technology*, pages 1–8, Dec 2016.

[6] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

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

[8] 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.

[9] N. Kapre. Marathon: Statically-scheduled conflict-free routing on FPGA Overlay NoCs. *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 00:156–163, 2016.

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

[11] J. Kwa and T. M. Aamodt. Small virtual channel routers on FPGAs through block ram sharing. In *2012 International Conference on Field-Programmable Technology*, pages 71–79, Dec 2012.

[12] Nallatech Inc. Dimetalk 3.1 reference guide. Technical report, Nallatech Inc.

[13] 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.

[14] Xilinx Inc. Ultrascale architecture memory resources user guide ug573. Technical report, Xilinx Inc., 2016.