Hoplite-DSP: Harnessing the Xilinx DSP48 multiplexers to efficiently support NoCs on FPGAs

Chethan Kumar H B
School of Computer Science and Engineering
Nanyang Technological University
Singapore, 639798
chethank001@e.ntu.edu.sg

Nachiket Kapre
School of Computer Science and Engineering
Nanyang Technological University
Singapore, 639798
nachiket@ieee.org

***Abstract*— We can embed the crossbar functionality of NoC (network-on-chip) routers onto the hard multiplexers of Xilinx DSP48E primitives to support resource efficient mapping of FPGA overlay NoCs. This embedding also permits the use of dedicated hard wiring resources of the DSP cascade links to support vertical NoC channels. This unique mapping allows us to significantly reduce soft logic (LUTs+FFs) utilization of FPGA overlay NoCs at the expense of DSP resources while also lowering the routing requirements on configurable FPGA interconnect. This embedding is made possible by the dynamic mode control feature of the DSP blocks that allows per-cycle modification of ALU operation and multiplexer data steering controls within the block. We multi-pump the DSP block by internally operating at 600–650 MHz speeds while delivering fabric-facing frequencies of 300–325 MHz. For 48b-wide chip-spanning 32×16 NoC mapped onto an XV7V485T (VC707 board), a LUT-only implementation of the Hoplite router requires ≈70 LUT+140 FFs@2.7 ns instead of 1 DSP48 block+≈13 LUTs+17 FFs@2.8 ns on average. For 15% toggle rates, across most system sizes, the DSP-based NoC exploiting hard resources requires 1.1–2× lower power than the LUT-based NoC. Across a range of statistical workloads, we are able to match the performance of LUT-only Hoplite delivering a sustained rate as high as 8–10% for injection rate of 100% for `LOCAL` traffic pattern when mapped to a 16×16 NoC. In previous work, a conventional hard NoC router with virtual channels, and FIFO buffers has been demonstrated to be 20–23× smaller, 5–6× faster, and up to 14× lower power than equivalent soft NoC routers. Our DSP-based Hoplite soft NoC router requires practically identical silicon area, runs only 3× slower, and consumes 43% less power than the conventional hard NoC router, while sacrificing certain communication properties in favor of a lean implementation.**

I. Introduction

Modern Xilinx FPGAs provide hardened resources such as thousands of DSP48E units along with specialized interconnect support between these units. For a range of signal processing and streaming applications, these DSP blocks provide unparalleled compute density made possible by the high-speed 650 MHz+ fixed-point multiply-adder support. However, in many scenarios such as cryptography, bit-twiddling computations, small-precision operations, these DSP blocks are of little use. Furthermore, not all DSP blocks may be utilized for a given application leaving many of them idle and underutilized. Even when applications require a large number of DSP blocks, they may not use them in each cycle thereby opening the possibility of time-sharing the DSP block between the computation and communication phases. In these scenarios, it may be useful to consider alternative creative uses of the highly flexible design of the DSP block. A key feature of these blocks for the Xilinx family of FPGAs is the ability to dynamically change the function and dataflow inside the blocks at runtime from user logic. The implications of this freedom for supporting network-on-chip multiplexing computations is an unintended, but remarkable side-effect. In this paper, we consider the potential for exploiting Xilinx DSP48E blocks for supporting deflection routed torus NoC operations for NoCs such as Hoplite [7].

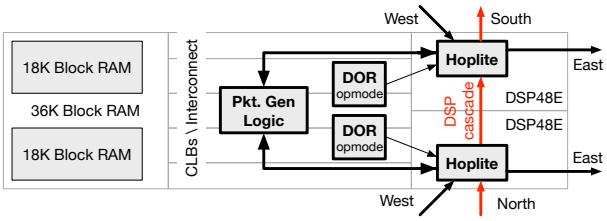


Fig. 1: Embedding Hoplite NoC functionality into the Xilinx DSP48E blocks. (grey wires use *hard* cascade links between DSP blocks, diagram adapted from Figure 1-2 in [11])

This may seem like an odd use of a DSP block as NoC operations are not dominated by any arithmetic operations supported by the DSP. However, the data steering capabilities and wide 48b multiplexer support are particularly attractive from the perspective of mapping the internal NoC crossbars. In [7], we introduce Hoplite, an extremely lightweight FPGA-friendly NoC router that implements deflection routing on a unidirectional 2D torus. The Hoplite router, shown in Figure 2, occupies 70 LUTs, 140 FFs on a modern Xilinx chip for 48b flits while operating at over 300 MHz through carefully folded layout of the 2D torus topology on Virtex-6 and Virtex-7 devices. Analysis of the resource utilization suggests that bulk of the LUT and FF usage (80–90%) arises from the need to implement the internal switching crossbar. Juxtaposed with the configurable dataflow potential and cascade link support of the Xilinx DSP48 units, we can see a pathway for cheap and fast implementation of the crossbar functions inside the DSP

block. What makes this mapping particularly appropriate for deflection routed unidirectional torus NoCs is the requirement to support 3-input multiplexers and the strict directionality of the DSP48 cascade connections. We show a high-level vision of how to configure and use the DSP48 blocks for Hoplite links in Figure 1. Here the packet generation logic is implemented in FPGA logic along with the DOR (dimension ordered routing) functionality that generates the dynamic `OPMODE` signals for data-steering within the DSP block. We can also exploit the Block RAMs for storing packets and use their fast connections to the DSP48 inputs for high-speed operation. Interestingly, the dedicated cascade connections allow the NoC to avoid consuming expensive configurable wiring resources from the spatial fabric while operating directly on high-speed dedicated wiring. In this context, we can approach the performance and cost of a hard NoC routing [3] on FPGAs that are available today using the lean Hoplite router design. One downside of the multi-pumped design is the potential increase in power utilization as we must route a 2× faster clock to the DSP resources. However, there is an opportunity in the use of hardened resources being able to offset the increased power requirement arising from the faster clock.

The key contributions of our paper include:

- Multi-cycle design of Hoplite NoC functionality on Xilinx DSP48E blocks. Support for full crossbar design (48b when using cascades, 27b without cascade).
- Additional considerations for time-multiplexing, and FPGA layout for enhanced operation.
- Resource, performance, and energy analysis of DSP-based Hoplite design over the original LUT-based implementation across a range of statistical and real-world workloads.

II. Background

A. Hoplite FPGA Router

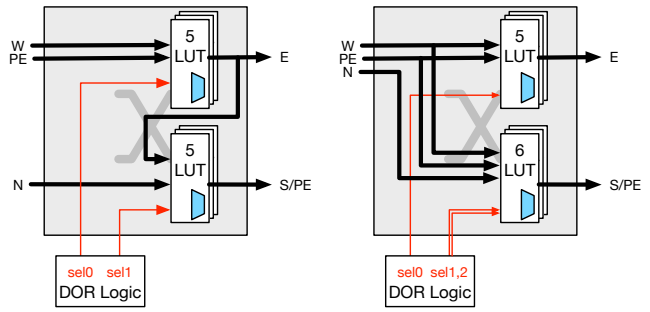


Fig. 2: High-Level diagram of Hoplite router microarchitecture. Mux mapped to fractured Xilinx 5-LUTs. Design on left implements *partial crossbar* while the one of the right supports *full crossbar*. The tradeoff is a factor of 2 reduction in resource usage when supporting partial connectivity.

Hoplite is an FPGA-friendly NoC router design that is designed to be compact, fast and scalable on large, modern FPGA devices. We show a high-level diagram of the Hoplite [7] NoC router in Figure 2. Hoplite uses a simplified 2D unidirectional torus topology and employs bufferless deflection routing for lean and fast operation on FPGA fabrics. By eschewing the need for FIFO buffers, the FPGA resource utilization is kept low and the associated control logic for managing deflection is also straightforward to implement. The choice of unidirectional torus implies that the internal switching crossbar must support three inputs and three outputs (one for `NORTH-SOUTH` link, another for `WEST-EAST` link and the final one for the `PE` connection). When considering modern fracturable Xilinx 6-LUT architectures, this offers the possibility of mapping the 3:1 MUXes into 2×5-LUTs if we share the `SOUTH` and `PE` exits as discussed in [7]. In Figure 2, we show two designs for Hoplite. For the *partial crossbar* design, we cascade the multiplexers to allow the switching crossbar to compactly fit in two 5-LUTs that can be charged to a single 6-LUT. Alternatively, we can implement the *full crossbar* directly in a 6-LUT and pay twice the cost of the *partial crossbar* design if the additional internal bandwidth is useful. This forced economy through LUT fracturing has a measurable impact of throughputs and worst case latency. With careful 2D folded layouts, Hoplite can run at high speeds 300-333 MHz even when spanning complete FPGA chips on the ML605 board (Xilinx Virtex-6 LX240T FPGA).

B. Xilinx DSP48E1/E2 primitives

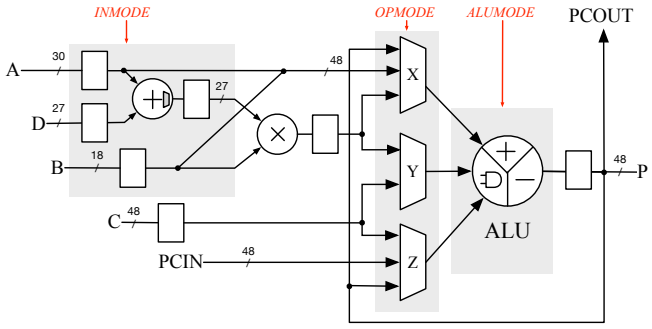


Fig. 3: Simplified diagram of Xilinx DSP48E2 block with configurable ALU function and programmable dataflow. (48b mux `X`, `Y`, and `Z` are used for our design, 27b bypass mux in pre-adder selects between `A` and `D`).

Modern Xilinx Ultrascale FPGAs provide configurable DSP resources, shown in Figure 3, for signal processing applications requiring 27×18 multiplications (25×18 for older devices supporting DSP48E1) or 48×48 add/subtract operations. These offer significantly superior logic density, speed and energy than equivalent LUT implementations of the same computations. Xilinx DSP blocks allow per-cycle reconfiguration of the arithmetic operation and dataflow inside the DSP blocks. While this is tricky to exploit, this opens the door to supporting complex dataflow expressions on the same DSP block through time-multiplexing. This is achieved by simply changing multiplexer controls to steer appropriate inputs and ALU mode controls to change arithmetic/logic function on a per cycle basis in a repetitive fashion.

DSPs as 48b Multiplexers: The X, Y and Z multiplexers and the embedded multiplexer in the A/D pre-adder are of particular importance when steering data within the DSP block. These multiplexers are controlled on a per-cycle basis from user logic through the signals INMODE and OPMODE. For mapping NoC crossbar functionality to the DSP block, these multiplexers and the control signals play a vital role. While it is also possible to dynamically control the 48b ALU with the ALUMODE signal, we simply configure it to add X+Y+Z and leave it at that setting. To then use the DSP block as a wide 3-input multiplexer, we select zero input for two of the X, Y or Z multiplexers thereby allowing us to select between X+0+0, 0+Y+0, or 0+0+Z choices. In this case, it is sufficient to control OPMODE signals alone to achieve desired routing functionality.

DSPs as 27b Multiplexers: An astute reader may have noticed that the 48b multiplexer support is only possible if we use the cascaded PCOUT and PCIN connections. One drawback when using the cascade connections is the requirement to use up a series of DSP blocks that are directly next to each other in the column. In some instances, this may not be possible. In such scenarios, we can still configure the DSP block as a 27b multiplexer by using the A, D and C inputs. The 27b width of the D input constrains overall width. In this scenario the A:D pre-adder and Y mux control are adequate for support programmable dataflow through the DSP block. Here, the control logic needs to drive both INMODE and OPMODE signals for correct operation.

Multi-Pumping: It is also possible to multi-pump the DSP blocks by internally operating at the maximum possible frequency of the DSP blocks 650 MHz+ while rest of the logic runs 2× slower. This comes in handy as we need to dynamically change the mode controls when implementing the NoC crossbars. Multi-pumped DSP designs have been demonstrated in the context of high-level synthesis before in [5]. Multi-pumping has also been explored in the context of other hard resources such as BRAMs [9]. The dynamic control feature has also been used for processor-oriented designs in [6] to resource share the DSP block across multiple instructions.

III. HOPLITE DSP

In this section, we describe the main idea behind mapping multiplexers onto DSP48E blocks on the Xilinx FPGA while also discussing layout considerations when targeting columnar DSP layouts.

A. DSP48Es and 48b Multiplexers

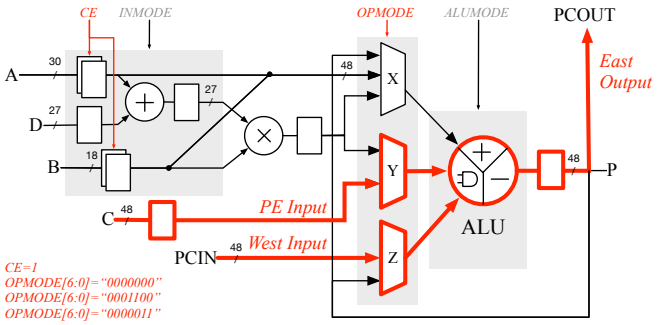
The key to embedding Hoplite functionality into the DSP block is (1) the assignment of NoC router inputs to the correct DSP block inputs, (2) splitting a single Hoplite-LUT design cycle into multiple Hoplite-DSP cycles, and (3) the ability to modify OPMODE signals based on the DOR (dimension-ordered routing) function. The DOR function forces packets to route in the X-dimension first before turning in the Y-dimension. This means that certain turns are disallowed, *e.g.* turning from Y-dimension to X-dimension is not permitted. We first focus on exploiting the 48b-wide multiplexers to the fullest extent, while showing how to achieve reduced bandwidth, but less constrained 27b operation later in Section III-E. In Figure 4, we can see how we map the internal crossbar of the NoC router to the X, Y, and Z multiplexers across 2 cycles. By carefully presenting the right subset of inputs to the multiplexer in the right cycle we can achieve identical behavior as Hoplite. We do have to split the computation across multiple cycles as the DSP block can only produce a single output, but multi-pumping helps us compensate for multi-cycle operation. The original Hoplite design presented in [7] used a cascaded crossbar implementation to permit perfect mapping to the fracturable Xilinx 6-LUT architecture. When using the DSP block, we no longer need to be limited by the partial crossbar functionality, and can directly support the full crossbar operation as shown in Figure 4. In this scenario, the EAST output is resolved in the first cycle by selecting between the PE and WEST inputs at multiplexer Z (For DOR, NORTH cannot turn EAST). As is clear, only one of these two signals can send a packet in a given cycle (in any direction). In the next cycle, the overloaded SOUTH/PE output is resolved by selecting between all three inputs with priority for the NORTH input. These two cycles are sufficient for correct operation of the Hoplite router. The logic used to implement DOR computation is a trivial comparison of the address fields of the packet with the position address of the switch being traversed. This logic drives the OPMODE controls sourced from the FPGA LUTs, but the PATTERNDETECT feature of the previous DSP blocks can also be used here to help reduce LUTs even further[1].

In this design, we use the X, Y and Z multiplexers in different cycles while programming the DSP unit to simply execute an addition operation on the 48b inputs. By correctly driving the OPMODE signals from user logic based on DOR functionality, we can correctly route all inputs to corresponding output ports. We show the precise OPMODE bit patterns that must be selected for each multiplexer mapping in Table I. ALUMODE is set constant to compute X+Y+Z in all cases. From the perspective of the processing logic, a time-multiplexed 325 MHz design (650 MHz internal DSP clock) is better than LUT-based Hoplite [7] as we are able to support full crossbar connectivity without extra cost. Finally, when we use the cascade connections, we are able to offload almost 25% of the NoC wiring requirements onto DSP cascade wiring thereby freeing up interconnect capacity for actual user logic.

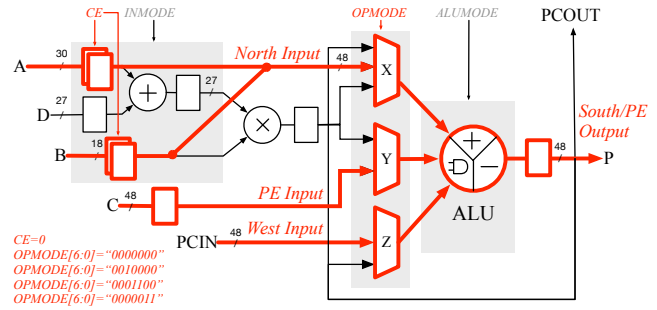
B. FPGA Layout Considerations

Now that we saw how a single Hoplite router can be embedded inside a single DSP48 primitive, we turn our attention to overlaying the complete NoC onto a real FPGA device. Xilinx FPGAs organize silicon resources into separate columns devoted to (1) CLBs and interconnect, (2) Block RAMs, and (3) DSP blocks. The dedicated cascade routes span the column

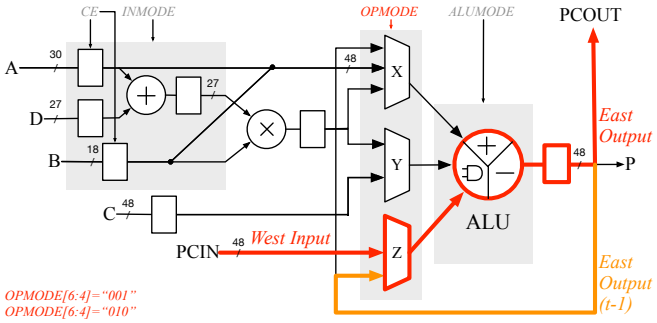
[1]Not implemented in our current design, but trivially possible



(a) Hoplite DSP – Cycle 1



(b) Hoplite DSP – Cycle 2



(c) Passthrough-DSP

Fig. 4: Packing Hoplite NoC multiplexing operations onto Xilinx DSP48E2 blocks for DOR routing algorithm. Passthrough DSPs are used to help align the duty cycle of the flit for next Hoplite and provide a fast routing path for vertical traffic. The NoC is rotated 90° such that the `EAST` output is going up the DSP cascade, and `SOUTH` output is traversing to the right horizontally along the chip (except wraparounds).

alone which depends on the height restrictions of the particular FPGA device. For implementing a 2D torus, we need wiring in both vertical (column) and horizontal (row) dimensions. This means that we must split our routing requirements across both resources. In Figure 5, we show the local connectivity pattern for a single DSP48 block when considering its immediate DSP neighborhood. The `P` output of the DSP block is used by packets exiting the NoC (to the processor/client) as well as packets traversing in horizontal dimension. Traffic in vertical dimension uses the dedicated cascade routes. This design

TABLE I: DSP48E2 Per-Cycle Configuration. (Listed in DOR priority order)

Cyc.	NoC Operation	OPMODE bits		
		X	Y	Z
1	`EAST = MUX (PE,WEST)`			
	`EAST (PCOUT) = WEST (PCIN)`	000	00	11
	`EAST (PCOUT) = PE (C)`	000	11	00
2	`SOUTH = MUX (NORTH,PE,WEST)`			
	`SOUTH (P) = NORTH`	001	00	00
	`SOUTH (P) = WEST (PCIN)`	000	00	11
	`SOUTH (P) = PE (C)`	000	11	00

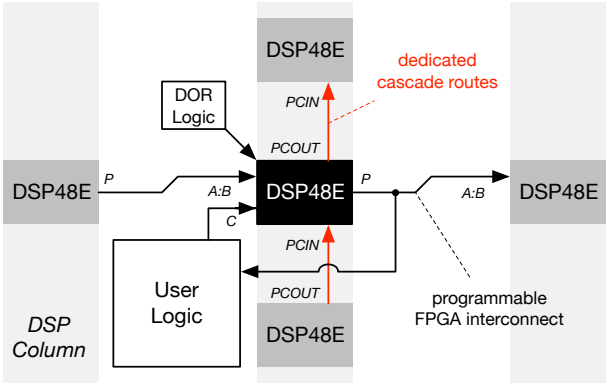


Fig. 5: Connectivity pattern for a DSP48 block and its immediate neighbors.

must use consecutive DSP blocks in a column and revert to programmable FPGA fabric for handing the torus loopback edges. For high-speed operation, all ports to/from the DSP are registered (including `OPMODE` bits).

C. Managing the DSP Cascade

Modern FPGAs contain thousands of DSP48 blocks organized into separate columns each with their own physically distinct cascade track. Since DSP cascade links only span in unidirectional fashion, we must implement loopback edges for ends of the torus in the configurable fabric. We show a high-level diagram of this implementation scheme in Figure 6. For the vertical dimension, the `PCOUT` and `PCIN` connections are not exposed to the logic fabric and must be accessed from within the DSP block itself. This means that, we must sacrifice two rows of DSPs (one at the top and one at the bottom) for enabling this vertical connection by configuring them to simply steer the cascades to/from the fabric. One row of DSPs at the top allow the `PCIN` connection to be routed to the `P` output. One row of DSPs at the bottom must be configured to connect `C` input (that is manually wired to the top row `P` output) to the `PCOUT` link. This is a small price to pay to exploit the high-bandwidth, dedicated cascade wiring available between DSPs in a column. Now that the torus wrap-around link is a long-distance wire, to minimize its impact on clock frequency, we must carefully add a suitable number of pipeline registers. Previously, in Hoplite [7], it was possible to perfectly

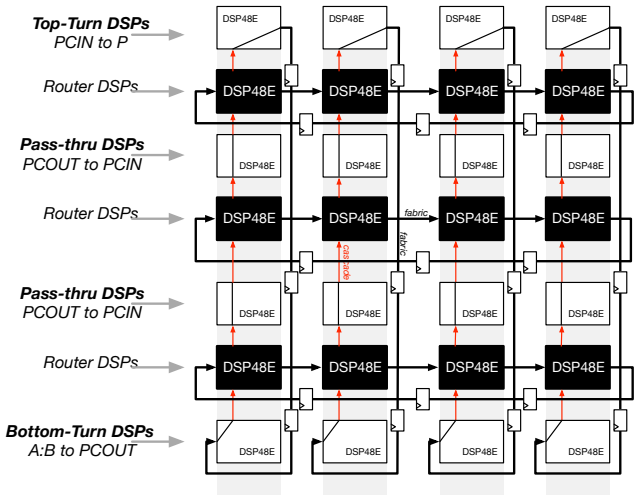


Fig. 6: Supporting torus wrap-around wiring for 2D unidirectional torus with fabric routing. Here, we need to configure edge DSPs for fixed functions. For flexibility in NoC sizing, intermediate DSPs can serve as simple route-throughs in vertical channels. For large multi-die FPGAs, the DSP cascade only extends the height of a *Super Logic Region* and must drop back to FPGA routing for crossing interposer links.

control router placement to permit a high-performance folded layout where all wires as a fixed length. Unfortunately, the presence of fixed cascades means this is no longer possible (unless using cascade-free 27b design shown in Section III-E). A minor side-effect of the lack of visibility of the cascade wires from the FPGA logic is the nominal need to modify DOR logic by pre-calculating route decisions upfront. This requires reading the `P` output in the correct cycle for data that is sent over `PCOUT` exit.

The design we have seen so far, suggests a need to use all DSPs in a column if we want to exploit the cascade links for routing vertical channels. This may be (1) wasteful of DSP resources, and (2) force us to size NoCs to be much larger than we need for our application. In this case, we can program the DSP blocks as simple *pass-through* connections with appropriate amount of pipelining. With time-multiplexing it is also possible to return almost 50% of the DSP bandwidth to the user logic in these pass-through DSPs. This flexibility allows us to embed various NoC sizes on top the DSPs while reducing wastage of DSP bandwidth with a resource-bound upper limit. Additionally, depending on congestion, unused DSP blocks can be used by user logic to run arithmetic operations in idle cycles. Again, the key here is the ability to reconfigure the DSP48 functionality on a per-cycle basis.

D. Multi-Pumping

We show the operational timing diagram of a DSP48 unit I/Os and the exact alignment of signals required to achieve correct operation in Figure 7. The multi-pumping of the DSP block requires that we carefully ensure signals along the NoC remain active for the full 2-cycle stretch at the faster clock

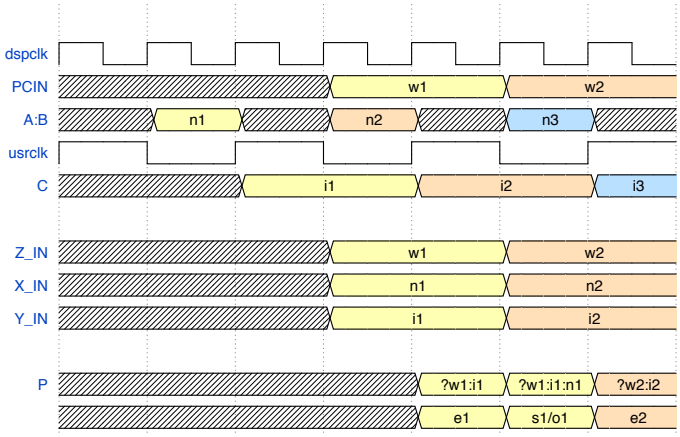


Fig. 7: Timing Diagram showing correct operation for the multi-pumped DSP48 block. The `Z_IN`, `X_IN` and `Y_IN` are inputs at the internal multiplexers in the DSP48 block.

to ensure correct sampling in the DOR decoder. We use the inbuilt programmable clock enable feature to spread the fast signal across two clocks along the `A:B` input. The signal traveling out on `PCOUT` port is suitably stretched by the intermediate DSP block in the vertical channel with the use of internal feedback. In Figure 7, we observe that the `A:B` input is valid on alternate clocks of the faster `dspclk`. This is internally stretched to occupy two `dsp` clock cycles at the `X` multiplexer. Inputs from user logic are driven by `usrclk` and are automatically valid for two `dsp` cycles. The final output `P` alternately produces data for the `SOUTH/OUTPUT` port and `EAST` port at the fast clock. The `EAST` output is dispatched along the `PCOUT` vertical channel and spread to occupy two cycles by using the neighboring DSP48 to alternately sample the `PCIN` and `P` output at the `Z` multiplexer. This allows the `PCIN` input to be valid for two clocks. The number of hops in the vertical dimension must be adjusted to be an odd number of stages to ensure correct cycle alignment with rest of the signals at the next Hoplite block.

E. 27b Cascade-free DSP Overlays

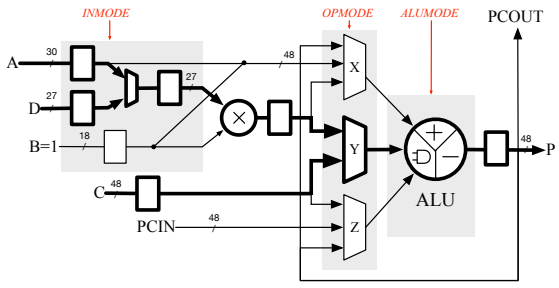
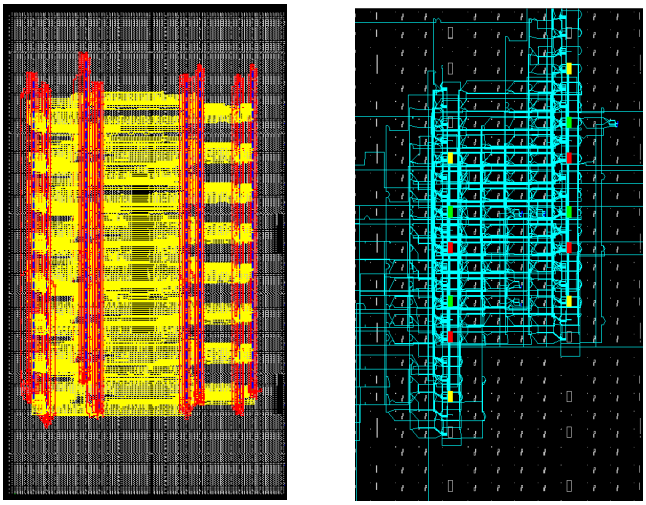


Fig. 8: 27b Hoplite design on the DSP48E block. Avoiding `PCOUT` and `PCIN` connections.

Alternatively, it is also possible to generate a smaller 27b design that does not use the 48b cascade connections at all. While this is clearly a lower-bandwidth solution, it offers more layout freedom by disentangling connectivity restrictions



(a) 8×8 NoC (Full chip) (b) 2×2 NoC (Zoom-in)

Fig. 9: FPGA Layouts for the ML605 board with wiring and DSPs highlighted. (produced via FPGA Editor in Xilinx ISE)

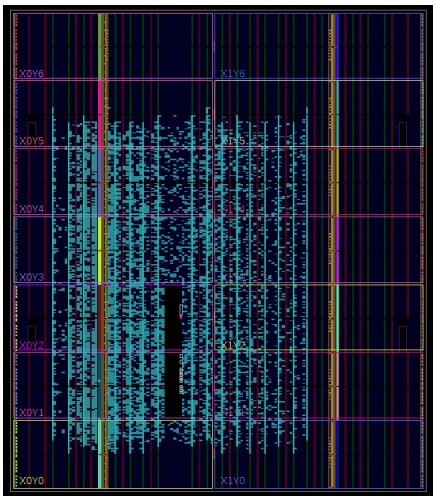


Fig. 10: 16×16 NoC FPGA Layouts for the VC707 board with DSPs in vertical channels highlighted. (produced from Vivado device view)

within DSP columns and permitting free placement anywhere on the FPGA fabric. Additionally, it also cleanly separates unused DSPs from the topology. However, it is clear that traffic that was previously relegated to dedicated cascade connections must now compete for space in the ordinary programmable FPGA fabric interconnect. From Figure 8, we see that the inputs are now mapped to `A`, `D` and `C` signals, while output is exclusively sampled from `P` port. In this case, we are using the internal multiplexer available in the pre-adder block as selected by `INMODE` signals as well as a selection at the `Y` multiplexer. We still need two cycles to resolve both outputs (one cycle per output), but can directly support the full crossbar design as all three inputs are available simultaneously and selectable by the mux cascade.

IV. Methodology

We synthesize and compile various FPGA layouts using Vivado 2015.4 for the VC707 board (XC7VX485T chip) and using ISE 14.7 for the ML605 board (XC6V240T chip). The compatibility of our approach across both Virtex-7 and Virtex-6 families concretely demonstrates the backwards compatibility of the DSP48-based NoC design. We can even go further behind in the Xilinx family tree to other FPGAs that support these reconfigurable hard DSP blocks. Thus, our NoC is a value-add to multiple FPGA generations and not just exclusive to the latest, expensive FPGAs.

We report resource utilization in LUTs, FFs and record Clock Period (ns) and Power (W) based on Vivado's analysis passes. With suitable PLL configuration and pin constraints to route signals to FMC connections (FPGA Mezzanine Card ANSI standard), we are able to correctly synthesize the design and record total board power with Energenie power meter. We simply plug the board's AC power connector into the Energenie socket and measure board-level power at that point. In this configuration, the board is maintained stand-alone with a host computer only connected over USB for FPGA programming. The packet injection logic is designed to be lightweight and small but setup to inject packets into the network at a specific rate. We consider different system sizes and generate XDC constraints that generate chip-spanning NoCs that use large portions of the DSP columns. While it is theoretically possible to use up all 100% of the DSP resource to implement a 140×20 NoC on the VC707 or 96×8 NoCs on the ML605, the logic/DSP balance leaves a paltry ≈100 LUTs per router for user logic. Furthermore, at 48b datapaths and 100% DSP utilization, the horizontal routing resources are completely exhausted and the router aborts the mapping phase. Instead, we consider more realistic scenarios with a need to interconnect user designs larger than ≈500 LUTs/PE (512 PEs or less per FPGA). For the multi-die Virtex-7 series devices, the DSP columns do not span across the SLRs [10] (super logic regions). In these cases, our tool inserts DSP→fabric connections to route the signals over the interposer connections between the dice.

When tuning for performance, we evaluate the effect of additional pipelining stages between horizontal links (non-DSP-cascade wires). For instance, we added between 1–4 stages of pipeline registers to improve the clock frequency, but generally failed to see noticeable improvements beyond a single stage for large system sizes. This is because, the inter-router spacing is short enough to support single-stage connections for large sizes. As the horizontal distance between the DSP columns increases, for smaller sizes, more pipelining stages are required. For vertical links, the DSPs themselves provide required pipelining and are never in the critical path. For chip-spanning layouts, with larger spacing between routers (larger PEs), it becomes necessary to introduce *gaps* between DSPs. The intermediate DSPs operate as dumb *pass-through*

links and can be optionally pipelined. This expansion of an N×N NoC to span across the FPGA fabric with intermediate gaps increases DSP usage, and power with a slight degradation of the NoC frequency (longer horizontal wires). However, this degradation is typically only 10–15% in frequency and 20–30% in power usage for the ranges considered.

We show representative layouts on the ML605 board for 8×8 NoC in Figure 9 along with a zoomed-in layout for a 2×2 NoC. Here the yellow colored links in the left panel show the higher density of horizontal connections using the FPGA routing fabric with fewer red colored vertical channels as most of that traffic is absorbed by the DSP cascades. A closer look at a toy 2×2 NoC in the right panel shows colored DSPs performing different roles – red DSPs serving as actual Hoplite routers, green DSPs simply being pass-throughs and yellow DSPs at the top and bottom rows providing wraparound torus connections. In Figure 10, we show an equivalent layout for a 16×16 NoC. The vertical lines are the DSP cascade blocks and the DOR routing logic and pipeline registers are the shaded irregularly-distributed blocks. The vertical NoC wiring is routed over the cascade links. The inter-DSP column spacing is different on the Virtex-7s than the older Virtex-6 parts making them more amenable to a uniform layout.

V. Results

We synthesize NoCs of varying sizes that fit the ML605 and VC707 boards and perform real power measurements when operating the boards stand-alone (without PCIe). We compute achievable bandwidths across various traffic patterns suitable for multi-processor-oriented applications by routing workloads with 16K packets per PE and recording completion time. While various injection patterns were investigated in Hoplite [7], we focus on the `LOCAL` pattern in this paper as it best captures real-world traffic locality. Here, we generate traffic that produces packets heading to destinations within a short radius of the injection position. For our experiments with set this radius to 3. We vary injection rates (packets injected per PE per cycle) between 1% and 100%. Based on this setup, we first present the FPGA mapping results, power data and then show bandwidths of the resulting NoCs. All comparisons are based on the full-crossbar implementation of the Hoplite router for reasons identified earlier in Section II-A.

A. FPGA Mapping Results

In Table II, we show the results of implementing Hoplite NoCs of various sizes on the VC707 FPGA board with specific XDC constraints for rectangular high-performance layout. As you can see, almost all designs run at very fast speeds 2–3 ns due to careful but simple placement constraints provided to each router block. Without the XDC constraints, we consistently lost speed across all sizes resulting in longer 4–5 ns clock periods. Furthermore, the portion of logic that must switch at the faster multi-pumped frequency is the DOR logic which is merely a few LUTs worth of computation and easily meets the faster timing target. When using LUT-based design, we can need as much as 10% of the soft resources of the FPGA chip and consume as much as 6.8 W of active power for systems as large as 32×16 configuration. In contrast, equivalent DSP-based mappings only consume at most 2% of the soft resources (almost 5–6× less) at the expense of as much as 56% of the hard DSP resources. Furthermore, this DSP implementation requires roughly 5% more power (7.2 W) due to the multi-pumped clock. We note that this increase in power is only observed for the 32×16 NoCs while rest of the system sizes are 1.1–2× power efficient when using DSPs. We believe this is due to extra loading on the clock distribution network to support the 2× faster multi-pumped clock. At large system size 32×16, we have a clear tradeoff between a modest increase in power due to multi-pumping and large resource reduction in LUTs by absorbing multiplexers into DSP blocks. For all other system sizes we considered, our DSP-based NoC delivers both superior energy-efficiency and LUT count reduction.

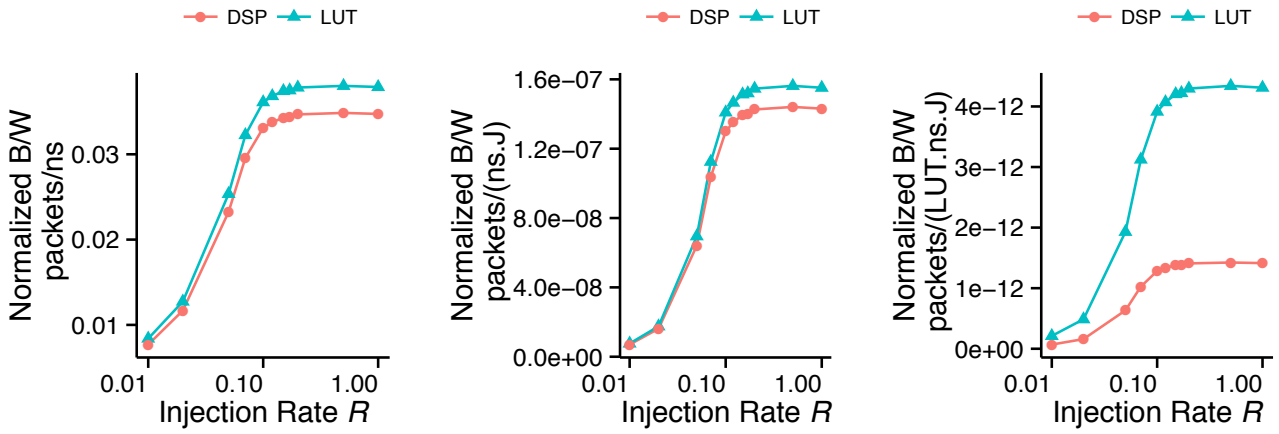
TABLE II: FPGA Implementation Results for Chip-Spanning Layouts on VC707 board (XC7VX485T).

X	Y	LUTs	(%)	FFs	(%)	DSPs	(%)	Clk ns	Pow. W
Without DSPs – Hoplite-LUT Implementation									
32	16	35K	10	62K	12	0	0	2.7	6.8
16	16	14K	4.8	36K	6	0	0	2.2	4.5
8	8	3.5K	1.1	9K	1.5	0	0	2.4	2.2
4	4	900	0.3	2.3K	0.3	0	0	2.8	1.2
With DSPs – Hoplite-DSP Implementation									
32	16	7.1K	2.0	9.1K	1.5	1.5K	56	2.8	7.2
16	16	4.3K	1.4	5.2K	0.8	800	28	2.4	4.1
8	8	1.4K	0.5	1.7K	0.3	208	7	2.1	1.1
4	4	500	0.2	600	0.1	56	2	2.1	0.5

We also show a visual representation of the various implementation costs and evaluation metrics for the 32×16 NoC in Figure 12 assuming an activity rate of 15%. This is the activity rate of the LUT/FF and DSP resources applied across all routers and is a high over-estimate compared to the actual activity even under a 100% injection rate for `LOCAL` traffic. For applications that do not use 100% of the DSP48 resources of the FPGA, our DSP-based NoC mapping approach provides an attractive alternative to repurpose unused DSPs for the NoC and still meeting overall design goals. Even when DSP resources are required for logic, they can be switched into communication mode when data must be routed to different parts of the chip, thereby sharing them for both computation and communication.

B. Bandwidth Measurements and Normalization

In Figure 11, we compare the effect of varying injection rate on the performance of a 16×16 NoC implemented using LUTs and DSPs when routing the statistically generated `LOCAL` traffic pattern for 16K packets/PE. When considering time alone (*i.e.* clock frequency), the effective sustained bandwidth in the NoC is marginally higher for the LUT-based NoC due to the faster clock (see columns `Clk` in Table II), as shown in Figure 11a. When we factor in the energy required to route the workload, we see that the use of multi-pumping in the



(a) Time (ns) (b) Time+Energy (ns·J) (c) Time+Energy+Area (ns·J·LUTs)

Fig. 11: Bandwidth of LUT-based and DSP-based FPGA Overlay NoCs normalized to various metrics for 16×16 design. Here one DSP is considered equivalent to 120 6-LUTs in area based on approximations from [2].

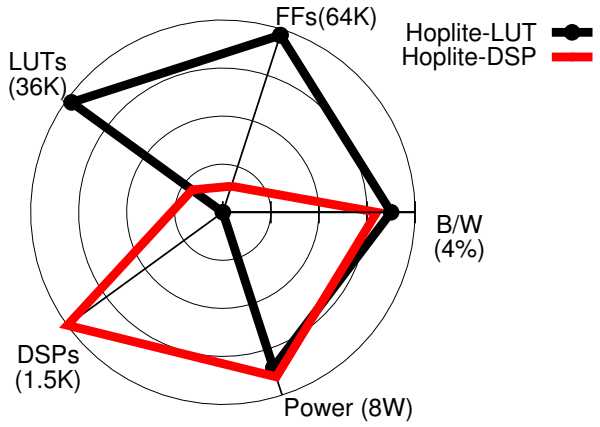


Fig. 12: Comparing LUT and DSP implementations of Hoplite NoC. Measuring resource utilizations (LUTs, DSPs, FFs) and metrics (power, sustained rates)

DSP blocks allows the LUT-based NoC to outperform the DSP-based design by ≈5–10%, as observed in Figure 11b. Finally, if we count resource cost in terms of normalized LUTs (1 DSP48 ≈ 120 6-LUTs)[2]. the DSP-based NoC is 3× less efficient in terms of normalized LUT resources (silicon area) as demonstrated in Figure 11c. This is expected as we are only using the multiplexer functionality of the DSP block leaving the arithmetic resources unused. For application that leave the DSP resources idle, the DSP mapping does not represent any loss in silicon area.

[2]Approximated from [2], where 1 DSP = 11.9 Altera LABs, each LAB = 20 Stratix-III ALUTs (4-LUTs). Rounding to 120 6-LUTs per DSP48.

C. Multi-Pumping

Multi-pumping is not strictly required for correct operation of Hoplite-DSP. Multi-pumping allows the user-side interface to retain identical cycle-by-cycle behavior as the LUT-based NoC. However, for most real-world multi-processor-oriented workloads that inject traffic into a NoC, we rarely observe injection rates of 100% and see rates between 15-20% instead. In these cases, we can avoid multi-pumping the DSP48 units at the expense of only supporting 50% peak injection rates from the PEs. We tabulate the reduction in power usage without multi-pumping in Table III. As expected, we get as much as 2× reduction in power by clocking the DSPs at the single slower clock. For streaming-oriented designs where 100% activity rates are possible, we can still revert to the multi-pumped design.

TABLE III: Comparing Multi-Pumping (100% injection rate) vs. Single Clock (50% injection rate) Power.

X	Y	Multi-Pump (W)	Single Clk. (W)	Ratio
32	16	7.2	3.3	2.1
16	16	4.1	1.9	2.1
8	8	1.1	0.7	1.5
4	4	0.5	0.4	1.2

D. FPGA Layout

We also visualize the horizontal and vertical congestion of the LUT and DSP-based 32b NoCs of size 32×16 for the VC707 board as shown in Figure 13. Both NoCs are able to keep congestion below 50% leaving adequate interconnect capacity for processing elements. LUT-based NoC has congestion hotspots scattered uniformly throughout while DSP-based NoC has larger hotspots in certain regions. This is somewhat

counter-intuitive as half the vertical traffic is absorbed by the DSP cascades. Closer inspection of the layout reveals that the DSP-based NoC requires pipelined loopback register paths unlike the folded torus layout of the LUT-based NoC. These seem to collect around the bottom portion of the NoC as determined by the Vivado placer. Furthermore, the vertical congestion is larger due to closer packing of the DSP columns (colored orange in Figure 13) in the FPGA. We also see spillover wiring extending beyond the NoC boundary. Both these effects result in somewhat larger congestion for the DSP-based NoC.

E. Board-Level Power Measurement

TABLE IV: VC707.

Reset	PLL
With DSPs	
14.6	15.2
Without DSPs	
14.6	15.4

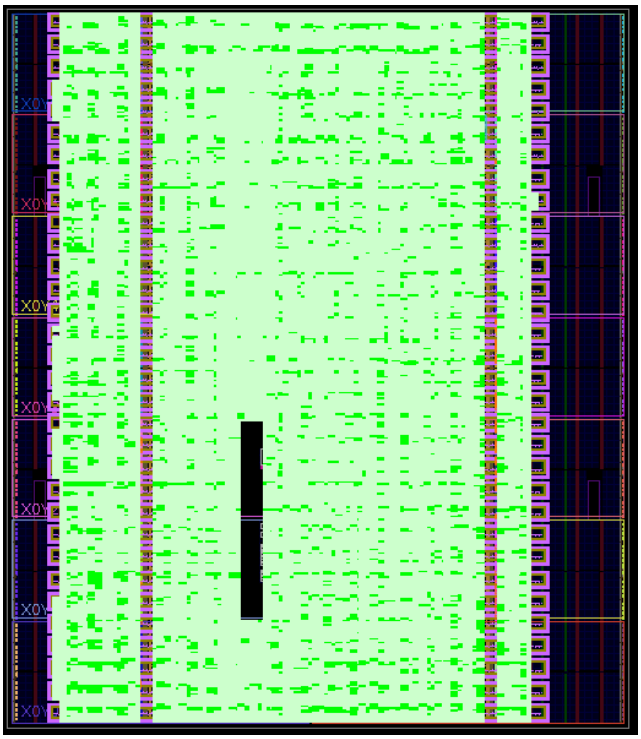
On the VC707 board, we mapped various NoC bitstreams and measured resulting power. In Table IV, we breakdown the various stages of the power measurements. Under FPGA Reset, the board still consumed a steady-state power of 14.6 W. When we only activated the PLLs while still keeping NoC operation disabled, this rose to 15.3–15.4 W. The slightly higher power measurement was for the PLL configuration driving the user clock as well as the multi-pumped DSP clock. We use the Active Power (= Total Power - PLL Power) attributed to the NoC operation in our efficiency calculations shown earlier in Figure 11.

VI. Discussion

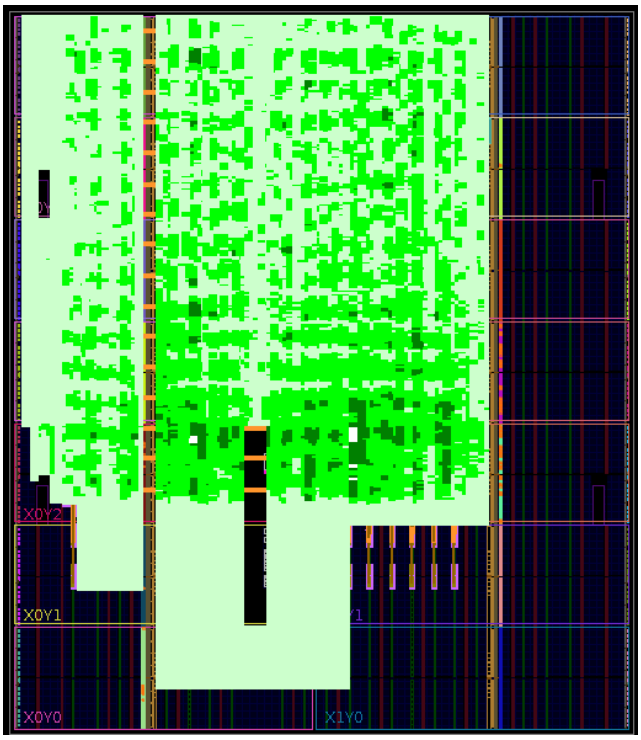
While the Hoplite-DSP design outperforms Hoplite-LUT on resource, speed and power metrics, it is also surprisingly competitive with a hard NoC [2], [1].

- From [2], we can approximate the resource utilization of a 1-VC (single virtual-channel design is closest approximation to our approach) 48b wide NoC router as 8.3 LABs$\times \frac{48}{32}$=12.45 LABs[3]. The paper also reports the DSP block to be 11.9$\times$ larger than a LAB. In relative terms, when adding the cost of DOR logic LUTs, this represents a small reduction in resource requirements over the 1-VC hard NoC router. For simplicity, we assume relative costs of Altera DSP and LAB to hold for Xilinx devices. We attribute this reduction to an absence of FIFOs, simplified control logic for deflection routing, and removal of other features that are supported by the hard NoC router. One could argue that the hard NoC is more capable than Hoplite, but for multi-processor traffic patterns, the deflection-based router is adequate and sufficiently capable. The few communication properties that Hoplite does not provide include in-order delivery, and low worst-case packet latency.
- Again from [2], we observe that the frequency of a 64-router 1-VC design to be 1035 MHz (32b) and 957 MHz (64b), which we approximate as 996 GHz for a potential 48b design. Our 8$\times$8 48b NoC with DSPs runs at 650 MHz

[3]See Section 5.3.1 in [2], projecting 32b data to a 48b design.



(a) Hoplite-LUT



(b) Hoplite-DSP

Fig. 13: Congestion Map for Hoplite with LUTs and DSPs on the VC707 board. (light-green 0–20%, green 20–40%, dark green 40–50%). DSP-based Hoplite has 40–45% vertical congestion and 20–30% horizontal congestion while Hoplite-LUT has 35–40% congestion along both dimensions.

(multi-pumped), for an eventual fabric-facing frequency of 325 MHz. This is representing a 3× performance reduction over a hard NoC. However, this is still better than the 6× advantage over a soft NoC claimed in [2].

- In [1], the authors report a 64-node hard NoC power consumption of between 1.21 W (32b) and 1.95 W (64b), which is averaged once again to 1.58 W for a 48b version. According the Vivado power models our 8×8 DSP-based NoC consumes only 1.1 W of power for a 15% activity rate resulting in a 43% power reduction. While this may be attributed to our slower clock frequency, this again reaffirms the closing of the gap over a hard NoC.

The current design of the DSP48E block provides a sufficient template for NoC embedding and actually works properly all generations of FPGAs that support the DSP48 primitive. For future FPGAs, we can significantly improve cost and performance of a DSP-based NoC, if we had the following wishlist additional features in the DSP.

- **Configurable Cascades**: We can approach Hard NoC behavior if we have freedom in directionality of the DSP cascade connectivity. In addition to horizontal cascade support, it would help significantly if we could *skip* DSPs entirely through some form of segmented routing for cascade links. The `A:B` cascades can be useful here, but in cascade mode, they do not permit the normal user input to be used. Furthermore, if horizontal cascade connections are provided, then the entire NoC wiring bandwidth can be offloaded from the global FPGA interconnect onto these dedicated resources. When not used by a NoC, these cascades can be used as ordinary DSP connections.
- **Pattern Detection Logic**: We can repurpose the pattern detector logic in the DSP to support DOR routing decoder. However, currently we are only able to pattern match against a single constant and mask, limiting it to single-dimension per DSP. For our case, we have to distribute the DOR decision logic across the vertical pass-through DSPs to make this work. The Altera DSP blocks [4] embed multiple constant coefficients internally which can be selected dynamically at runtime. Having similar freedom for storage of multiple masks and patterns would allow complete embedding of DOR decoder functionality inside the Xilinx DSP48 block as well.
- **Cascade Pipelining**: The cascade paths entering the DSP block do not have any pipelining support and have to rely on `PCOUT` pipelining from the previous DSP stage. Having configurable support for `PCIN` input pipelining with suitable clock enables would help align inputs and potentially simplify DOR decoder design.
- **SIMD Multiplexing**: While the DSP48E ALU supports configurable SIMD mode with the ability to fracture the 48b datapath into 2×24b or 4×12b implementations, the multiplexer input bits cannot be independently switched. With `OPMODE` configuration support for SIMD functionality, we can separately control two or four concurrent NoCs to permit an additional degree of flexibility to the user.
- **Energy Efficiency**: We expect multi-pumped designs to consume more power that is proportional to the increase in frequency ($P = f \cdot C \cdot V^2$). However, the use of hardened DSP resources helps us improve this gap [8] for most system sizes (except 32×16). At these larger system sizes, there may be opportunities to better tune the complete DSP fabric for energy-efficient multi-pumping through configurable power-gating of unused arithmetic functions in the DSP.

VII. Conclusions

We demonstrate a novel use of the Xilinx DSP48E blocks to implement expensive multiplexer operations in deflection-routed Hoplite NoC. For FPGA designs that do not fully utilize all available DSP units, or focus on other applications such as cryptography, this DSP-enhanced NoC conserves LUT resources while routing NoC traffic over hard DSP units and hard inter-DSP cascades. For 32×16 NoC mapped onto a VC707 board with the Xilinx XC7VX485T FPGA, we consume 1.5K DSP48E blocks (56%) but reduce LUT costs from 35K to 7K (5× saving), FF costs from 62K to 9K (6× saving), at the expense of increasing active power draw from 6.8 W to 7.2 W (15% activity rate, 5% increase) at 300 MHz user clock (DSP clock is 600 MHz). For smaller system sizes, the use of hardened DSP48 blocks and cascade wiring delivers lower power utilization by 1.1–2× over a LUT-based implementation. We also demonstrate practically no difference in silicon area, only a 3× loss in clock speed, coupled with 43% lower power requirement over a conventional single-virtual channel hard NoC router. This allows our DSP-based Hoplite design to close the gap over hard NoC resources championed in prior work. As part of future work, we also intend to investigate non-torus NoC topologies which may be a better fit to the directional nature of the DSP cascades.

References

[1] M. S. Abdelfattah and V. Betz. The power of communication: Energy-efficient NOCS for FPGAS. In *2013 23rd International Conference on Field programmable Logic and Applications*, pages 1–8, Sept 2013.

[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] M. S. Abdelfattah and V. Betz. The Case for Embedded Networks on Chip on Field-Programmable Gate Arrays. *IEEE Micro*, 34(1):80–89, Jan 2014.

[4] Altera Inc. *Enabling High-Performance DSP Applications with Stratix V Variable-Precision DSP Block*, 2011.

[5] A. Canis, J. H. Anderson, and S. D. Brown. Multi-pumping for resource reduction in FPGA high-level synthesis. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 194–197, March 2013.

[6] H. Y. Cheah, S. A. Fahmy, and D. L. Maskell. iDEA: A DSP block based FPGA soft processor. In *Field-Programmable Technology (FPT), 2012 International Conference on*, pages 151–158, Dec 2012.

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

[8] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):203–215, Feb 2007.

[9] C. E. LaForest and J. G. Steffan. Efficient multi-ported memories for FPGAs. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 41–50, New York, NY, USA, 2010. ACM.

[10] Xilinx Inc. *Large FPGA Methodology Guide*, 2012.

[11] Xilinx Inc. *UltraScale Architecture DSP Slice User Guide*, 2014.