On Bit-Serial NoCs for FPGAs

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

Abstract—
We can build lightweight bit-serial FPGA NoC routers that cost 20 LUT, 17 FF per router and operate at 800–900 MHz speeds. Each bit-serial router implements deflection-routing on a unidirectional torus topology requiring 1b-wide connection per port. The key ideas that enable this implementation are (1) reformulation of the dimension-ordered routing (DOR) function using compact 1 LUT, 1 FF streaming pattern matchers, (2) compact retiming of the datapath signals into SRL16 blocks, and (3) careful FPGA layout to efficiently pack the router logic into small rectangular regions 2×4 SLICEs on the chip. We anticipate these bit-serial NoCs can be used in a variety of scenarios including overlay support for triggered debug, lightweight control signal dissemination, massively-parallel bit-serial processing.

I. Introduction

Bit-serial representation has long been a cornerstone of both low-cost (embedded) and high-speed (high-performance) communication at the system level. However, inside the chip, communication subsystems tend to operate on wide words (*e.g.* 32b/64b AXI interfaces). FPGA overlay NoCs for shared routing of communication inside the chip also operate on wider words; for instance, the chip-spanning Hoplite [7] overlay NoC is 32b wide@300 MHz (Virtex-6 ML605 board) while the GRVI Phalanx NoC [5] is 290b wide@400 MHz (Kintex UltraScale KU040 board). This preference for wider link widths is natural, as on-chip wiring on modern FPGAs is abundantly available. This is in contrast to system-level constraints on the printed circuit board or enclosures where bit-serial links are preferred to wide bit-parallel cables. In this paper, we investigate whether bit-serial communication can be cheaply and effectively supported *inside* the modern FPGA chip. High-end commodity FPGAs are large devices and some even span multiple dies thereby placing a premium on global routing resources for supporting wide NoC links all over the chip. We envision this bit-serial NoC to be part of a communication system for an FPGA for debug support, distribution of lightweight global control signals, and other monitoring needs. The savings in global interconnect resources are then returned back to the spatial application instead of getting locked up in the NoC infrastructure. For instance, Figure 1 shows placed and routed layouts of 4×4 and 8×8 bit-serial NoCs occupying negligible LUT/FF resources (≈0.5%) as well as very little of the interconnect fabric while running as fast as 800-900 MHz which is 2–3× faster than equivalent bit-parallel NoCs [7], [5], [10], [6]. The per-bit area-delay product of the sub-word NoCs that employ a serial transmission style advocated in this paper matches that of wide-word NoCs.

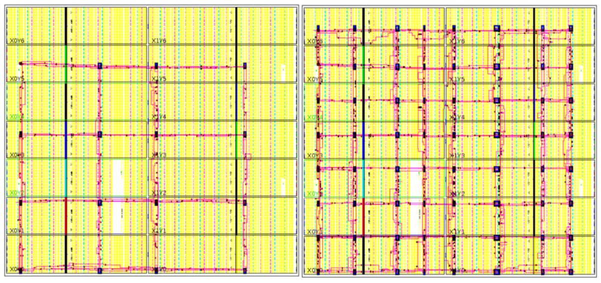

(a) 4×4 NoC (b) 8×8 NoC

Fig. 1: FPGA layout of bit-serial NoCs on the Xilinx VC707 board (XC7V485T) as seen in Vivado "Device View" after Place and Route. (804 MHz, and 909 MHz)

We anticipate bit-serial infrastructure to be useful in routing-constrained scenarios.

Specifically, we envision a bit-serial NoC implementation can support a variety of use cases enumerated below:

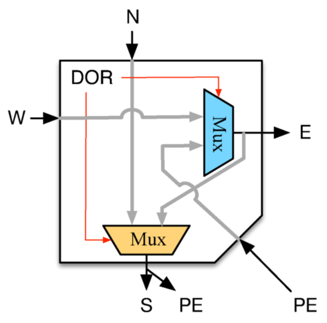
- **Debug Overlay**: Triggered FPGA debug logic [4] relies on the ability to tap various internal signals from a user design and routing it to monitoring ports or RAMs. A bit-serial NoC can effectively gather intermittent signal events without requiring persistent high cost connections to the FPGA debug buffers.
- **Bit-Serial Arithmetic**: Bit-serial arithmetic processors can provide a competitive resource-efficient, high-performance, and even energy-efficient design for many scenarios [8]. A bit-serial NoC can permit assembling a massively parallel array of such bit-serial processors in the flavor of a Connection Machine CM2/200 system from the past.

The key contributions of this paper include:

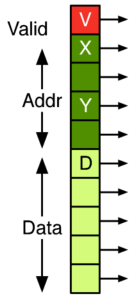
- RTL design and verification of a bit-serial deflection router for unidirectional torus overlay NoCs.
- Architecture-specific enhancements to support (1) LUT-level resource sharing optimizations to minimize route decoding cost, (2) retiming of delay balancing registers for tight SRL16 mapping, and (3) sub-word parallel extensions with bit-serial decoding for efficiency.
- Parametric pipelining and FPGA layout considerations for high-speed implementations for various NoC sizes on the Xilinx VC707 board (XC7V485T FPGA).

II. Bit-Parallel Hoplite Router

In this section we discuss the bit-parallel designs for an FPGA overlay router that is used in this work as well as highlight the key limitations of the design.



(a) Bit-Parallel Hoplite



(b) Bit-Parallel Packet

Fig. 2: Bit-Parallel Hoplite and packet format.

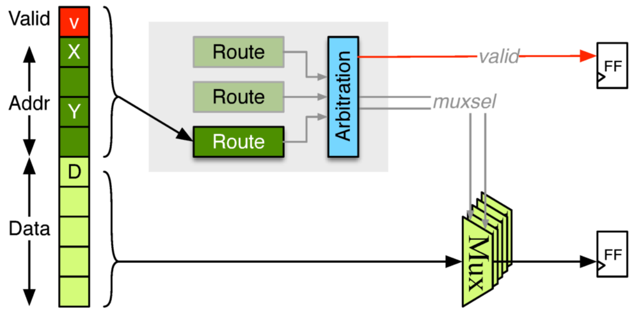


Fig. 3: Path of a packet through the bit-parallel Hoplite switch. (only showing one input to one output path)

A. *Base Hoplite*

We use the Hoplite bufferless, deflection-routed design [7] as a starting point for our work. Among existing published FPGA routers, the Hoplite design is simple, lean and most amenable to bit serialization. Other routers uses complex virtual-channel based designs [10] or use expensive FIFO buffers [6] throughout the design resulting in high FPGA costs. A high-level picture of Hoplite router is shown in Figure 2a and the packet format is shown in Figure 2b. The underlying unidirectional torus topology helps simplify the switching crossbar to a small three input, two output solution. Cascading of the switching multiplexers allows them to be mapped to a two 5-LUTs packed into a single 6-LUT on a Xilinx FPGA. This is the most efficient solution possible for implementing the switching multiplexer on a Xilinx FPGA. As this is a deflection routed NoC, the router has no FIFOs at the I/O ports and all incoming flits are routed to output ports. However, the PE injection port is allowed to be blocked if the network is busy. The address portion of the packet is used as input to the Dimension Ordered Routing (DOR) function block to determine which packet travels in which outgoing direction. A key restriction of the design is that packets have a fixed length of one as this is a deflection routed design that cannot support wormhole routing on longer packets without complicating the design [9].

B. *Dataflow inside Hoplite*

Once a packet arrives inside a Hoplite switch, the different portions of the packet are processed as shown in Figure 3. At a high level, there are two planes of operation: (1) control (address) plane, and (2) payload (data) plane. In the control plane, the routing decision logic inspects the address bits of the valid packet and detects permissible routing directions for the packet. As Hoplite uses DOR routing, packets traveling in `Y` dimension arriving on the North input can only continue South or exit the router. Packets traveling along the `X` dimension arriving from the West input can exit on any port as long its not already occupied. In all cases, if the desired output port is already assigned based on DOR priority, the packet is deflected. All these decision are made relatively cheaply with a few LUTs of logic depth to decide the outgoing valid signal indicators for the various directions. In the data plane, the result of the control plane are used to steer the multiplexers to the output ports. These multiplexer select signals are also controlled by the arbiter. All of this happens in a single cycle from input to output. A 32b implementation of this datapath on a Xilinx VC707 (XC7V485T) takes up 88 LUTs and 80 FFs while running at 1.5 ns (400 MHz). We tabulate the resource breakdown of the 32b Hoplite router in Table I. This result is slightly different from the 60 LUT, 100 FF design in [7] due to variation in the underlying FPGA architecture, CAD tool version, and mapping options.

TABLE I: Resource Breakdown of a single 32b (8b address) Hoplite Bit-Parallel Design. Vivado 2015.4, XC7V485T -2 FPGA, Settings: `Flow_AreaOptimized_high` + `post_route_phys_opt_design`

Component	LUTs		FFs
	Logic	SRL16s	
Switch Multiplexer	72	0	0
DOR logic	16	0	0
Output Registers	0	0	80
Total	88	0	80

C. *Key Limitations*

While Hoplite is an excellent router for wide FPGA overlays, there are limitations that may affect its broader appeal:

- **Interconnect Cost**: The Hoplite design is engineered to reduce the number of LUTs and FFs consumed by the architecture but restricts how the payload is routed. It is not possible to route packets with multiple flits due to the deflection-oriented nature of the NoC. Wider payloads are necessary if the addressing overheads are to be avoided during each flit transmission. Wider payloads consume more LUTs/FFs and also displace valuable general FPGA interconnect resource away from the spatial application. This may also result in underutilization of the provisioned wiring for bursty or streaming traffic patterns. For instance, Vivado 2015.4 reports horizontal and vertical interconnect congestion of 60–70% for a 32b-wide 16×16 Hoplite NoC. If wider payloads are not possible, we are then forced to send multiple flits with redundant address bits in each flit.

- **Fabric Frequency**: While individual Hoplite routers are simple and have few logic levels (LUT depth), when composed together in a system, the achieved frequency drops significantly even with floorplanning hints. Most of the mappings resulted in 300-450 MHz layouts for 32b designs which seem fast for contemporary FPGA designs; however, the underlying wires can run much faster[1]. As an example, without connections to other Hoplites, and compact floorplan, a 32b Hoplite router can run at 600–700 MHz. This depends on the number of payload bits as quantified later in Section V (Figure 12a and Figure 12b).
- **Register Use**: High-speed designs for chip-spanning Hoplite NoCs are only possible at the expense of extensive pipelining of the NoC links. While this does let the design achieve the 300-450 MHz speeds, it further steals vital FF resources away from the user designs. The 32b-wide 16×16 Hoplite NoC occupies 36K FFs which are about 6% of the chip.

To summarize: (1) Bit-parallel Hoplite design is economical in terms of LUT and FF usage per bit, (2) When considering wiring requirements, addressing overheads and circuit frequencies, they are expensive and inefficient. We seek to address these limitations as part of this work. With a high-speed bit-serial design we want (1) to lower the amount of global FPGA interconnect devoted to the NoC, (2) to allow the NoC to run as close-to-GHz speeds as possible, and (3) leave valuable registers available for user design. This also allows us to close the frequency gap with the hard NoC routers [1] which are capable of running at 900–1000 MHz, admittedly for a much narrowed bitwidth.

III. Bit-Serial Hoplite FPGA Router

In this section we describe our bit-serial Hoplite router design and associated FPGA architecture-aware optimizations that enable a cheap implementation. We first introduce a baseline bit-serial design to highlight the key principles required for composing the design. We then describe the specific optimization to each component to help deliver the LUT-optimized final solution.

A. Signal Format

Classic bit-serial signal representations include start/stop bits to enclose a fixed-length packet to support point-to-point asynchronous transmission behavior along with any error correction overheads. However, as deflection routing is switched (not point-to-point) and it cannot support multi-flit wormhole routing, we make some simplifying assumptions. All packets are of fixed-length and are injected at specific periodic intervals (instead of asynchronous any-time injection) determined by the packet length. For instance, we can assume that a packet is of 32b length (inclusive of valid, address, data) and that the PEs can inject packets once every 32 cycles starting from the designated cycle slot. This discipline ensures that packets arrive aligned at each switch at exactly the same cycle from all inputs. This prevents a deflection routed switch from receiving a packet in the middle of routing an earlier packet. As the processors are injecting bit-serial packets, this restriction is in fact in agreement with the internal timing of the PEs anyway. This can be implemented with 5-bit counter per PE mapped cheaply to a single SRL16 and a FF wired in loopback fashion. Our bit-serialized signal representation is shown in Figure 4.

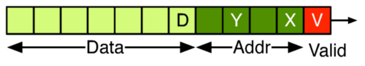


Fig. 4: Bit-serial Packet. (in-band valid and addr/data)

B. Routing and Arbitration

We show a high-level view of bit-flow through the core of the bit-serial Hoplite router in Figure 5. A key challenge in designing a bit-serial router is to keep pipelining costs low. As routing decision will be made at a clock cycle that's different from arrival of data, we may have to store the bits while the decision is pending.

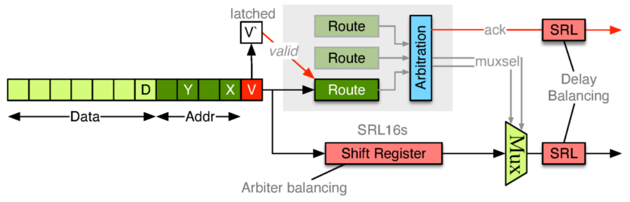


Fig. 5: Dataflow through a Simple Bit-Serial organization. (only showing one input to one output path)

As the address bits present themselves in sequence, we need to implement a simple streaming pattern matcher that detects the `X` and `Y` addresses of the packet. Once the XY matching all three input bit-streams (North, West and PE input) have been computed, the arbiter can make routing decisions and generate the multiplexer control and valid signals for the router output ports. The valid input (`v`) is sent in-band with the rest of the address/payload bits, so we separate it out and latch it into a persistent version for the duration of the packet (`v'`).

We illustrate the internal design of the bit-serial DOR (dimension-ordered routing) decoder and pattern matching logic in Figure 6. For DOR routing, we need to route packets along X dimension first, before they turn to Y. No turns are possible from Y to X. Here, we separately process the `X` and `Y` addresses of the incoming packet on all three input ports; North, West and PE. The resulting match indicators must be delayed through SRL16 shift registers[2] to ensure they arrive at the arbiter at the same cycle for correct decisions. The decoder itself costs a 5-LUT (SRL16) to delay the valids and

[1]Vaughn Betz FPT 2016 keynote, 730 MHz short wires. Hard wires can go as fast as 900 MHz [1]

[2]An SRL16 Xilinx primitive allows us to pack a 16-deep shift register into a fracturable Virtex-7 5-LUT. An SRL32 uses up a 6-LUT.

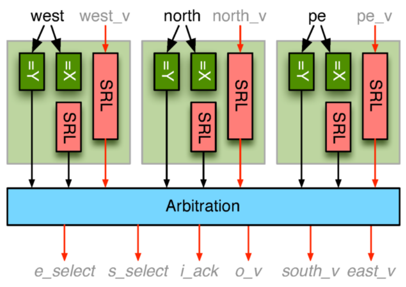


Fig. 6: DOR (Dimension-Ordered) decoder for bit-serial Hoplite.

X match signals per input. The matching logic is a simple streaming XOR that fits a 5-LUT easily. The XY address (pattern) is stored in a pattern ROM SRL16. Overall, this design requires 13–14 LUTs for the matching logic and 4 LUTs for the arbitration (one for each output valid, and one for acknowledge). Depending on Vivado synthesis and mapping optimization options, we observed 18–20 LUTs and FFs for the decoder component of the design.

C. Delay Balancing

The end-to-end delay in decision making requires the router inputs be suitably delayed to align them properly at the multiplexer inputs (shown by the *Arbiter balancing* SRL in Figure 5). Furthermore, once multiplexed, the outputs must also be delayed to ensure the downstream switches observe arriving inputs at packet length boundaries (labeled *Delay Balancing* in Figure 5) . This requirement is essential as all inputs arriving at each switch must be aligned with each other for correct operation. This is not a problem with the bit-parallel design as the complete packet arrives in one shot in the same cycle. As deflection routing cannot support wormhole routed packets (*i.e.* packets with multiple flits) arriving at arbitrary cycles, we must align all inputs to arrive at the same cycle in all router. Thus, this results in a substantial cost for implementing shift registers along all inputs and outputs. This results in a usage of 4 LUTs and 4 FFs per input yielding an overall cost of 12 LUTs and 12 FFs just for delay balancing component of the router. The SRL cost here is a function of packet length L and will be $\lceil \frac{L}{16} \rceil$ SRL16s per port.

We tabulate the LUT and FF resources of this baseline design in Table II. As expected the switch multiplexing costs have dropped linearly with number of bits being switched down to 1 LUT from 72 LUTs in Table I. However, the DOR matching costs have grown to require 10 LUTs, 8 SRL16s and 13 FFs over the simple 16 LUT design of Table I. It may be tempting to perform a series-to-parallel conversion of the XY bits and reuse the bit-parallel DOR design but that would also require a bank of FFs (12 FFs for 4×4 NoC). When considering the area-delay product (per bit) of the two designs, the bit-serial design is significantly worse (88 LUT×1.5 ns/32b vs. 30 LUT×1.4 ns/1b = 6.7× worse). While we do not expect to completely overcome the area-delay product metric based on current FPGA limits, we can attempt to close this gap with sub-word design explained later in Section III-E. On the Xilinx FPGA, we can map all the shift registers efficiently into SRL16 primitives.

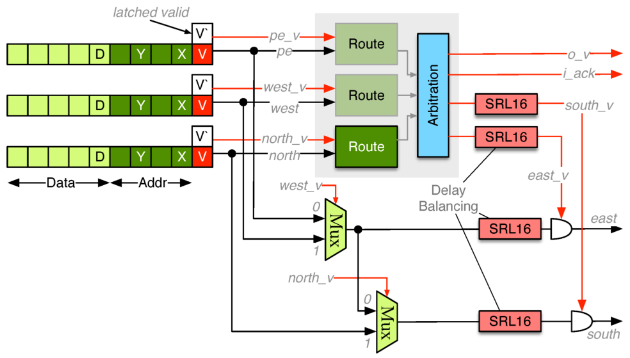


Fig. 7: Dataflow through an Optimized Bit-Serial organization.

TABLE II: Resource Breakdown of Baseline Design.

Component	LUTs		FFs
	Logic	SRL16s	
Switch Multiplexer	1	0	0
DOR Logic	12	8	14
├Pattern Matching	6	8	11
└Arbitration	6	0	3
Arbiter+Delay Balancing	0	8	12
Total	13	16	26
	(30 LUTs)		26

D. LUT-crafting (Optimizing) the Design

While the baseline bit-serial design is compact it still occupies 30 LUTs and 26 FFs while running at 1.4ns clock. This design routes $\frac{1}{32}$ the number of bits while reducing logic utilization by a mere 3.3–3.6×. We now show how to bring this down to the advertised 20 LUT, 17 FF solution that runs at 1ns. While this may seem like a modest saving, this reduction is crucial for two reasons: (1) to enable high-speed operation, and (2) to deliver per-bit, per-ns efficiency of LUT use when scaling up to larger NoC sizes.

- **Logic-free multiplexer selection**: The cascaded muxes we highlighted in Figure 2a as suitable for fracturable 5-LUT implementations have another useful property. The multiplexer select signals can directly be driven by the valid indicators as shown in Figure 7. This simplifies the arbitration logic and saves a handful of LUTs and latches. This is trivially possible as the first multiplexer chooses between the West and PE inputs and can directly be selected based on whether there is a valid packet along the West input. In this case, the PE acknowledge signal is relayed to the PE indicating the unavailability of the first multiplexer for any data exiting the PE. The PE then has to reattempt packet injection at a later cycle. The next stage of the multiplexer selects between the North input and the (West or PE) input. Here, we prioritize the North input in all cases. There is a sub-optimal choice being made here in the case

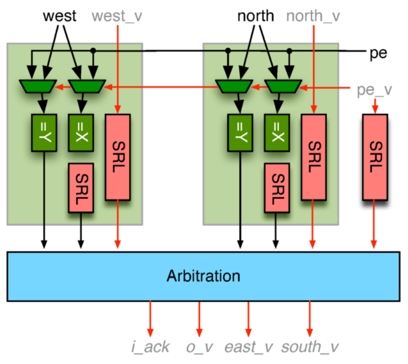


Fig. 8: Optimized DOR decoder for bit-serial Hoplite.

the PE input wants to route to South output in a conflict free manner with the West input wanting to travel East. This scenario will result in the PE getting blocked but this has a low impact of overall routing performance across different traffic patterns. Furthermore, the same effect is also present in the base Hoplite design using this fracturing strategy. We have to AND-mask the East-bound and South-bound outputs to validate the presence of NoC traffic on that link.

- **Retiming**: As the multiplexer selection happens instantly, the only delay in the output packet generation is the computation of valid signals. Thus we can now provide a single level of SRL16 delay bank for all outgoing wires. This further eliminates the arbiter balancing SRL16 from Figure 5. Additionally, when the inter-router links are registered for performance, we can simply borrow delay from this SRL16 bank to improve clock speed without losing end-to-end packet traversal latency. This is particularly valuable as inter-router links quickly become the performance bottleneck when mapped to span the complete chip real-estate.
- **Resource Shared DOR decoder**: As shown earlier, we can implement dimension ordered routing using bit-serial pattern matching on each of the three input channels. However, we observe that with the cascaded multiplexer design we can only allow two connections in the switch in a decision cycle. More specifically, the PE input will be ignored (not acknowledged) if there are valid packets on both the North and West inputs. This means, we can resource share the bit-serial pattern matchers with the PE input. This is possible using a 2:1 mux inserted at the input of the pattern matching logic. Luckily, this still fits the fracturable 5-LUT input constraints resulting in a net saving in LUT cost (3 LUTs and 1 FFs saved). We show this in Figure 8. Furthermore, the arbitration is also simplified down to 4 LUTs as the multiplexer selection signals no longer need to be explicitly generated (can directly use zero-LUT arbitration, and use valids as mux-control themselves).

- **Pattern Matcher**: We compactly implement the pattern matching logic for X or Y detection using a single LUT for streaming detection and an SRL16 to store the bit pattern of the address. In Figure 9, we show the logic equation that is mapped to the LUT for implementing this computation.

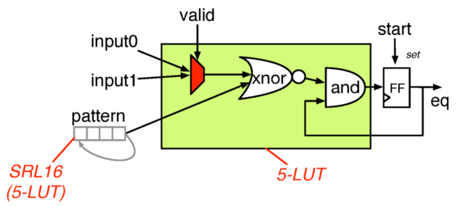


Fig. 9: Bit-serial Pattern Matching Circuit.

The input multiplexer chooses between the PE input or a NoC link based on the validity of the data (valid is latched). The XNOR gate compares the arriving bit-stream against a pattern generator signal that is aligned with data arrival. The feedback signal logs the match state across the relevant bit subset of the input data.

We tabulate the LUT and FF resources of this optimized design in Table III. Compared to the baseline bit-serial design summarized in Table II, the *lutcrafted* design takes up 50% fewer LUTs and FFs. These reductions are primarily due to the lack of requirement for *Arbiter Balancing* SRLs and resource-sharing of the DOR decoder logic. As hinted at earlier, we expect the HyperFlex registers on the Altera Stratix 10 to absorb most of the Xilinx SRL16 shift register usage further shrinking the design cost when mapped to the Altera device. As the core switch multiplexer is $1/20$=5% of the logic utilization of the router, this design trades off excess LUTs for keeping wiring costs low.

TABLE III: Resource Breakdown of Optimized Design.

Component	**LUTs**		**FFs**
	Logic	**SRL16s**	
Switch Multiplexer	1	0	0
DOR Logic	8	6	9
├Pattern Matching	3	6	9
└Arbitration	5	0	0
Delay Balancing	0	5	8
Total	9 (20 LUTs)	11	17 17

E. Sub-word parallel design

For sub-word-wide NoC routers with payload widths such as 2b–8b, the use of a full-blown bit-parallel design is wasteful as the XY address bits would add significant overhead to each packet transmission. For instance, a 4×4 NoC for 4b-wide payload will consume roughly 50% of the wires for routing addressing information (2b `X`+2b `Y` address). In these scenarios, using a bit-serial design may be a more efficient starting point due to better use of wiring resources. In this case, we simply add a LUTs for each extra bit to be routed to the "Switch Multiplexer" row of Table III while retaining rest of the logic without modification. Thus, the address bits are still routed serially, with additional wires used purely for routing the payload bits. If the total data that must be transferred per packet is large and split across multiple cycles of the bit-serial design, this also allows us to keep address overhead per packet to be low.

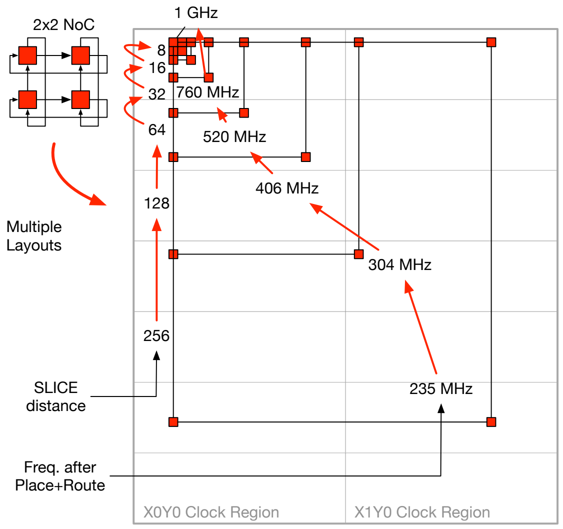


Fig. 10: Floorplanning Exploration of 2×2 bit-serial NoC without any inter-router pipelining registers. Evaluating the effect of spatial distance between routers on Freq.

IV. FPGA Mapping and Layout

In the previous section we showed how to design the bit-serial router to minimize the number of LUTs and FFs used by the implementation. In this section, we investigate the floorplanning potential for helping boost frequency of our design.

A. *Speed Calibration*

The base Hoplite router can run at 450–500 MHz on the VC707 for various system sizes from 2×2 to 16×16 routers. This requires pipelining the inter-router links. To clearly understand the extent of pipelining requirements of the bit-serial NoC, we placed and routed a 2×2 NoC with various inter-router distances first without pipelining and then with a configurable number of registers. In Figure 10, we visually represent the results of these frequency calibration experiments. We tabulate the impact of spatial distance and pipelining on circuit frequency for the 2×2 NoC in Table IV.

Impact of Distance: When the distance between the bit-serial routers is below 8 slices, we can easily meet even a 1 GHz timing constraint. Obviously this particular layout is not directly useful, but it points to an upper bound that should be achievable. As we widen the gap between the routers, we see a frequency degradation down to 235 MHz when using a 180×256 (X×Y) gap. Varying distance helps us achieve the full spectrum of frequencies between these two extremes. The PBLOCK constraints (physical chip region limits) for the four NoC routers seen in the device layout in Figure 10 are shown as red rectangular regions.

TABLE IV: Frequency Scaling of a 2×2 NoC on the Xilinx VC707 board (XC7V485T).

(a) Impact of Distance (No pipelining)

Distance (SLICEs)	2	4	8	16	32	64	128	256
Frequency (MHz)	1000	950	940	760	520	406	304	235

(b) Impact of Pipelining (180×256 SLICE inter-router distance)

Pipelining (FFs)	0	1	2	3	4	5	6	7
Frequency (MHz)	235	454	524	598	633	633	708	715

Impact of Pipelining: We conduct a separate experiment where the inter-router links are pipelined with a configurable number of registers to evaluate frequency improvements. In this scenario, Vivado's analytic placement engine is able to position registers at appropriate intervals along the link. The addition of pipelining registers allows the design to run as fast as ≈700 MHz. While nowhere close to a GHz rate, the long distance between the router links poses a challenge to the Vivado CAD engines for placing intermediate pipelining registers at suitable distances and choosing the appropriate routing segment. For larger-sized NoCs such as 16×16 NoCs, we are able to achieve a faster clock rate (800–900 MHz) with ease. In this case, the router PBLOCK constraint serves as a convenient place for limiting the wiring distance and delay. Thus the experiments with layout of a 2×2 NoC push Vivado to the limit and help us understand the worst-case performance possible for various configurations.

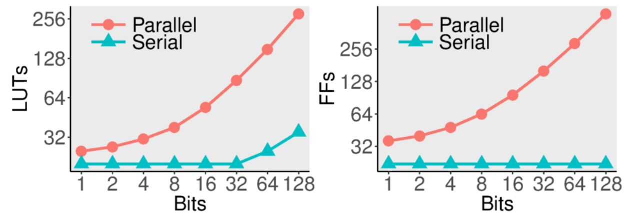
Folded Layout: For larger NoCs, the wrap-around links between the extreme edges of the NoC are long and can constrain performance. To avoid this problem, we use a folded layout by interleaving alternate router blocks thereby limiting the worst-case inter-router distance to just two routers. This is a well-known idea also used for performance optimization in the base Hoplite design.

V. Evaluation

In this section, we evaluate the various configurations of the different NoCs and report NoC metrics, resource utilization, clock frequency, and power usage information. We then understand the underlying trends and tradeoffs for optimizing the NoC for the end user. As mentioned earlier, we perform all our experiments on the Xilinx VC707 board with the XC7VX485T -2 FPGA device and use Vivado 2015.4 for compilation. Where appropriate we generate PBLOCK floorplanning constraints using our custom scripts for targeting various NoC system sizes. We use parameterized RTL based on the original Hoplite RTL with suitable adaptations for supporting bit-serial evaluation. The Clock BUFG F_{max} frequency range [11] that is supported by the Virtex-7 fabric is 625–741 MHz (depending on speed grade). Our layout frequencies easily surpass these limits, and represent a limit study of what the Xilinx fabric potential.

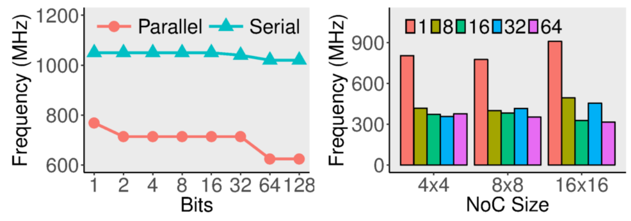
A. *Single Router*

We first assess the logic requirements of individual routers to compare and contrast the bit-parallel and bit-serial implementations. In Figure 13a and Figure 13b, we show the effect of varying bitwidth of the NoC link on LUT and FF count of the routers on the VC707 board. As expected the bit-parallel designs can be as much as 8× (LUTs) and 23× (FFs) larger than their bit-serial counterparts. The bit-serial design is mostly insensitive to variations in bitwidth (certainly FF



(a) LUTs (b) FFs

Fig. 11: Comparing LUT and FF requirements of the different bit-serial and bit-parallel NoC routers.



(a) Single Router (b) System-Level

Fig. 12: Clock Frequency of bit-serial and bit-parallel NoC routers.

counts stay constant), we observe a minor increase in LUT counts to store the address patterns in SRL16s.

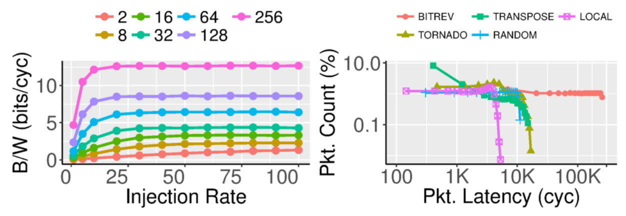
Next, we evaluate the clock frequency of the stand-alone routers that is achieved when mapping the routers without any constraints. In Figure 12a, we observe a frequency gap when comparing bit-serial over bit-parallel routers as large as 1.8×. Without the effect of loading from inter-router links, the individual bit-serial routers easily run as fast as a GHz. The bit-parallel designs, on the other hand, slow down by as much as 30–40% as we increase the bitwidth of the packets.

Thus, bit-serial routers are as much as 2.3× faster and require 8× fewer LUTs and 23× fewer FFs compared to 128b-wide routers. In terms of per-bit efficiencies the bit-serial routers lag behind their bit-parallel counterparts when the payload widths are large enough to amortize the addressing costs. When mapped, the FPGA interconnect fabric shows a

TABLE V: FPGA Utilization and Network Congestion Data reported by Vivado (XC7V485T mapping).

NoC	**Logic Utilization**		**Route Utiliz. (%)**		**Route**
Size	**LUTs**	**FFs**	Vert.	Horiz.	**Congest.**(%)
		Bit-Parallel 64b NoCs			
16×16	57.K (19%)	54K (9%)	7.5	6.7	66-89
8×8	8.7K (3%)	8.7K (1.5%)	0.5	2.5	47-65
4×4	2.1K (0.7%)	2.2K (0.3%)	0.2	0.5	50-53
2×2	0.4K (0.1%)	0.5K (0.1%)	0.3	0.6	27-51
		Bit-Serial NoCs			
16×16	6.4K (2%)	7.1K (1%)	0.3	0.2	Nil[1]
8×8	1.6K (0.5%)	1.8K (0.3%)	0.1	0.07	Nil[1]
4×4	0.4K (0.1%)	0.5K (0.07%)	0.08	0.03	Nil[1]
2×2	0.1K (0.02%)	0.1K (0.02%)	0.02	0.02	Nil[1]

[1]Vivado does not even report Congestion metrics for these cases.



(a) LUTs (b) FFs

Fig. 13: Throughput measurements of the different bit-serial NoC system sizes for uniform `RANDOM` workload at various injection rates. Latency distribution at 256 PEs at 10% injection rate.

Route Congestion factor of 20–30% for bit-parallel designs while only 1–2% for bit-serial designs. Route congestion factor captures the difficulty faced by the Vivado router in finding routes that meet user timing targets for the different circuit paths (lower values are better).

B. NoC System-level Metrics

We show performance (throughput, and latency) metrics of the bit-serial NoC in Figure 13 for workloads routing 1K packets/PE. The NoC throughputs saturate at various system sizes at injection rates above 20% in a manner identical to Hoplite [7]. The latency distribution also mimics Hoplite behavior and exhibits a long tail effect (few packets suffer long worst-case deflections). As expected `LOCAL` traffic has shorter packet latencies, while the `BITREV` workload has the worse latencies that are 10–20× larger than even uniform `RANDOM` traffic. The absolute throughputs and latencies for bit-serial NoC are derated by a factor equal to packet length which is expected as the wiring requirements are also reduced accordingly.

When composing N×N NoCs, the frequency degradation and route congestion effects become more pronounced for the bit-parallel design. In Table V, we tabulate the LUT, FF, Frequency, and Routing Congestion costs reported by Vivado for different sizes of the bit-serial and bit-parallel NoCs (64b).

In Figure 12b, we show the system-level clock frequencies achieved by the various NoCs for different system-sizes and bitwidths. The frequency of the bit-serial NoCs are 2–3× faster than equivalent bit-parallel routers even when the inter-router links are optimally pipelined. The larger bit-serial NoC (16×16) runs the fastest ≈900 MHz as the inter-router spatial distance is shorter than the other routers. For the smaller NoCs (4×4), the inter-route gaps are wider and must be explicitly pipelined with extra output registers. We rely on Vivado's excellent placement engine to identify the precise positions for these inter-router pipeline stages. However, we do observe an ≈100 MHz frequency drop for the smaller NoCs (as discussed earlier for the toy 2×2 NoC in Speed Calibration in Section IV-A) despite adequate pipelining due to the limits of the automated placement engine.

In Figure 14, we highlight the routing congestion on the FPGA when comparing bit-serial and bit-parallel NoCs. As is

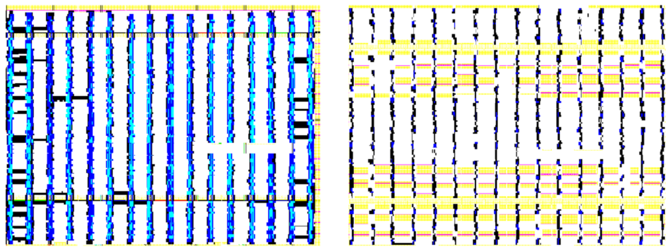

(a) 64b Bit-Parallel (b) Bit-Serial

Fig. 14: Routing Congestion Maps of 16×16 NoCs on XC7VX485T. Blue regions represent high congestion (dark: 20–50%, light: >50%), Black regions indicate moderate congestion 10–20%, White regions are low congestion <10%, Yellow regions have no routing.

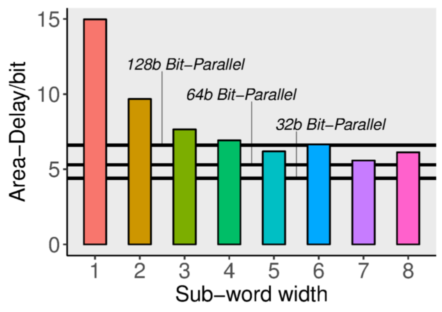


Fig. 15: Figure of Merit (Area-Delay product per bit) for the Sub-word Bit-Serial design vs. Bit-Parallel Design.

clear, the bit-parallel NoC exhibits congestion hotspots that are up to 90% congested in certain regions. In the bit-serial NoC, apart from a few moderate spots with ≈20–30% congestion, the chip is mostly empty. User designs are not only free to use the logic capacity of the chip but also most of its routing resources.

C. Area-Delay of Sub-Word Designs

The sub-word parallel NoC designs can match the area-delay product of purely bit-parallel NoCs and in some cases offer superior results (lower area-delay) per bit resulting in higher efficiency for transporting NoC traffic. A sub-word design introduced in Section III-E still retains the bit-serial transmission of valid and address, but adds extra wires to amortize DOR decision costs. This comparison assumes the sub-word design route the identical packet length of 128 bits (valid + address + payload). However, this can be enlarged trivially with an extra LUT (× 4 for valid + data on two NoC directions) for delay balancing for every 32b increase in packet size. We see a saturation of area-delay product for sub-word designs above 5b around 5–6 LUT·ns/bit. We also observe that the 7b design almost matches the efficiency of 64b bit-parallel NoC while a 128b bit-parallel NoC is only competitive below 4b sub-word solutions. Purely bit-serial design run fast, and require very few resources, but do not offer competitive area-delay results over bit-parallel NoCs. The various sub-word parallel designs require anywhere from 15–43 LUTs depending on the sub-word width and require around 0.9–1.2ns clock periods.

VI. Conclusions

We show how to design high-speed bit-serial NoCs on modern FPGAs that operate close-to-GHz rates by (1) LUT-level optimizations and pruning of design logic to minimize logic depth, and (2) compact floorplanning and flexible pipelining of the inter-router links. Using our methodology, we are able to design and deliver a lightweight 20 LUT, 17 FF NoC router that runs at a 1.01–1.04 GHz. Our layout tools are able to map NoCs of various sizes from 2×2 to 16×16 targeting the VC707 board (XC7VX485T FPGA) while requiring less than 2% of the LUT resources in the largest case and never occupying more than 0.3% of the FPGA routing resources. With suitable pipelining, the chip-spanning NoCs typically run at 800–900 MHz while relying on Vivado's placement engines to auto-place the registers for best performance. We expect bit-serial NoCs to be used in routing-constrained designs, or FPGA applications requiring distribution of infrequent control across the chip fabric at low cost, or to support low-cost in-system debugging infrastructure. This bit-serial design is part of the range of Hoplite variants such as Hoplite-DSP [2], and Hierarchical Hoplite [3].

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] 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.

[3] K. H. B. Chethan, A. Shubham, and N. Kapre. Deflection Routing for Multi-Level FPGA Overlay NoCs. In *Field-Programmable Technology (FPT), 2016 International Conference on*, pages 1–8, Dec 2016.

[4] F. Eslami and S. J. E. Wilton. An adaptive virtual overlay for fast trigger insertion for fpga debug. In *Field Programmable Technology (FPT), 2015 International Conference on*, pages 32–39, Dec 2015.

[5] J. Gray. GRVI Phalanx: A massively parallel RISC-V FPGA accelerator accelerator. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 17–20, May 2016.

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

[7] N. Kapre 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] A. Landy and G. Stitt. Revisiting serial arithmetic: A performance and tradeoff analysis for parallel applications on modern FPGAs. In *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pages 9–16, May 2015.

[9] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 196–207, New York, NY, USA, 2009. ACM.

[10] 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.

[11] Xilinx Inc. Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics, February 2015.