Networks-on-Chip for FPGAs: Hard, Soft or Mixed?

Mohamed S. Abdelfattah, University of Toronto
Vaughn Betz, University of Toronto

As FPGA capacity increases, a growing challenge is connecting ever-more components with the current low-level FPGA interconnect, while keeping designers productive and on-chip communication efficient. We propose augmenting FPGAs with networks-on-chip (NoCs) to simplify design, and we show that this can be done while maintaining or even improving silicon efficiency. We compare the area and speed efficiency of each NoC component when implemented hard vs. soft to explore the space and inform our design choices. We then build on this component-level analysis to architect hard NoCs and integrate them into the FPGA fabric; these NoCs are on average 20-23× smaller and 5-6× faster than soft NoCs. A 64-node hard NoC uses only ~2% of an FPGA's silicon area and metallization. We introduce a new communication efficiency metric: silicon area required per realized communication bandwidth. Soft NoCs consume 4960 mm^2/TBps, but hard NoCs are 84× more efficient at 59 mm^2/TBps. Informed design can further reduce the area overhead of NoCs to 23 mm^2/TBps, which is only 2.6× less efficient than the simplest point-to-point soft links (9 mm^2/TBps). Despite this almost comparable efficiency, NoCs can switch data across the entire FPGA while point-to-point links are very limited in capability; therefore, hard NoCs are expected to improve FPGA efficiency for more complex styles of communication.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: field-programmable gate-array, network-on-chip, application-specific integrated circuit, interconnect, area, delay, resource management, hard, soft

ACM Reference Format:
Mohamed S. Abdelfattah and Vaughn Betz, 2014. Networks-on-Chip for FPGAs: Hard, Soft or Mixed?. *ACM Trans. Reconfig. Technol. Syst.* 88, 88, Article 88 (February 2014), 22 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Programmable interconnect is key to FPGAs, but recently it has faced many challenges. First, increasing wire resistance and decreasing pass transistor performance in advanced process nodes lead to poor interconnect delay scaling [Ho et al. 2001]. Second, the high interconnect delay makes it difficult to predict critical paths from a functional (e.g. RTL) description, leading to time-consuming compile/analyze/re-pipeline iterations and increased development time. Third, high-speed I/O interfaces, such as DDRx and PCIe, produce high bandwidth data streams that require wide and fast datapaths within the FPGA fabric. This consumes large amounts of interconnect and presents a further timing closure challenge. Fourth, many individual switches must be programmed to create each link in a design, leading to a large CAD problem and long

This work is supported by NSERC and Altera.
Author's addresses: University of Toronto, Electrical and computer Engineering, 10 King's College Road, Toronto, ON, CANADA M5S3G4.

DOI: http://dx.doi.org/10.1145/0000000.0000000

compilation times. Finally, the low level of interconnect abstraction is a barrier to dividing a design into modules for independent optimization and compilation, or partial reconfiguration. The independently compiled modules must still be linked by physical wires, and timing paths between modules must meet their timing constraints.

Using a higher-level protocol for some of the communication within an FPGA, such as that provided by networks-on-chip (NoC), can address many of these problems. With an NoC, fixed wiring between communicating modules is replaced by a network that routes packets to and from those modules. This simultaneously improves wire utilization and raises the level of abstraction, which facilitates modular design styles [Dally and Towles 2001]. Use of an NoC within an FPGA does force a change in design style as the exact latency of a packet is usually not fixed, and hence the network links must be coded in a latency-tolerant manner. This design style has a side benefit; it simplifies timing closure because interconnect delay no longer affects the cycle time but instead affects the number of cycles a packet takes to reach its destination. NoCs can also simplify partial reconfiguration. When swapping modules, the newly configured module will only have to connect to an NoC interface to communicate to any part of the FPGA instead of having to route each of its nets in an already-functioning FPGA.

There are additional advantages to using a *hard* NoC on the FPGA. A hard NoC will not require configuration onto the soft fabric, making compilation simpler and faster. In addition, the modules communicating via an NoC are timing-disjoint which allows their independent synthesis, placement, routing, and timing closure; these tasks can therefore be performed in parallel, possibly by multiple designers. Communication bandwidth requirements can then be used to determine the optimum position in the network for each module. Hard interfaces on the FPGA such as DDRx, PCIe and gigabit Ethernet operate at high clock frequencies and require low latency, high bandwidth communication to various parts of the chip. A high-performance hard NoC is a good match to these interfaces as it can distribute data throughout the chip at similarly high rates without an excessive number of wires. Of course, a hard NoC also has area and delay advantages over a soft NoC, but presents additional challenges in terms of how it can be integrated within the FPGA fabric; we explore these questions in detail in this work[1].

There is prior work both on FPGA-ASIC efficiency comparison and on FPGA-based NoCs. A comparison of FPGAs and ASICs by [Kuon and Rose 2007] is based on a set of benchmarks with different logic/memory/multiplier ratios. We perform a narrower but more detailed comparison based on a high performance NoC router. [Lee and Shannon 2010] explore how topology parameters impact the frequency of a soft NoC. LiPar [Sethuraman et al. 2005], NoCem [Schelle and Grunwald 2008] and CONNECT [Papamichael and Hoe 2012] are three virtual channel (VC) NoCs implemented efficiently in soft logic on FPGAs. While architectural decisions for soft NoCs were based on FPGA utilization in prior work, we give recommendations based on FPGA silicon area. There has also been interest in a hard NoC on an FPGA. [Francis and Moore 2008] suggest that a circuit-switched network with time-division multiplexed links should be hardened on the FPGA. [Goossens et al. 2008] propose use of a hard-wired NoC for both functional communication and FPGA configuration programming. [Chung et al. 2011] present a programming model that abstracts the distribution of

[1]An earlier version of this work appeared in [Abdelfattah and Betz 2012] but focused only on NoC router component-level results. We extend that work to complete systems in several major ways. First, we analyze not only the router, but also the NoC links and the fabric port (see Section 2). Second, we propose two hard NoC architectures and perform a thorough analysis of their area and delay, including a new study on their impact on metal (wiring) resources. Finally, we introduce a figure-of-merit that expresses the area overhead of communication per supported bandwidth, and use it to compare the efficiency of different NoCs and traditional FPGA interconnect.

data from external memory throughout the FPGA and mention that their application could benefit from a hard NoC. The literature has shown both the implementation of soft NoCs and the desire for hard NoCs on FPGAs but there has not been a detailed comparison of soft (FPGA) and hard (ASIC) NoCs to date. We make that comparison on the NoC component level and use this data to inform our investigation of complete NoC systems. Our contributions include:

(1) Quantifying the area and performance gap of NoC components on FPGAs and ASICs.
(2) Investigating how these area and performance results impact NoC design decisions.
(3) Presenting two novel hard NoC architectures for FPGAs, and detailing how each can be integrated into the FPGA fabric. One uses hard links between routers and the other uses soft links.
(4) Introducing the area-per-bandwidth figure-of-merit and using it to compare the efficiency of complete NoCs implemented on FPGAs vs. conventional soft point-to-point links.

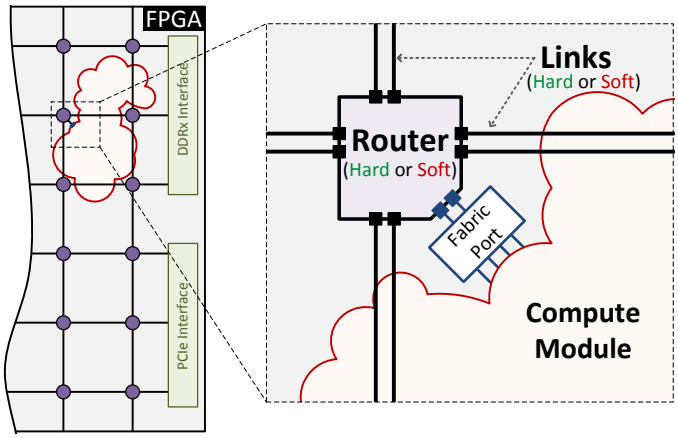

Fig. 1: A mesh NoC implemented on an FPGA.

2. NOC MICROARCHITECTURE

Fig. 1 shows an NoC implemented on an FPGA, used to connect both compute and I/O modules. This includes soft logic constructed from the FPGA fabric, hard logic such as DSP or memory blocks and I/O interfaces such as DDRx and PCIe. There are three main components in the NoC as shown. Routers are used to dynamically switch data from one port to another thus deciding how data moves between two modules on the FPGA. Links carry the data between routers, and the fabric port contains logic (such as clock-crossing and data width adaptation) necessary to interface an FPGA soft module to the NoC. As labeled on the figure, each of these three components can either be implemented in soft logic constructed from the FPGA fabric, hard logic embedded as an unchangeable block or a mixture of both hard and soft. We study the efficiency tradeoff for these implementation options per NoC component in Section 4 and for complete systems in Section 5. This introductory section is used to detail the microarchitecture of each of the NoC components.

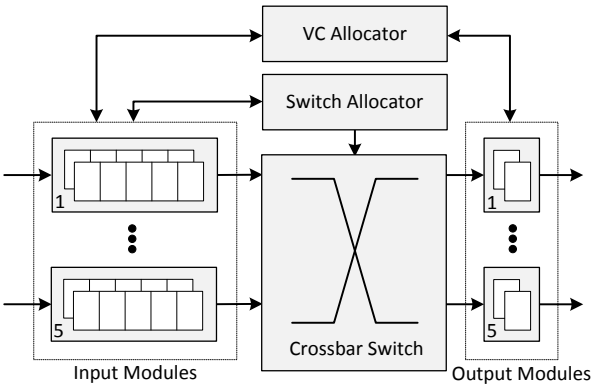

Fig. 2: A virtual channel router with 5 ports, 2 VCs and a 5-word input buffer per VC.

2.1. NoC Routers

We use a parametrized open-source state-of-the-art virtual channel router [D. U. Becker 2012]. We focus on this full-featured router for two reasons. First as FPGAs increase in capacity and hence contain larger applications, it will often be necessary to traverse many network nodes to communicate across the chip, and consequently we expect routers with low latency, such as the chosen router, to be important. As we show in Section 5, this router can also achieve high clock frequencies, helping it keep up with the throughput demands of the high bandwidth I/O interfaces on modern FPGAs. Second, since our focus is quantitatively examining the speed and area of hard and soft implementations of a wide variety of NoC components, use of a more full-featured router with more components yields a more thorough study; simpler routers would use a subset of the components we study. Compared to an FPGA-tuned router such as CONNECT, the state-of-the-art router we use is 140% larger but achieves 60% higher frequency when implemented on an FPGA [Papamichael and Hoe 2012].

The router operates in a 3 stage pipeline that can be reduced to 2 stages if speculation succeeds [D. U. Becker 2012]. Ingress flits are stored in the input buffers and immediately bid for VC allocation; this is followed by switch allocation and switch traversal. Lookahead routing is performed in parallel to switch allocation and routing information is appended to the head flit immediately before traversing the crossbar switch. Finally, flits are registered at the output modules then traverse inter-router links. Fig. 2 shows a block diagram of the router; we detail each component below.

2.1.1. Input Module. The input module buffers incoming flits until routing and resource allocation are complete. Depending on the VC identifier of the packet, it is stored in a different part of the input buffer. Routing information, already computed by the preceding router hop, is decoded and forwarded to the VC and switch allocators to bid for VCs and switching resources. The flit remains in the buffer until both a VC is allocated and the switch is free for traversal, at which point route lookahead information is attached to the head flit and it is ejected from the input module onto the switch. A two-phase routing algorithm, known as Valiant's, routes first to a random intermediate node then to the destination node [Valiant and Brebner 1981]. The algorithm used to route each of the two phases is dimension-ordered routing which routes in each dimension sequentially and deterministically [Dally and Towles 2004].

Dual-ported memory implements the input buffer. Internally, it is organized as a statically allocated multi-queue (SAMQ) buffer which divides the memory into equal

portions for each VC [Tamir and Frazier 1988]. Memory width is always the same as flit width to allow reading and writing flits in one cycle [Balfour and Dally 2006]. In this implementation, the memory buffer has one write port and two read ports to allow the simultaneous loading of a packet's tail flit and the next packet's head flit, thus avoiding a pipeline stall between packets.

Fig. 3a shows a block diagram of the input module used in this study. The VC control units include the route-computation logic and state registers to keep track of the input VC status, the destination output port and the assigned output VC [Balfour and Dally 2006]. The SAMQ controller manages the flit buffer; it selects the read and write addresses depending on the input VC and the granted output VC from switch allocation respectively. A backpressure control unit tracks buffer space availability per VC and transmits credits to upstream router ports on dedicated flow control links.

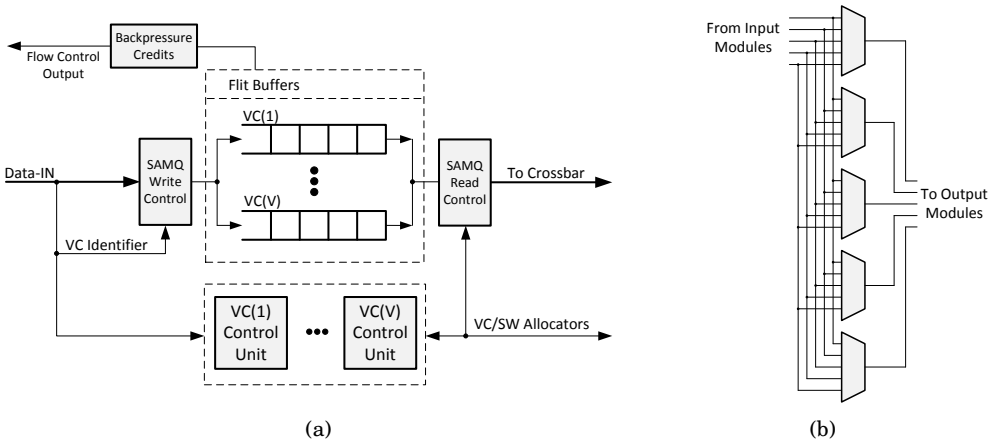

Fig. 3: Router subcomponents: (a) Input module and (b) multiplexer-based crossbar.

2.1.2. Crossbar. We use a multiplexer-based crossbar, as depicted in Fig. 3b, rather than a tri-state buffer design. Modern FPGAs can only implement crossbars with multiplexers, and modern ASIC crossbars are also usually implemented this way. In the best-case scenario, five flits may traverse the crossbar simultaneously if each of them is destined for a different output port. However this rarely occurs, thus requiring queuing of flits in the input module buffers until the crossbar becomes available and the flit can proceed. We found the ASIC crossbar area to be gate limited and not metal limited for width up to 256 bits (with 5 ports) and up to 15 ports (at 32-bit width); other recent work has shown that crossbars as large as 128 ports are also gate limited [Passas et al. 2012].

2.1.3. Switch and VC Allocators. To share the crossbar between router ports, a switch allocator arbitrates between requests in a round-robin fashion. This decides which router input port is given access to a router output port for each cycle. Similarly, a VC allocator decides which incoming packet is assigned each output VC; however, this lasts for the duration of a whole packet. In both cases a separable input-first allocator is used to handle all requests. Previous work has detailed its implementation and operation [Abdelfattah and Betz 2012; Becker and Dally 2009].

2.1.4. Output Module. The crossbar output can be connected to the outgoing wires and the downstream routers directly. However, to improve clock frequency a pipeline stage is placed at the crossbar outputs [Balfour and Dally 2006]. Furthermore, the output

registers are replicated per VC to buffer an additional flit before proceeding downstream, effectively making the downstream input buffers one flit deeper.

2.2. NoC Links

NoC links are the wires used to connect routers together to form a network. Like other NoC components, the links can either be implemented hard or soft. Soft links use the FPGA's interconnect (wires and programmable multiplexers) to connect routers in the same way logic blocks connect to the FPGA interconnect. This has the advantage of configurability; soft links can construct a custom NoC topology as shown in Fig. 4. Furthermore, soft interconnect that is unused by the NoC is not wasted and can be used to connect other blocks on the FPGA. Hard links refer to dedicated wires between routers that can only be used with the NoC. Although hard links lack configurability and can only construct a fixed topology, they are more area efficient and faster than soft links as they do not use programmable multiplexers.

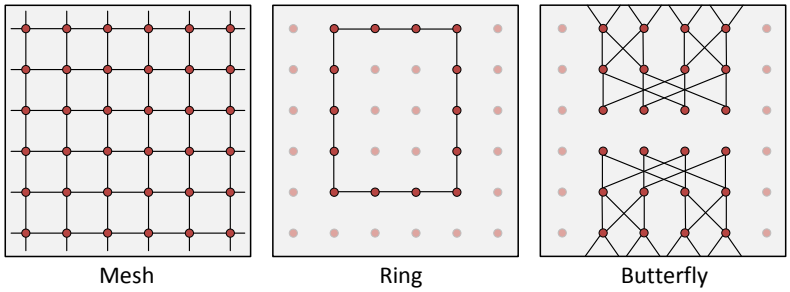

Fig. 4: Examples of different topologies that can be implemented using an NoC that has hard routers and soft links.

2.3. Fabric Port

There is a speed mismatch between the FPGA fabric and an NoC. The FPGA fabric uses multiple relatively slow (~100-400 MHz) clocks [Hutton et al. 2005], while the NoC runs on a single very fast clock (~1 GHz) as we show in Section 5. To use the NoC to efficiently connect FPGA fabric modules running at different speeds, we fix the NoC frequency to its maximum speed (which maximizes bandwidth without increasing area) and use a fabric port to match the fabric bandwidth to the NoC bandwidth. The FPGA fabric achieves high computation bandwidth by using wide datapaths at low speeds, while the NoC is faster and can have a smaller data width. This is why we require both time-domain multiplexing (TDM) logic and a clock crossing FIFO in fabric ports as shown in Fig. 5; we perform both width adaptation and clock crossing. The example in Fig. 5 shows an FPGA module running at 150 MHz with a data width of 128 bits. TDM logic first converts this into 32-bit data width running at $150\,MHz \times 4 = 600\,MHz$, then a dual-port FIFO crosses the clock domain to the 900 MHz NoC clock. The dual-port FIFO is required to maintain the freedom of optimizing the fabric frequencies independently from the NoC frequency; that is, the NoC frequency need not be a multiple of the fabric frequency or vice versa. Note that the TDM factor (4:1) and the clock speeds annotated on Fig. 5 are examples; the TDM factor can be decreased by configuring the counter (to implement 2:1 TDM for example), or increased by augmenting the shown circuit with more soft logic, to implement 8:1 TDM if needed. For router outputs, the same circuit is used with a demultiplexer instead of the multiplexer.

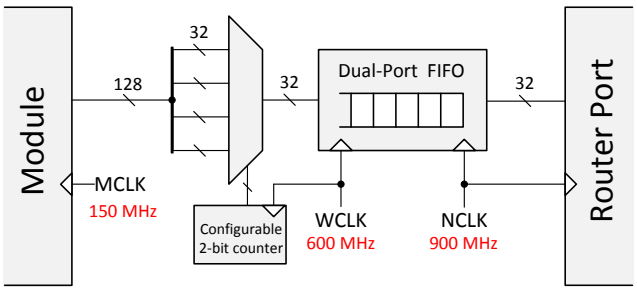

Fig. 5: Fabric port with configurable 4:1 time-domain multiplexing logic. Example frequencies are annotated.

3. METHODOLOGY

The NoC is implemented both on the largest Stratix III FPGA (EP3SL340) and TSMC's 65 nm ASIC process technology. This allows a direct FPGA vs. ASIC comparison since Stratix III devices are manufactured in the same 65 nm TSMC process [Altera Corp. 2007]. Moreover, the area for Stratix III resources is publicly available [Wong et al. 2011]. Table I shows the area, including interconnect, of relevant FPGA blocks.

Table I: FPGA Resource Usage Area [Wong et al. 2011].

Resource	LAB/MLAB	BRAM - 9 kbit	BRAM - 144 kbit	DSP Block	Total Core
Area (mm^2)	0.0221	0.0635	0.5897	0.2623	412

3.1. NoC Routers

A single parameter is varied in each experiment to study the effect of this parameter in isolation. The rest of the parameters are set to the default values for a "baseline" router shown in Table II. One exception is that the buffer depth is varied proportionally when the number of virtual channels is swept.

Table II: Baseline router parameters.

Width	Num. of Ports	Num. of VCs	Buffer Depth
32	5	2	10 (5/VC)

3.1.1. FPGA CAD Flow. Altera Quartus II v11.1 software is used with the highest optimization options to implement the router components. We set an impossible timing constraint of 1 GHz to force the tools to optimize for timing aggressively and report the maximum achievable frequency. Clock jitter and on-die variation are modeled using the "derive_clock_uncertainty" command which applies clock uncertainty constraints based on knowledge of the clock tree [Scoville 2010]. All the circuit I/Os, except the clock, are tied to lookup tables (LUT) using the "virtual pin" option. This mimics the actual placement of an NoC router, and avoids any placement, routing or timing analysis bias that could result from using actual FPGA I/O pins.

Resource utilization is used to calculate the occupied FPGA silicon area by multiplying the used resource count by its physical area in Table I. Simply counting the used logic array blocks (LABs) or adaptive logic modules (ALMs) can overestimate the area

required, as many LABs and ALMs are only partially occupied, and could accept more logic in a very full design. Instead, we use the post-routing area utilization from the resource section in the fitter report which accounts for the porosity in packing, thereby giving a realistic area estimate for a highly utilized FPGA.

The fastest FPGA speed grade corresponds to typical transistors, whereas the slowest FPGA speed grade matches the worst-case transistors of a process. For a fair comparison, we use the fastest FPGA speed grade and the typical transistor model for the ASIC tools. For purely combinational modules, such as the crossbar, registers are placed on the inputs and outputs to force timing analysis. Maximum delay is extracted from the timing reports using the most pessimistic (slow, 85 ^{o}C) timing model, assuming a 1.1 V power supply.

3.1.2. ASIC CAD Flow. Synopsys Design Compiler vF-2011.09-SP4 is used for synthesis, and area and delay estimation. The general-purpose typical process library is used with standard threshold voltage and 0.9 V supply voltage. Unlike the FPGA CAD flow, timing constraints and timing-related tool optimizations impact the ASIC area dramatically. This is because timing optimizations entail standard cell upsizing and buffer insertion whereas FPGA subcircuits are fixed. For this reason, a two-step compilation procedure, described below, is used to reach a realistic point in the large tradeoff space between area and delay.

We perform compilation using a top-down flow, with "Ultra-effort" optimizations for both area and delay. This turns on all optimization options in the synthesis algorithm and accurately predicts post-layout critical path delay and area [Synopsys Inc. 2010]. All registers are replaced with their scan-enabled equivalent to allow the necessary post-manufacturing testing for ASICs. In modeling the wire delay, a conservative wire model from TSMC is used. Capacitance and resistance per unit length are used together with a fanout-dependent length model to estimate the wire delay.

In step one, we perform an ultra-effort compilation with an impossible 0 ns timing constraint and extract the negative slack of the critical path from the timing report. Area numbers are bloated when trying to satisfy the impossible timing constraints and are discarded from this compilation. The negative slack from step one is used as the target clock period in step two of the ASIC compilation. This provides a reasonable target for the CAD tools and results in realistic cell upsizing, logic duplication and buffer insertion, and hence realistic area numbers. With the clock period adjusted, the design is recompiled and the implementation area and delay are extracted from the synthesis reports. Note that any positive or negative slack in this step is also added to the critical path delay measurement.

Generally ASICs are not routable if they are 100% filled with cells. To account for whitespace, buffers inserted during placement and routing, and wiring, we assume a 60% rule-of-thumb fill factor and so inflate the area results by 66.7%. Fill factors as low as 10% and as high as 90% have been used in the literature [Passas et al. 2012; Kuon and Rose 2007] but we chose 60% to model the typical case after conversations with ASIC design engineers. We do not run place-and-route tools in the ASIC flow because ASIC synthesis tools go to great lengths to produce accurate post-synthesis results [Bhatnagar 2002]. Further, unlike the FPGA flow, place-and-route on ASICs would require a great deal of designer effort for such a large design space exploration.

3.1.3. Methodology Verification. To verify the methodology, we compare the results obtained with our methodology to those of Kuon and Rose on their largest benchmark, raytracer [Kuon and Rose 2007]. As shown in Table III, the area and delay ratios are quite close; we expect some difference as our results are from a 65 nm process while theirs are from 90 nm.

Table III: Raytracer benchmark area and delay ratios.

	[Kuon and Rose 2007]	This Work
FPGA Device	Stratix II	Stratix III
ASIC Technology	90 nm	65 nm
Area Ratio	26.0	25.6
Delay Ratio	3.5	4.1

3.2. NoC Links

3.2.1. Soft (FPGA) Wires. Soft NoC links are implemented using the prefabricated FPGA "soft" interconnect. On Stratix III FPGAs, there are four wire types: vertical length four (C4) and length 12 (C12), and horizontal length four (R4) and length 20 (R20). We connect two registers using a single wire segment and measure the delay of this wire segment. Next, we investigate different connection lengths by connecting wire segments of the same type in series and measuring delay. Registers are manually placed using location constraints to define the wire endpoints, and the connection between the registers is manually routed by specifying exactly which wires are used in a routing constraints file (RCF).

3.2.2. Hard (ASIC) Wires. We use TSMC's 65 nm intermediate-layer metal and simulate lumped element models of ASIC wires to measure the delay of hard NoC links. First, we conduct a series of experiments using HSPICE vF-2011.09.SP1 to design and optimize our ASIC wires. To match the FPGA experiments, the supply voltage is set to 1.1 V and the simulation temperature is 85 ^{o}C. We reach a reasonable design point with metal width and spacing of 0.6 μm, drive strength of 20-80$\times$ that of a minimum-width transistor (depending on total wire length) and rebuffering every 3 mm. Propagation delay is measured for both rising and falling edges of a square pulse signal, and the worst case is taken to represent the speed of this wire.

4. AREA AND SPEED ANALYSIS

In this section we perform a component-level analysis of the area and speed of NoC components; routers (divided into 5 subcomponents), links and fabric ports. We quantify the efficiency gap when implemented hard vs. soft to better understand the two options and to explore the design space of implementing NoCs on FPGAs.

4.1. NoC Routers

Figures 6 and 7 show the FPGA/ASIC area and delay ratios for the router components as they vary with the four main router parameters: flit width (16-256), number of ports (2-15), number of VCs (1-10) and input buffer depth (5-65). These results are summarized in Table IV in which the minimum, maximum and geometric mean is given for each component. We choose a realistic range of router parameters, based on a study of the literature, such that the geometric average of the area or delay ratio is indicative of the FPGA-to-ASIC gap for an NoC that is likely to be constructed. On average, NoC routers use 30$\times$ less area and run 6$\times$ faster[2] when embedded in hard logic compared to a soft implementation (see Table IV).

4.1.1. Input Module. The input module consists of a memory buffer and mixed logic for routing and control. To synthesize an efficient FPGA implementation, the memory

[2]Note that the speed factor is different from our previous work [Abdelfattah and Betz 2012] as we have improved the methodology for its calculation. Instead of stitching together the critical path from router components (previous work), we implement the whole router and then perform timing analysis.

Table IV: Summary of FPGA/ASIC (soft/hard) router area and delay ratios.

Router Component	Area			Delay		
	Min.	Max.	**Geomean**	Min.	Max.	**Geomean**
Input Module	8	36	**17**	2.2	4.0	**2.9**
Crossbar	57	169	**85**	3.3	6.9	**4.4**
VC Allocator	27	76	**48**	2.0	4.8	**3.9**
Switch Allocator	24	94	**56**	1.9	4.2	**3.3**
Output Module	30	47	**39**	3.1	3.7	**3.4**
Router	13	64	**30**	4.7	8.0	**6.0**

buffer is modified to target the three variants of RAM on FPGAs: registers, LUTRAM[3] and block RAM (BRAM). LUTRAM uses FPGA LUTs as small memory buffers and BRAMs are dedicated hard memory blocks that support tens of kilobits. Both LUTRAM and BRAM can implement dual-ported memories (1r1w) and the second read port (2r1w) required for the input module (see Section 2.1.1) is added by replicating the RAM module. Registers are more flexible and can construct multiple read ports by replicating the read port itself and not the storage registers. On ASICs, the memory buffer is always implemented using a 2D flip-flop array which is the norm for building small memories. Fig. 8 shows the FPGA area of various buffers when implemented using the three alternatives mentioned. The minimum-area implementation is selected for the comparison against ASICs. In all the data points when varying the buffer depth, the BRAM-based implementation has the lowest area. In fact, the area remains constant since the 9-kbit BRAM can handle up to 256 memory words. LUTRAM is slightly less efficient than BRAM with shallow buffers, but the area increases rapidly whenever another LUTRAM is used to increase buffer depth. BRAMs and LUTRAMs have width limitations but can be grouped together to implement wider memories. This explains the linear area increase with width shown in Fig. 8.

The bit density for a register-based memory buffer is 0.77 kbit/mm^2 compared to 23 kbit/mm^2 for a LUTRAM and 142 kbit/mm^2 for a 9-kbit BRAM [Wong et al. 2011]. This means that a 9-kbit BRAM with only 16% of its bits used is just as area-efficient as a fully utilized LUTRAM on a Stratix III FPGA, explaining the lower BRAM area with very shallow buffers. Although prior work has gravitated towards the use of LUTRAM [Papamichael and Hoe 2012], when looking at it from a silicon perspective, the high density of BRAM makes it more area-efficient for most width$\times$depth combinations. However, in architectures with deeper BRAM, such as Virtex 7, LUTRAM may be the more efficient alternative for shallow buffers.

As Table IV shows, the input module has the lowest area and delay gaps of the presented components. The area gap varies from 8-36$\times$ with a geometric mean of 17$\times$. The lower gap occurs when a deep buffer is used and the FPGA BRAM is well-utilized. Width is found to have only a small effect on the input module area and delay ratios, but varying the number of VCs presents a more interesting result. The input module consists of both control logic, which is inefficient on FPGAs, and memory buffers implemented as compact hard blocks. As we vary the number of VCs in Fig. 6 the FPGA implementation becomes twice as efficient between 1 and 6 VCs because we are able to pack more buffer space into the same BRAM module. However, as we increase the number of VCs further, the efficiency drops because the control logic for a large num-

[3]Stratix IV was used for LUTRAM experiments to avoid a bug in Stratix III LUTRAM.

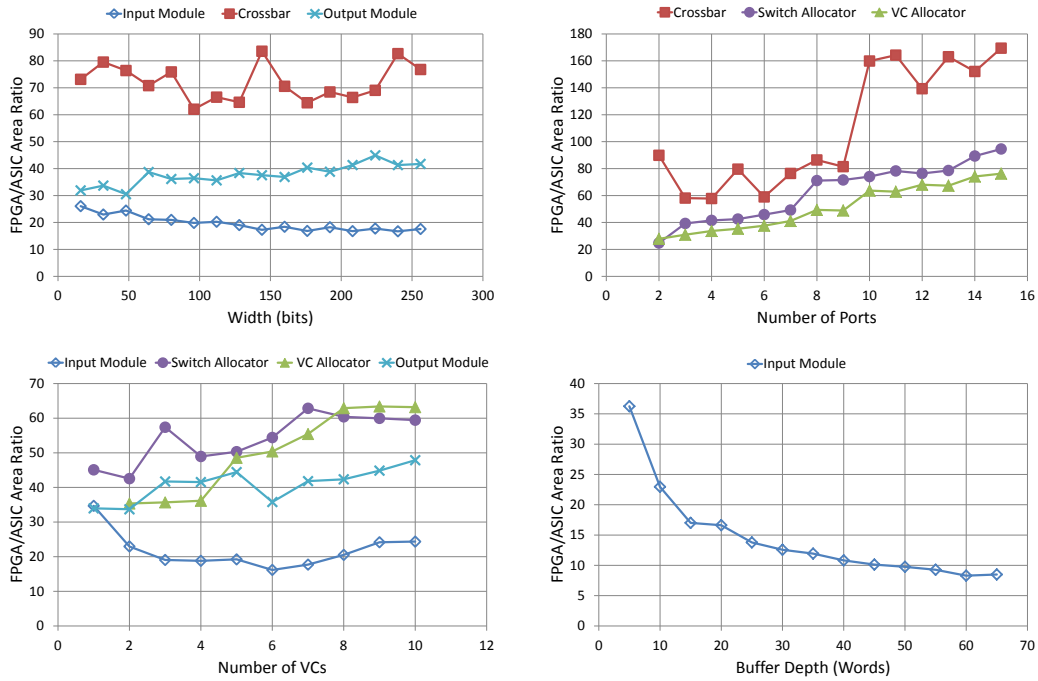

Fig. 6: FPGA/ASIC (soft/hard) area ratios as a function of key router parameters.

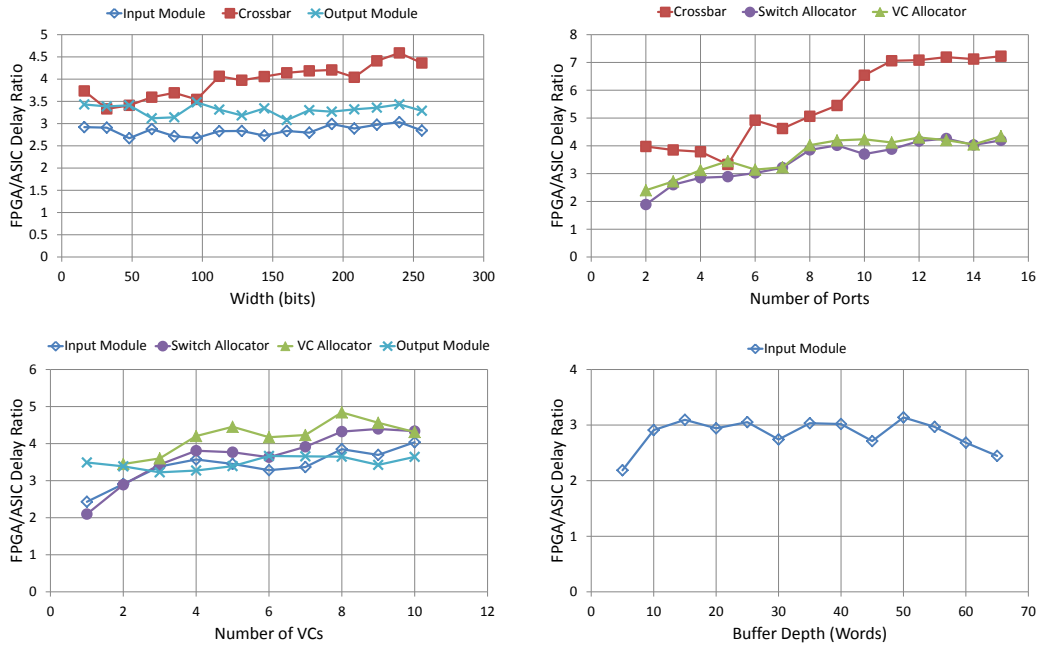

Fig. 7: FPGA/ASIC (soft/hard) delay ratios as a function of key router parameters.

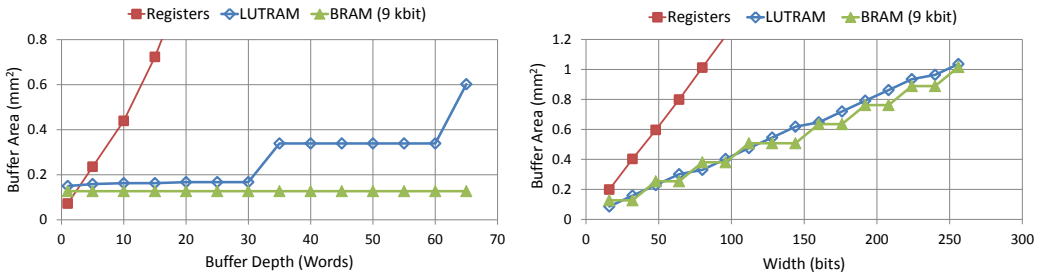

Fig. 8: FPGA silicon area of memory buffers implemented using three alternatives.

ber of VCs becomes the dominant area component. The FPGA-to-ASIC delay ratio is 2.9× and is always limited by the logic component of the input module and not the fast BRAM memory component.

4.1.2. Crossbar. Crossbars show the largest area gap; a minimum of 57×, a maximum of 169× and a geometric average of 85×. It is worth noting that there is a 2× FPGA efficiency loss for crossbars with 10 or more ports. This is due to two causes. First, the required FPGA LUTs per multiplexer port increases faster than the ASIC gates per port. Second, there is an increased demand for interconnect ports at the logic module inputs causing LUTs in that logic module to be unusable; the ratio of LUTs that are unused for this reason grows from 1% to 10% between 9 and 10 multiplexer ports. When the width is varied however we see very little variation between a 16-bit and a 256-bit wide crossbar; the variations in the width plot are due to better or worse mapping of different-size multiplexers onto the FPGA's LUTs.

The crossbar delay gap grows significantly from 3.3-6.9× with increasing port count. This trend is due to the increase in FPGA area, which causes the multiplexers to be fragmented over multiple logic modules thus extending the critical path. Overall, the average delay gap is 4.4× for the crossbar; the largest out of all the components.

The results show that crossbars are inefficient on FPGAs and their scaling behavior is also much worse than ASICs. This is a prime example of a circuit that would bring area and delay advantages if it were hardened on the FPGA. In this scenario, the crossbar can be overprovisioned with a large number of ports so that it can support different NoC organizations. If a small number of router ports are required but a large number of ports are available, the additional ports can be used towards crossbar speedup which relieves crossbar traffic by allowing multiple VCs from the same input port to traverse the switch simultaneously. This also simplifies switch allocation [Dally and Towles 2004].

4.1.3. VC and Switch Allocators. Allocators are built out of arbiters which consist of combinational logic and some registers. Ideally the ratio of LUTs to registers should match the FPGA architecture; for Stratix III a 1:1 ratio would use the resources most efficiently. Deviation from this ratio means that some logic blocks will have either registers or LUTs used but not both. The unused part of the logic block is area overhead when compared to ASICs.

Although there are other sources of FPGA inefficiencies, there is a direct correlation between the LUT-to-register ratio and the FPGA-to-ASIC area gap. For the VC allocator the average LUT-to-register ratio is 8:1 and the area gap is 48×, while the speculative switch allocator has an average LUT-to-register ratio of 20:1 and the area gap is higher; approximately 56×. This difference between the two allocators is due to

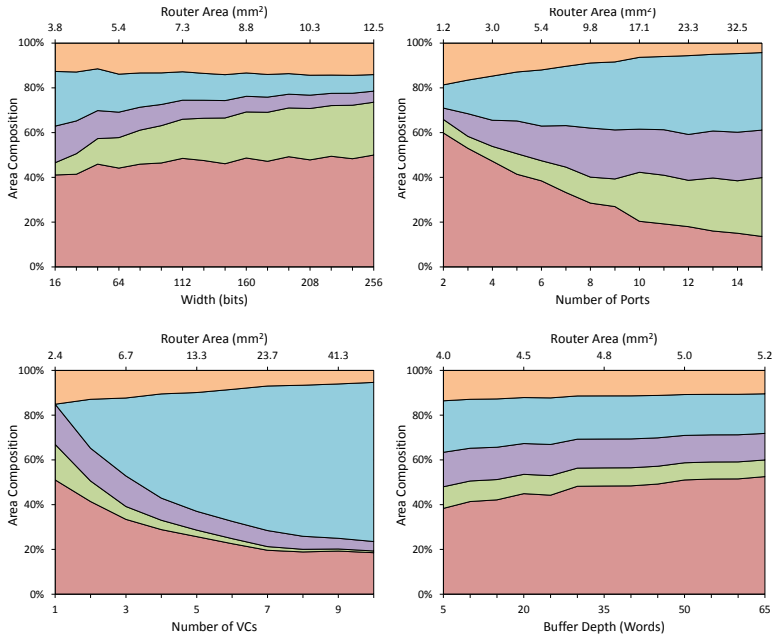

Fig. 9: FPGA (soft) router area composition by component. Starting from the bottom(red): Input module, crossbar, switch allocator, VC allocator and output module.

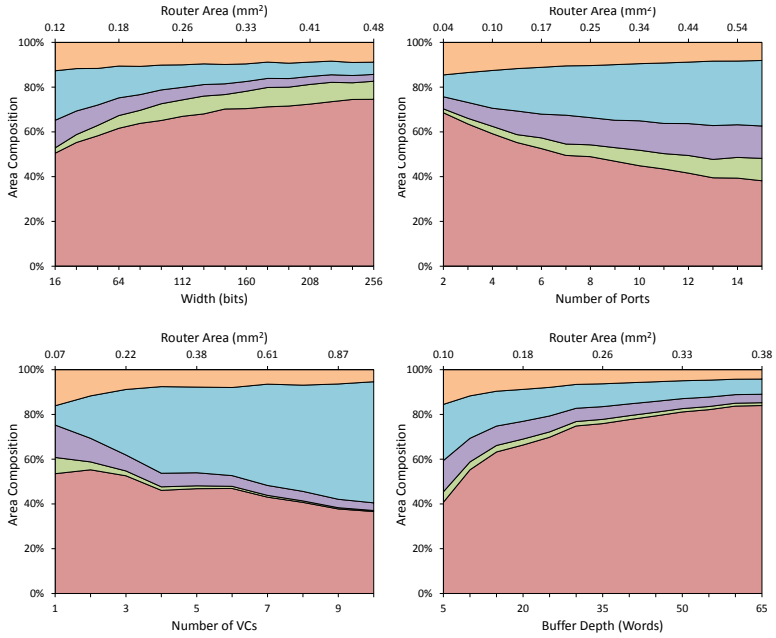

Fig. 10: ASIC (hard) router area composition by component. Starting from the bottom(red): Input module, crossbar, switch allocator, VC allocator and output module.

the selection logic which is used in the speculative switch allocator and absent from the VC allocator.

Allocator delay increases with circuit size for both hard and soft implementations; however, the delay rises more rapidly for the soft version. Consequently, the delay ratio of the allocators is proportional to the circuit size, and grows with increasing port or VC count. We suspect this is because the fixed FPGA fabric restricts the placement and routing optimizations that can be performed on large circuits while ASIC flows have more options, such as upsizing cells or wires on critical paths. Overall, the delay gap is around 3.6× for the allocators.

4.1.4. Output Module. The output module is the smallest router component and is dominated by the output registers. Indeed, the LUT-to-register ratio is 0.6:1 contributing to its smaller area gap of 39× when compared to the allocators. The average delay ratio of 3.4× is also relatively low because the simple circuitry does not stress the FPGA interconnect.

4.1.5. Router Area Composition on FPGA and ASIC. Figures 9 and 10 show the router area composition on FPGAs and ASICs respectively. Moreover, the total router area of select data points is given on the top axes.

The main discrepancy between the FPGA and ASIC router composition is the proportion of the input modules and the crossbar. The input modules are the largest components for most router variants on both the soft and hard implementations. It follows from the area ratios that the input modules are relatively larger on ASICs than on FPGAs; in fact, they occupy 36-83% of the ASIC router area compared to 14-60% on the FPGA. The crossbar is the smallest component of an ASIC VC router. On FPGAs, however, it becomes a critical component with a wide datapath or a large number of ports where it occupies up to 26% of the area.

With an increasing number of VCs, the VC allocator area dominates on both FPGAs and ASICs. Increasing the number of ports also increases the VC allocator area but to a lesser extent. This is due to the second stage of VC allocation which occupies most of the area and is constructed out of $P \times V{:}1$ arbiters. They require an additional P inputs per arbiter when the number of VCs is increased whereas only V additional inputs are required when the number of ports is raised. Since P is larger than V for the baseline router, the VC allocator's area grows more slowly with the number of ports than it does with the number of VCs. The speculative switch allocator also grows with increasing port and VC counts but is more affected by the number of ports. With 15 router ports, the switch allocator makes up 22% of the FPGA router area.

4.2. NoC Links

4.2.1. Speed. Fig. 11 shows the speed of hard and soft wires. Soft wires connect to multiplexers which increase their capacitive loading and add switch delay, making them slower. However, these multiplexers allow the soft interconnect to create different topologies between routers, and enables the reuse of the metal resources by other FPGA logic when unused by the NoC. We lose this reconfigurability with hard wires but they are, on average, 2.4× faster than soft wires. A detailed look at the different soft wires shows that long wires (C12, R20) are faster, per mm, than short wires (C4, R4). An important metric is the distance that we can traverse between routers while maintaining the maximum possible NoC frequency. This determines how far we can space out NoC routers without compromising speed. In the case of soft links and a soft (programmable) clock network, the clock frequency on Stratix III is limited to 730 MHz. At this frequency, short wires can traverse 2.5 mm while longer wires can traverse 5 mm of chip length between routers. When using hard links, we are only

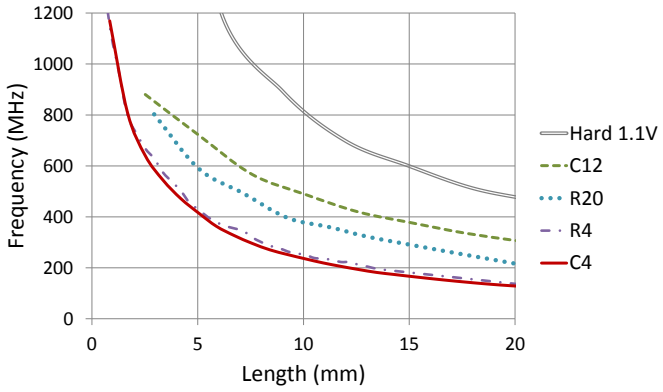

Fig. 11: Hard and soft interconnect wires frequency.

limited by the routers' maximum frequency, which is approximately 900 MHz. At this frequency, hard links can traverse 9 mm.

4.2.2. Area. To find the transistor area overhead of soft wires, we look at the area required to implement programmable multiplexers at LAB inputs and outputs and assume that the type of multiplexers and interconnection flexibility used to connect routers to the programmable (soft) interconnect matches that of a LAB. 50% of a Stratix III LAB area (0.011 mm^2) consists of programmable interconnect and supplies 52 input ports and 40 output ports [Lewis et al. 2013]. This means that each input or output occupies $\frac{0.011\ mm^2}{92} = 120\ \mu m^2$ of silicon area.

For hard wires, the area overhead is lower as only CMOS drivers are required at wire ends, and no multiplexers are needed. We use the exact transistor layout parameters from TSMC to hand layout the CMOS drivers and measure total transistor area. The drivers used with 3 mm wires are 20$\times$ the minimum transistor width (=2.4 μm) which makes the total area of one CMOS driver 13 μm^2. Therefore hard wires are 9$\times$ more area efficient than soft wires, per router port, in terms of silicon area.

FPGAs use metal very heavily for interconnect; therefore it is a valuable and scarce resource, much like silicon, and must be studied to better understand the overhead of NoCs. A reasonable approximation is that both hard and soft wires utilize metal equally for links that are the same length. We derive the metal area and interconnect stress for complete NoCs in our system-level discussion in Section 5.

4.3. Fabric Port

We generated a library of fabric ports with 2:1, 4:1 and 8:1 TDM factors, with widths ranging from 16-256 bits on the router side. After taking the geometric average over all generated circuits, they were found to be 23$\times$ smaller and 3.3$\times$ faster when implemented hard compared to soft. Fabric ports are especially necessary when there is a speed gap between the compute module and the NoC; it serves to bridge the two components in that case. This also means that fabric ports are only relevant when the NoC routers are implemented in hard logic, and can therefore run considerably faster than the FPGA fabric (~900MHz). For that reason, only hard fabric ports may be used with hard routers; they can run at ~1.5 GHz compared to ~550 MHz for soft fabric ports.

5. FPGA NOC SYSTEMS

In this section we combine component-level results from Section 4 to investigate complete NoC systems that are hard, soft or a mixture of hard and soft. We then calculate

the area-per-bandwidth to compare NoCs against the simplest form of FPGA communication: point-to-point links created from soft wires.

5.1. Soft NoCs

Soft NoCs require no architectural changes to the FPGA because they are configured out of the existing FPGA fabric. As such, the strengths of a soft NoC lie in its reconfigurability. Based on the results in Section 4, we summarize the design recommendations for a soft NoC that makes efficient use of the FPGA's silicon area:

(1) BRAM was most efficient for memory buffer implementation even for shallow buffers, so buffer depth is free until the BRAM is full.
(2) To increase bandwidth, it is more efficient to increase the flit width rather than the number of ports or VCs.
(3) The number of ports and number of VCs scale poorly on FPGAs because of the quadratic increase of allocator and crossbar area.

However, because of their high area overhead and meager operating frequency, soft NoCs are unlikely to replace current interconnect structures, such as buses or point-to-point links. This is especially true for high bandwidth and streaming applications where both throughput and latency are a concern. We now look at the gains of hardening NoC components. One viable option is to harden the crossbar and allocators and leave the input and output modules soft. This solution moves the critical path from the switch allocator to the input module allowing the router to run at 386 MHz compared to 167 MHz for a fully soft implementation. Such a heterogeneous router occupies 2.34 mm^2 for the baseline parameters. The 1.8$\times$ area improvement over a soft implementation is, however, unconvincing. Furthermore we have not yet accounted for the area of the interconnect ports; that is, the switch and connection blocks that would route wires into, out of and around the hard component. For that reason we look more closely into using completely hard routers in building FPGA NoCs.

5.2. Mixed NoCs: Hard Routers and Soft Links

In this NoC architecture, we embed hard routers on the FPGA and connect them via the soft interconnect. While this NoC achieves a major increase in area-efficiency and performance versus a soft NoC, it remains highly configurable by virtue of the soft links. The soft interconnect can connect the routers together in any network topology as shown in Fig. 4, subject only to the limitation that no router can exceed its (prefabricated) port count. To accommodate different NoCs, routing tables inside the router control units are simply reprogrammed by the FPGA CAD tools to match the new topology.

5.2.1. Area and Speed. Fig. 12 shows a detailed illustration of an embedded router connected to the soft interconnect. Note that we must ensure that a sufficient number of interconnect wires intersect the hard router to connect to all of its inputs and outputs. This prevents any interconnection "hot spots" that would over-stress the FPGA's wiring; we aim to have the same interconnection flexibility with NoC routers as we do with LABs. We achieve this by ensuring:

— Connection blocks and switch blocks are only present on the router perimeter. For example the router in Fig. 12 can only connect to 8 connection/switch blocks because 8 LABs are on its perimeter (although its area equals 9 LABs). This makes physical layout much simpler.
— Hard routers do not over-stress the soft interconnect; they cannot have more inputs/outputs per unit of perimeter than regular LABs.

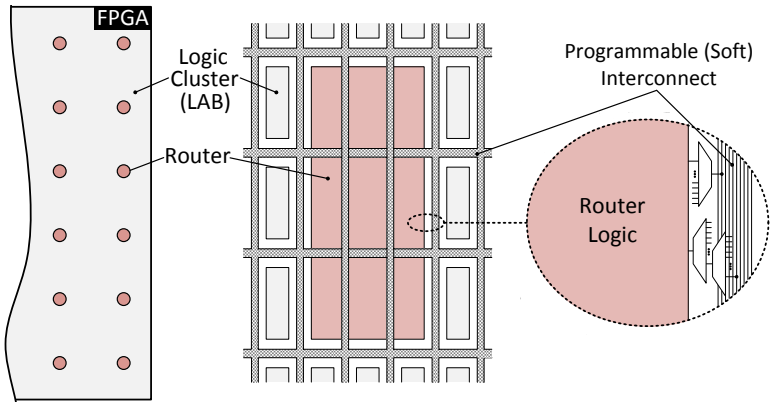

Fig. 12: Floor plan of a hard router with soft links embedded in the FPGA fabric. Drawn to a realistic scale.

— Hard routers have equivalent interconnection flexibility as LABs. This implies using 2 levels of programmable multiplexers on each input and one level of programmable multiplexers between an output and a soft interconnect wire.
— Soft wires continue across hard routers but cannot start or end within the router area, only at its perimeter.

Using the component-level results in Section 4, we compute the area and performance of an NoC with hard routers and soft links. We assume a router with the baseline parameters and one fabric port with 4:1 TDM. Similarly to LABs or block RAMs on the FPGA, a hard router requires programmable multiplexers on each of its inputs and outputs to connect to the soft interconnect in a flexible way. The baseline router has 32 data plus 2 backpressure inputs and outputs per port; furthermore, the fabric port has 128 data plus 2 backpressure inputs and outputs; making the sum of input and output ports 532. Therefore, the total area of the router, the fabric port and the interconnect multiplexers is $0.14\,mm^2 + 0.009\,mm^2 + 532 \times 120\,\mu m^2 = 0.21\,mm^2$, which is equivalent to 9.5 LABs, rounded up to 10 LABs. Assuming that the hard router occupies the area of 5×2 LABs, its perimeter can connect to the equivalent of 10 LABs (or 520 inputs and 400 outputs). This is more than enough to supply the 266 inputs and outputs required by the router, ensuring that the soft interconnect is not stressed; on the contrary, this router has lower interconnect stress than a regular LAB on the FPGA. Note that routers only become input/output pin limited when data width is greater than 220 bits and 4:1 TDM is used. We repeat these area calculations for all of the design space and take the geometric average; there is a 20× area improvement over soft router implementations (as opposed to 30× when we excluded interconnect).

The speed of an NoC with hard routers and soft links is limited by the soft interconnect and the fabric clock network. We choose the FPGA maximum clock network frequency (730 MHz in Stratix III) as the target NoC frequency, and find that short wires can traverse ~2.5 mm at this speed while longer soft wires can traverse ~5 mm (see Fig. 11). The FPGA's core dimensions are ~21 mm in each dimension; therefore, an 8×8 mesh of hard routers using the soft interconnect would allow operation at the maximum frequency of 730 MHz even when using the slower short wires.

5.2.2. FPGA Silicon and Metal Area Budget. To see the cost of a complete system, consider hardening a 64-node NoC on the FPGA. Using baseline parameters, this will occupy area equivalent to $10\,LABs \times 64\,nodes = 640\,LABs$. The total core area of the largest

Stratix III FPGA is 412 mm^2 [Wong et al. 2011]. Therefore, a 64-node hard NoC composed of state-of-the-art VC routers will occupy 3.3% of the core area (~2.2% of total chip area) of this FPGA, compared to 64% of the core area (~43% of total chip area) for a soft implementation.

We now estimate the soft interconnect stress caused by this NoC. Table V compares the demand for soft wires by NoC links to both the total wire supply and the supply of wires in NoC regions. We define NoC regions as the interconnect channels between routers, and in this example we assume the routers are configured in a mesh topology. In this case NoC links can be constructed most efficiently in the "NoC regions" by concatenating interconnect resources in a straight vertical or horizontal path between routers. We evaluate two cases: making the soft links with short C4/R4 wires, or with the longer C12/R20 wires. Our baseline routers have 32 bidirectional links between channels and 4 bits of flow control for a total of 68 bit-links between two routers. If we only use short wires each bit-link is constructed by stitching together 3 C4 wires in the vertical direction or 4 R4 wires in the horizontal direction. We compute the resulting total interconnect utilization in Table V; note that the NoC links require less than 2% of the total C4/R4 interconnect. The interconnect stress is concentrated in the NoC regions but even there is not excessive; Table V shows between 8% and 11% of the C4/R4 interconnect is used in these regions. It is difficult to implement NoC links on long wires exclusively; there are not enough R20 wires in NoC regions as shown in Table V.

Table V: Soft interconnect utilization for a 64-node 32-bit mixed NoC using either C4/R4 or C12/R20 wires on the largest Stratix III device.

		Short Wires		Long Wires	
		C4	R4	C12	R20
Demand	–	14,688	19,584	4,896	4,896
Supply	Regional	166,400	159,936	8,320	4,704
	Chip-wide	639,360	1,074,944	34,336	34,944
Utilization	Regional	7.8%	10.9%	58.9%	104%
	Chip-wide	2.0%	1.6%	14.3%	14.0%

5.3. Hard NoCs: Hard Routers and Hard Links

Both routers and links are implemented hard for this NoC architecture. Routers are connected to other routers using dedicated hard links; however, routers still interface to the FPGA through programmable multiplexers connected to the soft interconnect. When using hard links, the NoC topology is no longer configurable. However, the hard links save area (as they require no multiplexers) and run at higher speeds compared to soft links, allowing the NoC to run at the routers' maximum frequency. Drivers at the ends of dedicated wires charge and discharge data bits onto the hard links as shown in Fig. 13.

5.3.1. Area and Speed. Following the same assumptions itemized in Section 5.2.1, we derive the complete area overhead of a baseline router including its inputs, outputs and the fabric port. There are 272 hard links connecting a router to its neighbours and 260 soft interconnect ports connecting the router fabric port to the FPGA fabric. Therefore, the total area of the router, fabric port, and hard and soft inputs/outputs equals $0.14\,mm^2 + 0.009\,mm^2 + 272 \times 13\,\mu m^2 + 260 \times 120\,\mu m^2 = 0.18\,mm^2$, or 8.3 LABs, rounded up to 9 LABs. We repeat these calculations for all of the design space and take the

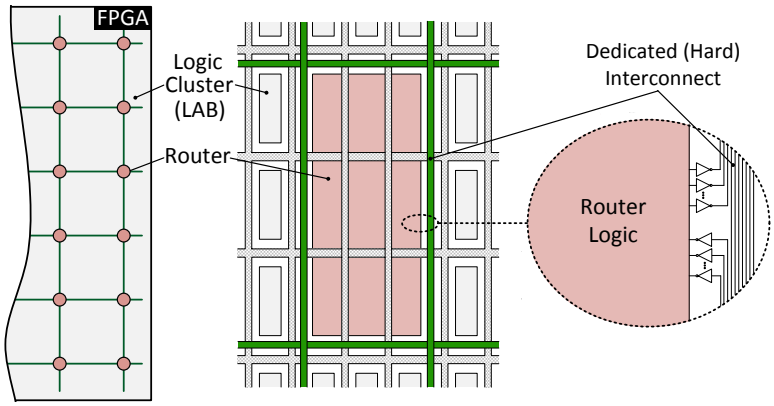

Fig. 13: Floor plan of a hard router with hard links embedded in the FPGA fabric. Drawn to a realistic scale.

geometric average; there is a 23× area improvement over soft router implementations (compared to 20× for hard routers and soft links).

The maximum frequency of the baseline router is 943 MHz. At this frequency, hard wires can reach more than one third of the FPGA's dimension (~8mm) as measured in Fig. 11. Unlike soft links, hard links do not limit the speed of the NoC (to 730 MHz); rather, the routers can operate at their maximum frequencies and can be spaced-out more if desired. This makes this NoC with hard links 20% faster than the hard NoC with soft links, and 6× faster than a completely soft NoC.

5.3.2. FPGA Area Budget. A 64-node NoC of hard routers and hard links occupies silicon area equivalent to $9\,LABs \times 64\,nodes = 576\,LABs$, or 3.1% of the FPGA core area (~2.1% of total chip area). However, we should check not only the transistor area but also the metal utilization of hard links. Each hard wire is 2.5 mm long and has a pitch of 1.2 μm. The 64 routers of this 32-bit NoC require a total of 9792 wires; making the total metal area equal $9792 \times 1.2\,\mu m \times 2.5\,mm = 29.4\,mm^2$. The FPGA core area is 412 mm^2, and this is also the area of each metal layer on top of the FPGA core. If 2 metal layers are used for the NoC, then the utilization of each metal layer is only 3.6% for all 9792 wires used in a hard NoC.

5.4. Comparing NoCs and FPGA Interconnect

We suggest the use of NoCs to implement global communication on the FPGA; as such, we must compare to existing methods. There are two main types of communication that can be configured on the FPGA as shown in Fig. 14. The first uses only soft wires to implement a direct point-to-point connection between modules or to broadcast signals to multiple compute modules. The second type of communication uses wires, multiplexers and arbiters to construct logical buses. This is often used to connect multiple masters to a single slave, for example connecting multiple compute modules to external memory. Although the proposed NoCs can implement both of these communication requirements (point-to-point and arbitration), we compare our NoC area with the simplest alternative: point-to-point links that are equal in length to a single NoC link between two routers. This simplest form of communication serves as a *lower bound* of any communication overhead.

As a generalization of the area-delay product, we compute the area overhead of NoCs (or other communication methods) per supported data bandwidth. This figure of merit quantifies the area cost of each Terabyte-per-second of data bandwidth on different

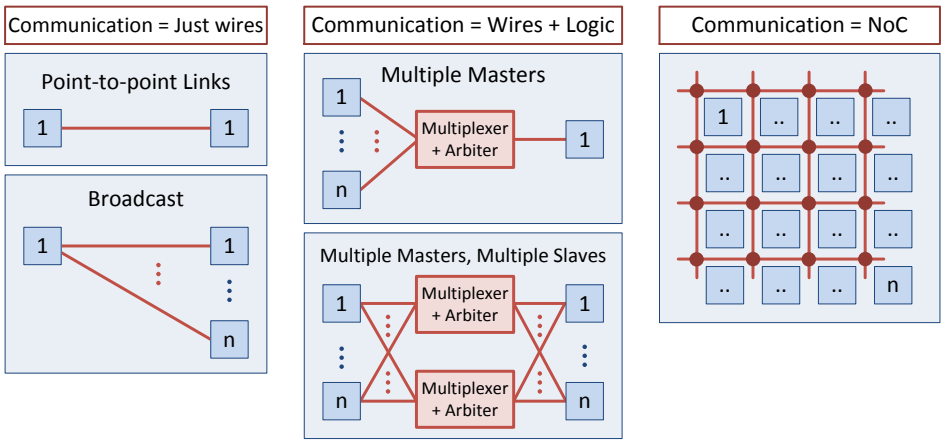

Fig. 14: Different types of on-chip communication.

communication architectures. The aggregate bandwidth of point-to-point links is simply the product of the link width and its speed. To find the aggregate bandwidth of NoCs we perform a cycle-accurate simulation of NoC routers using ModelSim and attempt to inject packets randomly on each cycle at each port; this represents worst-case uniform-random traffic. Naturally the router stalls due to switch contention and limited buffer space thus limiting bandwidth. We measure this steady-state worst-case bandwidth and report it for different NoC variants in Table VI.

5.4.1. Point-to-point Links. We compute a *lower bound* of the area required for communicating data on a conventional FPGA by analyzing the simplest form of communication: point-to-point links using the soft interconnect. To move 250 GB/s of data, one could use 10,000 soft wires running at a reasonable 200 MHz clock frequency. Each unit of data must be transmitted 2.5 mm, the length of one NoC link, which requires stitching four R4 soft wires together. The silicon area overhead of these wires comprises the soft interconnect multiplexers in the switch blocks and logic block inputs. For 10,000 bit-links that consist of 4 R4 wires each, we require at least 200 switch blocks ($\frac{10{,}000\,bits \times 4\,wires\,long}{200\,wires\,switch\,capacity}$), based on our estimate that each switch block has an achievable routing capacity of 200 signals. We then need to connect 10,000 wires to LABs; each of which has 52 inputs. Therefore the input multiplexers of 193 LABs are also taken into account as area overhead. 50% of a LAB's area is interconnect with 30% being associated with input multiplexers and another 20% in the switch blocks and outputs [Lewis et al. 2013]. Using this information we can then estimate the total soft interconnect area as 2.2 mm^2. This translates into 8.8 mm^2 of silicon area to support 1 TB/s of data bandwidth.

5.4.2. Hard and Soft NoCs. A completely soft NoC can be configured onto the FPGA fabric without any architectural changes but a 64-node soft NoC consumes about half the area of an FPGA. Furthermore, it has a low aggregate bandwidth owing to its modest clock frequency as shown in Table VI. This leads to the prohibitively high area-per-BW of 4960 mm^2/TBps. Next, we look at hard NoCs. A hard NoC with soft links is limited to the maximum speed of the FPGA interconnect; nevertheless, this is enough to push this NoC's aggregate bandwidth to 238 GB/s. The total area of this NoC is also greatly reduced compared to soft NoCs (~20×) making its area-per-BW 84× lower than soft NoCs, or 59.4 mm^2/TBps. With hard routers and hard links the NoC can run

Table VI: System-level area-per-bandwidth comparison of different FPGA-based NoCs.

FPGA-based NoCs					
Routers	Links	Description	Area	Bandwidth	Area per BW
Soft 64-NoC	Soft	167 MHz, 32 bits, 2 VCs	269 mm^2	54.4 GB/s	4960 mm^2/TBps
Hard 64-NoC	Soft	730 MHz, 32 bits, 2 VCs	14.1 mm^2	238 GB/s	59.4 mm^2/TBps
Hard 64-NoC	Hard	943 MHz, 32 bits, 2 VCs	11.3 mm^2	307 GB/s	36.8 mm^2/TBps
Hard 64-NoC	Hard	1035 MHz, 32 bits, 1 VC	7.07 mm^2	236 GB/s	30.0 mm^2/TBps
Hard 64-NoC	Hard	957 MHz, 64 bits, 1 VC	10.1 mm^2	437 GB/s	23.1 mm^2/TBps

Conventional Point-to-Point FPGA Interconnect				
FPGA Interconnect	Description	Area	Bandwidth	Area per BW
C4 and R4	200 MHz, 10000 bits	2.2 mm^2	250 GB/s	8.8 mm^2/TBps

as fast as the routers at 943 MHz raising its aggregate bandwidth to 307 GB/s. The area-per-BW for this NoC is 1.6× lower than hard NoCs with soft links.

Some have suggested that VCs consume area and power excessively [Huan and DeHon 2012]. We investigate a one-VC version of our hard NoC with hard links and find that it does, in fact, improve area-per-BW. Moving to one VC increases blocking at router ports, reducing aggregate bandwidth by 23%. However, area drops by 60% resulting in a reduced area-per-BW of only 30 mm^2/TBps. Finally, by increasing the flit data width of the NoC from 32 to 64 bits, we double its bandwidth while increasing area by only 61%. This increases area efficiency to 23.1 mm^2/TBps, as the router control logic area is amortized over more data bits. This area-per-BW is only 2.6× higher than that of the conventional FPGA wires (8.8 mm^2/TBps).

The results show that soft NoCs consume much area and are impractical for high-throughput applications on FPGAs; however, they may be useful for control-plane and low throughput purposes. Hard NoCs are two orders of magnitude more efficient than soft NoCs. Additionally, lower VCs and higher data widths are favorable in their implementation. When compared against the overhead of point-to-point links, an efficient hard NoC is only 2.6× larger for the same supported bandwidth. This is by no means a head-to-head comparison because, unlike point-to-point links, NoCs are capable of switching data and arbitrating between multiple communicating modules. However, this comparison against the lower bound puts hard NoCs in perspective and strongly suggests that hard NoCs will exceed the efficiency of more complex types of soft interconnect that can also perform arbitration and switching.

6. CONCLUSION

Augmenting future FPGAs with NoCs would facilitate interfacing to high-speed I/Os, simplify compilation and partial reconfiguration via modularity, and ease the timing closure bottleneck. To inform the architecture of such an NoC we have investigated the area and delay gap per NoC component for hard versus soft implementation. Further, we presented NoC architectures that are embedded within the FPGA fabric; one uses soft links and the other hard links. Both the mixed and hard NoCs showed large area (20-23×) and delay (5-6×) improvements over conventional soft NoCs. A thorough analysis of the available area budget showed that 64-node hard NoCs use only a very small fraction (~2%) of the FPGA area.

By simulating NoC routers we were able to find the actual aggregate bandwidth of NoCs. We use this to compute an area-per-bandwidth metric to compare NoCs against conventional point-to-point links. Our optimized NoC spends only 2.6× more area-

per-BW compared to soft point-to-point links; therefore, we expect that hard NoCs will easily exceed the efficiency of more complex soft communication systems such as logical buses.

ACKNOWLEDGMENTS

The authors would like to thank Daniel Becker for the open-source router, Natalie Enright Jerger, David Lewis and Dana How for valuable discussions, and CMC for the ASIC tools.

REFERENCES

M. S. Abdelfattah and V. Betz. 2012. Design Tradeoffs for Hard and Soft FPGA-based Networks-on-Chip. In *FPT*. 95–103.

Altera Corp. 2007. Stratix III FPGA: Lowest Power, Highest Performance 65-nm FPGA. Press Release.

J. Balfour and W. J. Dally. 2006. Design Tradeoffs for Tiled CMP On-Chip Networks. In *ICS*. 187–198.

D. U. Becker and W. J. Dally. 2009. Allocator Implementations for Network-on-Chip Routers. In *SC*. 1–12.

Himanchu Bhatnagar. 2002. *Advanced ASIC Chip Synthesis using Synopsys Design Compiler, Physical Compiler and Primetime*. Kluwer Academic Publishers, Norwell, MA.

E. S. Chung, J. C. Hoe, and K. Mai. 2011. CoRAM: An In-Fabric Memory Architecture for FPGA-based Computing. In *FPGA*. 97–106.

D. U. Becker. 2012. *Efficient Microarchitecture for NoC Router*. Ph.D. Dissertation. Stanford University.

W. J. Dally and B. Towles. 2001. Route Packets, Not Wires: On-Chip Interconnection Networks. In *DAC*. 684–689.

W. J. Dally and B. Towles. 2004. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, Boston, MA.

R. Francis and S. Moore. 2008. Exploring Hard and Soft Networks-on-Chip for FPGAs. In *FPT*. 261–264.

K. Goossens, M. Bennebroek, J. Y. Hur, and M. A. Wahlah. 2008. Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects. In *NOCS*. 45–54.

R. Ho, K. W. Mai, and M. A. Horowitz. 2001. The Future of Wires. *Proc. IEEE* 89, 4, 490–504.

Y. Huan and A. DeHon. 2012. FPGA Optimized Packet-Switched NoC using Split and Merge Primitives. In *FPT*. 47–52.

M. Hutton, D. Karchmer, B. Archell, and J. Govig. 2005. Efficient static timing analysis and applications using edge masks. In *FPGA*. 174–183.

I. Kuon and J. Rose. 2007. Measuring the Gap Between FPGAs and ASICs. *TCAD* 26, 2, 203–215.

J. Lee and L. Shannon. 2010. Predicting the Performance of Application-Specific NoCs Implemented on FPGAs. In *FPGA*. 23–32.

D. Lewis, D. Cashman, M. Chan, J. Chromczak, G. Lai, A. Lee, T. Vanderhoek, and H. Yu. 2013. Architectural Enhancements in Stratix V. In *FPGA*. 147–156.

M. K. Papamichael and J. C. Hoe. 2012. CONNECT: Re-Examining Conventional Wisdom for Designing NoCs in the Context of FPGAs. In *FPGA*. 37–46.

G. Passas, M. Katevenis, and D. Pnevmatikatos. 2012. Crossbar NoCs Are Scalable Beyond 100 Nodes. *TCAD* 31, 4, 573–585.

G. Schelle and D. Grunwald. 2008. Exploring FPGA Network on Chip Implementations across Various Application and Network Loads. In *FPL*. 41–46.

R. Scoville. 2010. TimeQuest User Guide.

B. Sethuraman, P. Bhattacharya, J. Khan, and R. Vemuri. 2005. LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip. In *GLSVLSI*. 452–457.

Synopsys Inc. 2010. *Design Compiler Optimization Reference Manual*.

Y. Tamir and G. L. Frazier. 1988. High-Performance Multi-Queue Buffers for VLSI Communication Switches. In *ISCA*. 343–354.

L. G. Valiant and G. J. Brebner. 1981. Universal Schemes for Parallel Communication. In *STOC*. 263–277.

H. Wong, V. Betz, and J. Rose. 2011. Comparing FPGA vs. Custom CMOS and the Impact on Processor Microarchitecture. In *FPGA*. 5–14.

Received May 2013; revised September 2013; accepted November 2014