AUGMENTING FPGAS WITH EMBEDDED NETWORKS-ON-CHIP

Mohamed S. Abdelfattah and Vaughn Betz

Department of Electrical and Computer Engineering
University of Toronto, Toronto, ON, Canada
{mohamed,vaughn}@eecg.utoronto.ca

ABSTRACT

FPGAs are increasing in capacity, allowing the implementation of ever-larger systems with correspondingly increasing bandwidth demands. Additionally, modern FPGAs are becoming a heterogeneous platform that includes many fast dedicated blocks such as processor cores, and high-speed I/Os such as DDR3 memory and PCIe. It is becoming a challenge to connect systems in these large heterogeneous FPGAs using the existing low-level interconnect that consists of wires and programmable multiplexers. We propose embedding a high-bandwidth network on-chip (NoC) on future FPGAs to transport data in large systems. Two embedded NoC types offer different tradeoffs; mixed NoCs – that use embedded "hard" routers and programmable "soft" links – have a configurable topology, while hard NoCs – that consist of both hard routers and hard links – are more efficient and faster. Both mixed and hard NoCs demonstrate a significant advantage in both efficiency and performance over soft NoCs that are traditionally implemented on FPGAs; 20-23× smaller, 5-6× faster and 9-11× lower power. This work summarizes our previous findings that appeared in FPT 2012 and FPL 2013 [1, 2].

1. INTRODUCTION

Field programmable gate-arrays (FPGAs) are reconfigurable computer chips used for a wide range of applications including video broadcast, wireless communication and packet processing for example. FPGAs consist of a mesh array of hundreds-of-thousands of logic elements that can be connected in many flexible ways using a reconfigurable interconnect. This FPGA interconnect consists of different-length wires and programmable multiplexers; wires are stitched together using these multiplexers to create flexible connections between any of the FPGA's logic blocks or I/Os.

1.1. Problems with Current FPGA Interconnect

While essential to FPGAs, this programmable interconnect is facing many challenges:

1. Interconnect scaling: FPGA connections consist of a number of wires and multiplexers; these multiplexers are made from pass-transistors [3]. Both metal wires and pass transistors are not scaling as well as logic transistors meaning that interconnect delay is accounting for an ever-larger portion of a design's critical path delay. This poor delay scaling of FPGA interconnect is thus limiting FPGA speed.

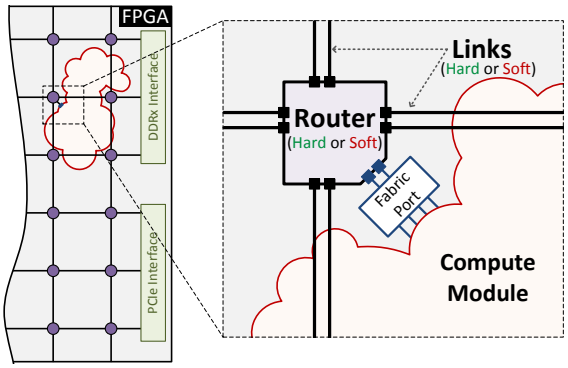

Fig. 1: A mesh NoC implemented on an FPGA. The example shows one router connected to a compute module and three links connected to each of the DDR and PCIe interfaces.

2. Design hurdles: Poor interconnect scaling also makes design more difficult. We cannot accurately predict the delay of interconnect wires except at the very-last stages of compilation (placement and routing). A designer will often find that his/her design does not meet its target clock frequency because of interconnect delay, requiring him to go back and change the design (for example insert pipeline stages in the interconnect) multiple times; a time- and effort-consuming process. Furthermore, FPGA design tools need to configure each and every interconnect multiplexer which slows down compilation.
3. FPGA bandwidth demands: FPGAs now include many on-chip dedicated (or "hard") compute and memory elements, and I/O interfaces and controllers. Examples are dedicated multiplication units, small block RAMs and even complete processor cores. More importantly, FPGAs include dedicated fast I/O modules such as DDR3 memory controllers and PCIe or Ethernet transceivers. These dedicated blocks run much faster than the FPGA logic, often 8 times faster in the case of DDR3 memory for example. This requires a very wide datapath (8 times wider than DDR3 width because it is 8 times slower) to be configured on the FPGA to transport the immense bandwidth coming from off-chip memory. This datapath is both difficult to design (for reasons mentioned in point 2) and uses much FPGA logic and interconnect resources.
4. Modularity: The low-level abstraction of FPGA interconnect is a barrier to dividing a design into modules

for independent optimization and compilation. Modularity can aid new FPGA domains such as partial reconfiguration and help with parallel compilation of hardware designs.

1.2. Proposed Solution: Embedded Networks-on-Chip

To tackle current interconnect problems we propose *augmenting* FPGAs with embedded Networks-on-Chip (NoCs) for system-level interconnection. Our embedded NoC does not rid the FPGA of its current interconnect, it augments it; the situation is analogous to a growing metropolitan city. Small roads and bridges are not sufficient to handle the city's ever-increasing traffic –we must have wide high-speed freeways to move traffic efficiently. Similarly, we aim to embed an NoC on the FPGA using dedicated (hard) logic thus making it fast enough to transport high-bandwidth data streams across the FPGA efficiently.

An NoC can be regarded as a network of *pre-pipelined* interconnect since it consists of short connections between pipelined routers. Because the timing properties of an embedded NoC are well-known when the FPGA is implemented, they ease many of the design hurdles imposed by poor interconnect scaling. 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 [4]. For instance, NoCs can 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 connect each of its new wires in an already-functioning FPGA.

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. Dedicated 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 embedded 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, an embedded "hard" NoC also has area, delay and power advantages over a conventional"soft" NoC configured from FPGA fabric elements, 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.

1.3. Prior Work

There is prior work both on soft:hard efficiency comparison and on FPGA-based NoCs. A comparison of FPGAs and ASICs (soft vs. hard) by Kuon and Rose 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 [5]. Most prior FPGA

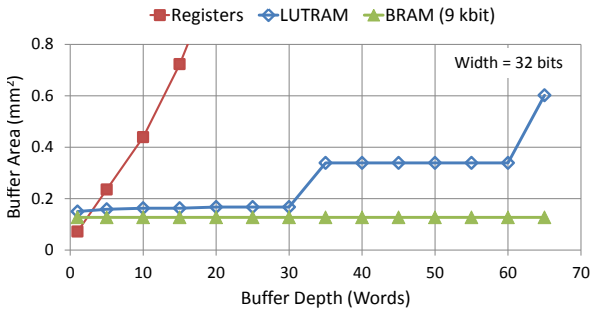

Fig. 2: Variation of physical FPGA area of memory buffers using three implementation alternatives.

NoC research aimed to build soft NoCs efficiently out of the FPGA fabric. LiPar [6], NoCem [7] and CONNECT [8] are three virtual channel (VC) NoCs implemented efficiently in soft logic on FPGAs. There has been little work on embedded NoCs. Francis and Moore suggest that a circuit-switched network with time-division multiplexed links should be hardened on the FPGA [9]. Goossens et al. propose use of a hardwired NoC for both functional communication and FPGA configuration programming [10]. Chung et al. present a programming model that abstracts the distribution of data from external memory throughout the FPGA and mention that their application could benefit from an embedded NoC [11].

Our work has focused on the implementation of NoCs on FPGAs. First, we analyzed the detailed area and delay of NoC components in [1, 12], then looked at complete embedded NoC systems and analyzed their power and performance in [2]. This paper presents a summary and discussion of our findings thus far.

2. NETWORK ARCHITECTURE

NoCs consist of routers and links. Routers perform distributed buffering, arbitration and switching to decide how data moves across a chip, and links are the physical wires that carry data between routers. Additionally, embedded NoCs require a "fabric port" to connect an NoC to the FPGA fabric as shown in Fig. 1. Unlike multiprocessor NoCs for instance, we cannot tailor the NoC to a specific application because the FPGA application is not known when manufacturing the FPGA; we must design the NoC in such a way that it can connect to any module configured onto the FPGA. We outline this key component, the "fabric port", at the end of this section after presenting the different embedded NoCs.

Both routers and links can be either "soft" or "hard". Soft implementation means configuring the NoC out of the conventional FPGA fabric (logic blocks, etc..) while hard implementation refers to embedding the NoC as unchangeable logic on the FPGA chip. We therefore discuss three kinds of NoCs: "soft NoCs" configured out of the FPGA's soft logic resources, "hard NoCs" implemented as dedicated unchangeable logic, and "mixed NoCs" that include both hard and soft components.

2.1. Soft NoCs: Soft Routers and Soft Links

Soft NoCs can be implemented on current FPGAs without making any architectural changes to the FPGA. It is simply

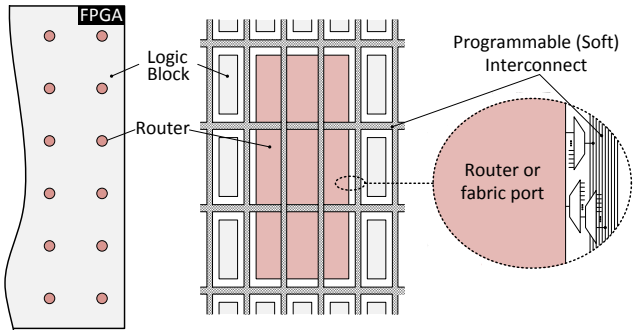

Fig. 3: Floor plan of a hard router with soft links embedded in the FPGA fabric. Drawn to a realistic scale.

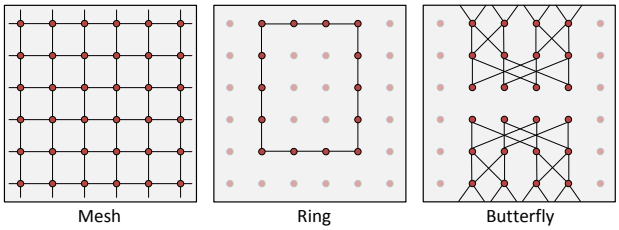

Fig. 4: Examples of different topologies that can be implemented using the soft links in a mixed NoC.

the implementation of NoC hardware on FPGA fabric resources such as logic blocks, small memory blocks and programmable interconnect. The advantage of soft NoCs lies in their reconfigurability; only exactly what is needed can be configured onto the FPGA. However, high-performance NoCs such as the ones we aim to implement are both area-and-power-inefficient and slow on FPGAs.

We make sure that our NoC hardware description language (HDL) code targets the FPGA fabric resources efficiently. A critical component of the virtual-channel routers we use is the input buffers; they occupy a large fraction of the total NoC area. On FPGAs, we can implement such buffers using three different resource types:

1. Registers: There are many registers on the FPGA but they are sparsely spaced; buffers created out of registers are therefore very large.
2. LUTRAM: FPGA logic elements are created out of lookup tables (LUTs) which are essentially small memories (~64 bits each). Multiple LUTs can be combined and used as a small RAM memory.
3. BRAM: These are densely-packed block RAM (BRAM) modules, typically 9 kbits each.

As Fig. 2 shows, we found that BRAM were most silicon-area-efficient in implementing most buffers, even those of a very small size [1]. We therefore use them in implementing soft NoCs that are used to compare to the mixed and hard NoCs.

2.2. Mixed NoCs: Hard Routers and Soft Links

In this NoC architecture, we embed hard routers on the FPGA and connect them via the soft FPGA interconnect. Similarly to logic blocks 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. We connect the router to the interconnect fabric with the same multiplexer flexibility as a logic block and we ensure that enough programmable interconnect wires intersect its layout to feed all of the inputs and outputs. Fig. 3 shows a detailed to-scale illustration of such an embedded router. A 32-bit 2-VC hard router occupies the area equivalent to 9 logic blocks on Stratix-III FPGAs even after accounting for the programmable multiplexers to connect to the FPGA interconnect [1, 12]. In comparison, soft routers occupy an area equivalent to ~270 logic blocks (30× larger).

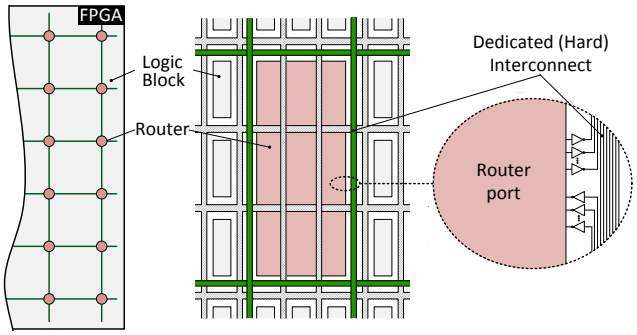

Fig. 5: Floor plan of a hard router with hard links embedded in the FPGA fabric. Drawn to a realistic scale.

The speed of mixed NoCs is limited by the soft interconnect as hard routers can operate at higher frequencies as we show in Section 4. 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 subject only to the limit that we cannot exceed the router port count at FPGA fabrication time for our routers. This includes implementing topologies that use only a subset of the available routers or implementing two separate NoCs as shown in Fig. 4. To accommodate different NoCs, routing tables inside the router control units are simply reprogrammed to match the new topology.

2.3. Hard NoCs: Hard Routers and Hard Links

This NoC architecture involves hardening both the routers and the links. 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 can run at higher speeds than soft links, allowing the NoC to achieve the router's maximum frequency. Drivers at the ends of dedicated wires charge and discharge data bits onto the hard links as shown in Fig. 5. A hard NoC's speed (above 900 MHz) is beyond that of the programmable clock networks on most FPGAs; accordingly it also requires a dedicated clock network to be added to the FPGA. Such a clock network is fast and very cheap in terms of metal usage since it is not configurable and has only as many endpoints as the number of routers in an NoC; typically less than 64 nodes. In contrast, FPGAs have more than 16 configurable clock networks with ~600 endpoints each.

A hard NoC is almost completely disjoint from the FPGA

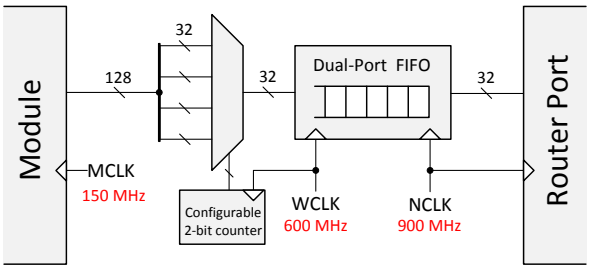

Fig. 6: Fabric port with configurable 4:1 time-domain multiplexing logic. Example frequencies are annotated.

fabric, only connecting through router-to-fabric ports. This makes it easy to use a separate power grid for the NoC with a lower voltage than the nominal FPGA voltage. This is desirable because we can trade excess NoC speed for power efficiency. The only added overhead is the area of the voltage crossing circuitry at the router-to-fabric interfaces, and this is minimal. We therefore explore hard NoCs both at the FPGA's nominal voltage (1.1 V for Stratix III) and low-power versions at 0.9 V.

2.4. "Fabric Port" – How FPGA NoCs are different

Although we can use most of the concepts of well-established multiprocessor NoCs in designing embedded NoCs for FPGAs, there are a few key differences. We know little about the final designs that will be implemented on the FPGA while architecting it, and hence as we embed a system-level interconnect we cannot tailor it to one class of application; we must over-provision our embedded interconnect to account for *any* application that could fit on an FPGA. This is why NoCs are of interest, and not other widely used interconnects such as buses. NoCs conform well to the island-style mesh organization of the FPGA logic blocks unlike hierarchical buses whose design involves assumptions about how modules are floorplanned. Additionally, we must overprovision the NoCs to be able to transport the maximum bandwidth of big FPGA applications. To determine that, we compute the bandwidth of the high-speed interfaces and controllers (e.g. DDR3, PCIe, Ethernet) present on modern FPGAs; these I/Os are crucial because they are used in essentially all commercial FPGA designs. Our embedded NoCs must therefore be capable of distributing data from these interfaces throughout the FPGA. Finally, we do not know the speed or data width of the end applications during the manufacture of the FPGA; for that we use a "fabric port" at the interface between NoC routers and FPGA modules to bridge the frequency and data width between the two while maximizing NoC bandwidth.

The FPGA fabric uses multiple relatively slow (~100-400 MHz) clocks [13], while the NoC runs on a single very fast clock (~1 GHz) as we show in Section 3. 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. 6; we perform both width adaptation and clock crossing. The example in Fig. 6 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. 6 are examples; the TDM factor can be decreased by configuring the counter (to implement 2:1 TDM for example), or increased by augmenting the depicted 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.

Table 1: Summary of Soft (FPGA) vs. Hard (ASIC) router area, delay and power ratios.

Module	Area	Delay	Power
Input Module	17	2.9	10
Crossbar	85	4.4	64
VC Allocator	48	3.9	41
Switch Allocator	56	3.3	41
Output Module	39	3.4	16
Router	**30**	**6.0**	**14**
Links	**9**	**2.4**	**1.4**

3. NOC COMPONENT ANALYSIS

In this section we analyze NoC components when implemented hard or soft; these are the building blocks used in the analysis of complete soft, mixed and hard NoCs. For soft implementation we use the largest Stratix III FPGA (EP3SL340) and for hard implementation we use TSMC's 65 nm ASIC process technology. This allows a direct FPGA vs. ASIC (soft vs. hard) comparison since Stratix III devices are manufactured in the same 65 nm TSMC process [14]. We use a high-performance packet switched router [15] in designing our NoCs to keep up with the high-bandwidth and low-latency demands of modern FPGAs. For details of the router microarchitecture, please see [1, 15].

3.1. Routers

We perform an analysis of router subcomponents and compare the area, speed and power in Table 1. Routers used in this study are packet-switched virtual-channel (VC) routers [1, 15]; they have 5 components: input modules, crossbar, output modules, and switch and VC allocators.

Table 1 shows that there is a significant gap in efficiency and performance between hard and soft components. Input modules, which account for ~60% of router area have the lowest area, delay and power gaps because its soft implementation includes efficient BRAM modules for memory buffers

as outlined in Section 2.2. Crossbars on the other hand had the largest soft:hard gap. This is because crossbars consist of multiplexers which are inefficient on FPGAs for various reasons [1]. Their high demand for wiring causes many logic blocks to be partially packed with "pieces" of a multiplexer. Area-and-power hungry interconnect is used to connect those pieces into a crossbar which reduces the efficiency of soft crossbars compared to hard ones, and makes them considerably slower as well.

Logic blocks on the FPGA contain both LUTs to implement logic gates, and flip flops for sequential elements; it is therefore a waste of area when the registers are unused and this makes the soft implementation area inefficient. We found a direct correlation between the LUT-to-register ratio (1:1 on Stratix III FPGAs) and the FPGA-to-ASIC (or soft-to-hard) 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×. For output modules the LUT-to-register ratio is 0.6:1 contributing to its smaller area gap of 39× when compared to the allocators.

Overall, routers are 30× smaller, 6× faster and use 14× less power when implemented hard rather than soft [1, 2, 12] – the case for hardening NoC routers is therefore compelling which motivates our mixed and hard NoCs.

3.2. Links

Soft wires connect via multiplexers which increases their capacitive and resistive loading, making them slower and more power hungry. 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, 9× smaller, 2.4× faster and consume 1.4× less power than soft wires as shown in Table 1.

4. SYSTEM-LEVEL ANALYSIS

In this section we analyze complete mixed/hard NoC systems for FPGAs.[1] We therefore quantify the area and power overhead of these NoCs as a fraction of the available FPGA area and power budgets. We also compare the raw efficiency of different NoC types to conventional point-to-point links implemented on FPGAs; the simplest form of FPGA interconnect.

4.1. NoC Systems Comparison

There is a large area difference between soft NoCs and embedded (mixed/hard) NoCs; the area gap equals 20× and 23× for hard and mixed NoCs respectively as shown in Table 2. Additionally, both the performance and power differences are considerable; mixed and hard NoCs are 5-6× faster and 9-11× lower power. We study 64-node examples of these NoCs in Fig. 7 to put these differences in perspective.

[1]To access and visualize our complete area/delay/power results, please visit: `www.eecg.utoronto.ca/~mohamed/noc_designer.html`

Table 2: Efficiency and performance of mixed and hard NoCs, compared to soft NoCs.

NoC	Area	Delay	Power
Soft	1×	1×	1×
Mixed	20× smaller	5× faster	9× lower
Hard	23× smaller	6× faster	11× lower

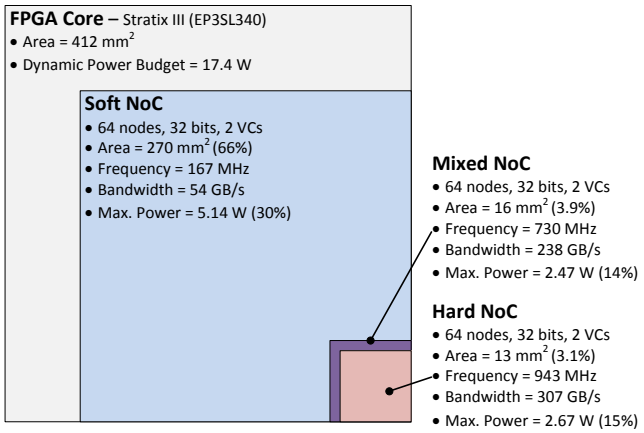

Fig. 7: A to-scale illustration of the areas of 64-node NoCs relative to the largest Stratix III FPGA device. NoC properties and metrics are annotated on the figure.

Fig. 7 shows the relative areas of different NoC types and the largest 65-nm Stratix III FPGA device. The figure also shows how much of a large FPGA each NoC type consumes. Soft NoCs are prohibitively large. A 64-node version uses most of the FPGA area; this is 66% of the entire FPGA core area. In addition, a soft NoC cannot operate at high frequencies thus limiting total NoC bandwidth to tens of gigabytes per second (54 GB/s). This explains why soft NoCs have not been adopted in mainstream FPGA applications; they are very inefficient and low-performance.

On the other hand, embedded (mixed/hard) NoCs are small and fast. The same 64-node NoC, when implemented using hard routers and hard links occupies only 13mm^2 (3% of the FPGA) and its higher frequency boosts its bandwidth to the hundreds of gigabytes per second (307 GB/s). If a soft NoC were required to provide that amount of aggregate bandwidth, it would not fit on the Stratix III FPGA device.

We see the same trend with power. Soft NoCs dissipate ~9 W for each 100 GB/s of bandwidth transported by the NoC, while embedded NoCs only require ~1 W per 100 GB/s. The large efficiency and performance gains achieved by hardening NoC components motivates the implementation of embedded NoCs onto modern FPGAs to implement system-level communication. Embedded NoCs support more than enough bandwidth to transport high-speed data from fast I/Os across the FPGA efficiently. For instance, if we were to transport the maximum theoretical bandwidth of a 64-bit 933 MHz DDR3 memory and PCIe Gen3 x8 transceivers (2 of the fastest I/Os connected to FPGAs) we would require a total bandwidth of ~126 GB/s which is well within the limits of our embedded NoCs [2].

Table 3: Raw efficiency of different NoCs and conventional FPGA point-to-point links.

	Energy per Data	Area per Bandwidth
P2P Links	4.73 mJ/GB	8.8 mm^2/TBps
Soft	94.5 mJ/GB	4960 mm^2/TBps
Mixed	10.4 mJ/GB	59.4 mm^2/TBps
Hard	8.68 mJ/GB	36.8 mm^2/TBps
$Hard^{Optimized}$	4.47 mJ/GB	23.1 mm^2/TBps

Please see [2, 12] for a more thorough study and methodology.

4.2. FPGA Area and Power Budgets for NoCs

We anticipate choosing embedded NoCs that provide 200-300 GB/s of bandwidth, similar to the 64-node NoCs that we show in Fig. 7. As such, we quantify the area and power budgets of these NoCs as a percentage of the total available FPGA area and power. We found that embedded (mixed/hard) NoCs only require 3-4% of the FPGA core area, compared to 66% for soft. For power, ~15% of the typical device power budget was sufficient to support 230-300 GB/s of bandwidth in the case of mixed and hard NoCs. On the other hand, soft NoCs that provide the same bandwidth are impossible to implement on FPGAs as they would require ~24 W; 40% more than the typical dynamic power budget of a large FPGA. Not to mention that their area would also be too large to implement on our example FPGA device.

4.3. Embedded NoCs vs. Point-to-Point Interconnect

Currently I/O interfaces are connected to FPGA applications not using soft NoCs, but instead using soft buses that are constructed out of multiplexers and arbiters that use the FPGA fabric elements. These soft buses are tailored to the exact requirements of the application running on the FPGA and hence make use of the FPGA's reconfigurability. A meaningful comparison would then be to compare these tailored soft buses – used to connect large systems – to our embedded NoCs since they provide an alternative interconnection solution. Indeed this efficiency vs. configurability comparison is the next logical step, and this constitutes our current and future work.

We started by looking at the *raw* efficiency of our NoCs, and by comparing this to the *raw* efficiency of interconnect that can be configured out of current FPGAs. As mentioned, soft buses are typically configured onto FPGAs to interconnect systems – we defer that comparison to future work. Another simpler form of interconnect is point-to-point (P2P) links; this is interconnect that consists mainly of FPGA wires that connects two modules together. We define our raw efficiency metrics as normalized area and power per unit of bandwidth; that means, we compute how much energy is dissipated for each gigabyte transported on different interconnects, and we measure how much area is spent with each kind of interconnect to support one gigabyte-per-second of bandwidth [2, 12]. These metrics can compare different kinds of interconnect while abstracting their implementation details.

Table 3 offers a summarized comparison; the NoCs have the same properties as annotated in Fig. 7, and the P2P links provide similar bandwidth and are constructed out of a mixture of FPGA wires that are the length of one NoC link. It is apparent from the table that there is a huge difference between soft NoCs and everything else; however, it is more interesting to compare embedded NoCs to the P2P links. Even though embedded NoCs can implement more complex forms of on-chip communication, their raw efficiency comes close to soft P2P links. The hard NoC we present here is only twice as power hungry as P2P links, and we have shown that for higher bandwidth and with different design choices, low-power hard NoCs can even reach 4.47 mJ/GB equaling a P2P link. Similarly, hard NoCs can be as area-efficient as 23.1 mm^2/TBps which is a very low area overhead for the switching capabilities offered; arbitration, buffering and virtual channels. The results are very promising; even when comparing with the simplest form of interconnect, P2P links that are limited in capability and just consist of wires, embedded NoCs score similar efficiency metrics. We therefore believe that embedded NoCs will be more efficient than any other more complex interconnect such as soft buses implemented on FPGAs.

5. FUTURE WORK

Our immediate future work will be to do more in-depth comparisons and case studies of FPGA efficiency and performance. For instance, how do embedded NoCs compare to a soft (tailored) bus-based interconnect that is used to distribute DDR3 memory data across an FPGA? Other microbenchmarks and application studies will help quantify the gain of embedded NoCs and guide its design choices.

REFERENCES

[1] M. S. Abdelfattah and V. Betz, "Design Tradeoffs for Hard and Soft FPGA-based Networks-on-Chip," *FPT*, 2012, pp. 95–103.

[2] M. S. Abdelfattah and V. Betz, "The Power of Communication: Energy-Efficient NoCs for FPGAs," *FPL*, 2013.

[3] D. Lewis and J. Chromczak, "Process technology implications for FPGAs (Invited Paper)," *IEDM*, 2012.

[4] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *DAC*, 2001, pp. 684–689.

[5] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *TCAD*, vol. 26, no. 2, pp. 203–215, 2007.

[6] B. Sethuraman, *et al.*, "LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip," *GLSVLSI*, 2005, pp. 452–457.

[7] G. Schelle and D. Grunwald, "Exploring FPGA network on chip implementations across various application and network loads," *FPL*, 2008, pp. 41–46.

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

[9] R. Francis and S. Moore, "Exploring Hard and Soft Networks-on-Chip for FPGAs," *FPT*, 2008, pp. 261–264.

[10] K. Goossens, *et al.*, "Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects," *NOCS*, 2008, pp. 45–54.

[11] E. S. Chung, *et al.*, "CoRAM: An In-Fabric Memory Architecture for FPGA-based Computing," *FPGA*, 2011, pp. 97–106.

[12] M. S. Abdelfattah and V. Betz, "Networks-on-Chip for FPGAs: Hard, Soft or Mixed?" *TRETS*, 2013.

[13] M. Hutton, *et al.*, "Efficient static timing analysis and applications using edge masks," *FPGA*, 2005, pp. 174–183.

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

[15] Daniel U. Becker, "Efficient Microarchitecture for Network-on-Chip Router," Ph.D. dissertation, Stanford University, 2012.