CONNECT: Re-Examining Conventional Wisdom for Designing NoCs in the Context of FPGAs

Michael K. Papamichael
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
<papamix@cs.cmu.edu>

James C. Hoe
Electrical & Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA, USA
<jhoe@ece.cmu.edu>

ABSTRACT

An FPGA is a peculiar hardware realization substrate in terms of the relative speed and cost of logic vs. wires vs. memory. In this paper, we present a Network-on-Chip (NoC) design study from the mindset of NoC as a synthesizable infrastructural element to support emerging System-on-Chip (SoC) applications on FPGAs. To support our study, we developed CONNECT, an NoC generator that can produce synthesizable RTL designs of FPGA-tuned multi-node NoCs of arbitrary topology. The CONNECT NoC architecture embodies a set of FPGA-motivated design principles that uniquely influence key NoC design decisions, such as topology, link width, router pipeline depth, network buffer sizing, and flow control. We evaluate CONNECT against a high-quality publicly available synthesizable RTL-level NoC design intended for ASICs. Our evaluation shows a significant gain in specializing NoC design decisions to FPGAs' unique mapping and operating characteristics. For example, in the case of a 4x4 mesh configuration evaluated using a set of synthetic traffic patterns, we obtain comparable or better performance than the state-of-the-art NoC while reducing logic resource cost by 58%, or alternatively, achieve 3-4x better performance for approximately the same logic resource usage. Finally, to demonstrate CONNECT's flexibility and extensive design space coverage, we also report synthesis and network performance results for several router configurations and for entire CONNECT networks.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks*

General Terms

Design, Experimentation, Performance

Keywords

Network-on-Chip, NoC, FPGA, System-on-Chip, SoC

FPGA'12, February 22–24, 2012, Monterey, California, USA.

1. INTRODUCTION

The rapidly-growing capacity of Field Programmable Gate Arrays (FPGAs), combined with the steady introduction of hardwired support for a multitude of diverse interfaces and functionalities, has promoted FPGAs to an attractive and capable platform for hosting even extended System-on-Chip (SoC) designs [9]. As the scale of designs targeting FPGAs grows, designers need a systematic and flexible Network-on-Chip (NoC) infrastructure to support communication between the tens and in the future potentially hundreds of interacting modules. In this paper, we present our investigation in synthesizable NoC designs specifically architected and tuned for FPGAs for use with the development of SoCs and other demanding systems applications, such as full-system prototyping [5] and high performance computing [4].

The research literature offers a large body of work on NoCs mapped onto FPGAs for the purpose of NoC simulation studies [25, 31] and for the purpose of prototyping and SoC emulation [24, 30, 15]. In these cases, the actual performance and efficiency of the originally ASIC-oriented NoC designs when mapped to FPGA is not a first-order concern; instead these prior works are motivated to instantiate largely unmodified ASIC-oriented NoC designs to ensure modeling fidelity. There have been only relatively few papers that point out FPGA-specific NoC design issues. We refer to them in our discussions in Sections 3 and 6.

Although the FPGA design flow and the ASIC design flow have much in common in their similar-looking RTL-based design and synthesis environments, they are in fact very different when it comes to making design decisions affecting cost and performance optimizations. A "literal" adaptation of an ASIC-optimized RTL-level NoC design on an FPGA will almost certainly prove to be suboptimal. What may be a compactly optimized router on an ASIC can incur a disproportionately high cost when synthesized for an FPGA because of the FPGA's very different relative cost tradeoff between logic, wires and memory. Worse yet, ASIC-motivated optimizaitons will likely not be as effective due to the FPGA's also very different relative speeds in logic, wires and memory. FPGA design optimizations are further complicated by quantization effects because user logic and memory are realized using discretized underlying physical structures of fixed capacity and geometry.

In this work, we take full consideration of FPGAs' special hardware mapping and operating characteristics to identify their own specialized NoC design sweet spot, which we will show is very different from the conventional wisdom stemming from NoC designs on ASICs. Specifically, the con-

siderations that have motivated this work to rethink NoC design for FPGAs are (1) FPGAs' relative abundance of wires compared to logic and memory; (2) the scarcity of on-die storage resources in the form of a large number of modest-sized buffers; (3) the rapidly diminishing return on performance from deep pipelining; and (4) the field reconfigurability that allows for an extreme degree of application-specific fine-tuning.

To support this investigation, we created the CONNECT NoC design generator that can generate synthesizable RTL-level designs of multi-node NoCs based on a simple but flexible fully-parameterized router architecture. The CONNECT NoC architecture embodies a set of FPGA-motivated design principles that uniquely influence key NoC design decisions, such as topology, link width, router pipeline depth, network buffer sizing, and flow control.

In the results section, we compare FPGA synthesis resource usage and network performance results for two instances of CONNECT NoCs to a high-quality state-of-the-art ASIC-oriented NoC design, to demonstrate the effectiveness of the FPGA-specialized tuning and features of the CONNECT NoC architecture. In addition, to highlight the flexibility and adaptability of the CONNECT NoC architecture, we also include synthesis and network performance results for a variety of diverse router and network configurations. Overall, the results of our investigation support that through FPGA specialization, we can gain approximately a factor of two savings in implementation cost without experiencing any significant performance penalty—in many cases, the CONNECT FPGA-tuned router design can actually lead to better performance at a lower implementation cost.

The rest of this paper is organized as follows. Section 2 provides a brief review of key NoC terminology and concepts. Section 3 introduces the motivations behind the CONNECT NoC architecture, and Section 4 presents the architecture of CONNECT-based routers and NoCs. In Section 5 we evaluate a CONNECT-based 4x4 mesh network against an equivalent NoC implemented using publicly available high-quality state-of-the-art RTL and show FPGA synthesis and network performance results for various CONNECT router and NoC configurations. Finally, we examine related work in Section 6, discuss future directions in Section 7 and conclude in Section 8.

2. NOC BACKGROUND

This section offers a brief review of key NoC terminology and concepts relevant to this paper. For a more comprehensive introduction please see [8]. Readers already familiar with NoCs may continue directly to Section 3.

Packets. Packets are the basic logical unit of transmission at the endpoints of a network.

Flits. When traversing a network, packets, especially large ones, are broken into flits (flow control digits), which are the basic unit of resource allocation and flow control within the network. Some NoCs require special additional "header" or "tail" flits to carry control information and to mark the beginning and end of a packet.

Virtual Channels. A channel corresponds to a path between two points in a network. NoCs often employ a technique called virtual channels (VCs) to provide the abstraction of multiple logical channels over a physical underlying channel. Routers implement VCs by having non-interfering flit buffers for different VCs and time-multiplexed sharing of the switches and links. Thus, the number of implemented VCs has a large impact on the buffer requirements of an NoC. Employing VCs can help in the implementation of protocols that require traffic isolation between different message classes (e.g. to prevent deadlock [7]), but can also increase network performance by reducing the effects of head-of-line blocking [22].

Flow Control. In lossless networks a router can only send a flit to a downstream receiving router if it is known that the downstream router's buffer has space to receive the flit. "Flow control" refers to the protocol for managing and negotiating the available buffer space between routers. Due to physical separation and the speed of router operation, it is not always possible for the sending router to have immediate, up-to-date knowledge of the buffer status at the receiving router. In credit-based flow-control, the sending router tracks credits from its downstream receiving routers. At any moment, the number of accumulated credits indicates the guaranteed available buffer space (equal to or less than what is actually available due to delay in receiving credits) at the downstream router's buffer. Flow control is typically performed on a per-VC basis.

Input-Output Allocation. Allocation refers to the process or algorithm of matching a router's input requests with the available router outputs. Different allocators offer different trade-offs in terms of hardware cost, speed and matching efficiency. Separable allocators [8] form a class of allocators that are popular in NoCs. They perform matching in two independent steps, which sacrifices matching efficiency for speed and low hardware cost.

Performance characterization. The most common way of characterizing an NoC is through load-delay curves, which are obtained by measuring packet delay under varying degrees of load for a set of traffic patterns. A common metric for load is the average number of injected flits per cycle per network input port. Packet delay represents the elapsed time from the cycle the first flit of a packet is injected into the network until the cycle its last flit is delivered. For a given clock frequency, load and delay are often reported in absolute terms, e.g. Gbits/s and ns.

3. TAILORING TO FPGAS

Compared to ASICs, an FPGA is a peculiar hardware realization substrate because it dictates a very different set of design tradeoffs between logic, wires, memory and clock frequency. In this section we focus on specific FPGA characteristics and show how they have influenced fundamental CONNECT design decisions.

3.1 "Free" Wires

FPGAs are expected to be able to handle a wide range of designs with varying degrees of connectivity. Consequently, as previously also noted by other work [26, 20], FPGAs are provisioned, even over-provisioned, with a highly and densely connected wiring substrate. For the average application, this routing resource is likely to be underutilized. In these cases, one could view wires as plentiful or even "free", especially relative to the availability of other resources like configurable logic blocks and on-chip storage (flip-flops and SRAMs). This relative abundance of wires speaks against the conventional wisdom in NoC design where routers are typically viewed as internally densely connected components that are linked to each other through narrow channels that try to multiplex a lot of information through a small set of wires.

Implications. A NoC for FPGAs should attempt to make maximal use of the routing substrate by making the datapaths and channels between routers as wide as possible to consume the largest possible fraction of the available (otherwise unused) wires. Moreover, flow control mechanisms could also be adapted to use a wider interface, which as we show later, can indirectly also reduce router storage requirements. Design decisions such as widening the datapath can even have an indirect effect on issues like packet format. For instance, information that would otherwise be carried in a separate header flit could instead be carried through additional dedicated control wires that run along the data wires. Furthermore, on FPGAs (as well as ASICs actually), the boundaries between one router and another are not sharp—there is not a 10-foot cable separating them like in the old days. We will see later where the CONNECT NoC architecture allows the logic in one router to reach directly into another router for a more efficient implementation of buffer flow control.

3.2 Storage Shortage

Modern FPGAs provide storage in two forms: (1) SRAM macros with tens of kilo-bits of capacity, and (2) small tens-of-bits SRAMs based on logic lookup tables. In this paper, we refer to the former as Block RAMs and the latter as Distributed RAMs, following Xilinx's terminology. The bulk of the available storage capacity (in terms of bits) come in the form of a modest number of Block RAMs. These monolithic memory macros can not be subdivided. This leads to an inefficiency because a full Block RAM must be consumed, even if only a fraction of its capacity is required. Compared to Block RAMs, the Distributed RAMs are very expensive, especially when forming large buffers, since every Distributed RAM consumed is taking away from valuable logic implementation resources. This sets up a situation where NoCs on FPGAs pay a disproportionately high premium for storage because NoCs typically require a large number of buffers whose capacities are each much bigger than Distributed RAMs but much smaller than the Block RAMs. This premium has the consequences of not only limiting the scale of NoCs that can be practically built, but also reducing the resources available to the user logic.

Implications. Given the comparatively high premium for storage, a NoC tuned for FPGA should have a higher threshold for optimizations that increase buffer size in exchange for performance or functionality (e.g., number of VCs), especially when the increase requires consuming Block RAMs that are likely to be in high-demand from the user logic as well. The CONNECT NoC Architecture avoids using Block RAMs entirely and uses Distributed RAMs exclusively for its packet buffers. Furthermore, the choice of buffer sizing and configurations in the CONNECT NoC architecture takes into consideration the specific dimensions and sizes of the Distributed RAM building blocks to make the most efficient use of each consumed LUT.

3.3 Frequency Challenged

A design on an FPGA will operate at a much lower clock frequency than when implemented in ASIC; this was one of the gaps studied in [19]. First of all, Look-Up Tables used to implement arbitrary logic functions are inherently slower compared to fixed-function ASIC standard cells. Secondly, in order to emulate arbitrary logic blocks, FPGAs often need to chain a large number of LUT elements, which in turn requires using long interconnect wires. The time spent traversing these wires often ends up being the largest fraction of the critical path in FPGA designs.

Implications. From the perspective of this work, the most important implication from the difference in performance between ASICs and FPGAs actually manifests most strongly in the rapid diminishing return when attempting to deeply pipeline a FPGA design to improve its frequency. In most cases, beyond a small number of stages, it becomes impossible to further subdivide into balanced finer pipeline stages due to the quantization effects of the underlying realization structures and the difficulty in controlling physical details like logic placement, wiring routing, and driver sizing. We will see later that in fact, for FPGA synthesis, the single-stage router used in the CONNECT NoC architecture reaches lower, but still comparable frequency as an ASIC-tuned 3-stage-pipelined router. The FPGA's performance penalty from running at lower frequency is much more efficiently made up by increasing the width of the datapath and links. The shallow pipeline in the CONNECT NoC architecture has the added benefit of reducing network latency as well as greatly reducing the number of precious flip-flops consumed by a router.

3.4 Reconfigurability

The reconfigurable nature of FPGAs sets them apart from ASICs and creates unique opportunities and challenges for implementing an FPGA-oriented NoC. Given the flexibility of FPGAs, an effective NoC design is likely to be called to match up against a diverse range of applications. Fortunately, the NoC itself, making use of the same reconfigurability, can also go to an extreme degree of application-specific customizations that no one would ever consider for a design to be committed to ASICs.

Implications. Instead of a single design instance library IP, the CONNECT NoC Architecture relies on a design generator that can produce NoC instances specifically adapted to match the application or even the specific run-by-run workload. To cover the needs of such a diverse and rapidly changing set of applications, the CONNECT NoC generator is fully parameterized and more importantly topology-agnostic, which means that individual routers can be composed to form arbitrary custom network topologies. Moreover, to minimize changes in the user logic, all CONNECT networks adhere to the same simple standard common interface. From the user's perspective the NoC appears to be a plug-and-play black box device that receives and delivers packets. Rapid prototyping and design space exploration become effortless as any CONNECT network can be seamlessly swapped for another CONNECT network that has the same number of endpoints.

4. CONNECT NOC ARCHITECTURE

CONNECT-based NoCs are meant to be part of larger FPGA-based systems and, as such, must co-exist with the rest of the FPGA-resident components. This means that CONNECT NoCs need to balance between two conflicting goals: (1) provide sufficient network performance to satisfy the communication requirements of the target application; and (2) minimize the use of FPGA resources to maximize the resources available to the rest of the system. CONNECT addresses both goals by making the NoC implementation as efficient as possible, following the principles discussed in the previous section. When compared to ASIC-optimized NoC designs, in many places, the CONNECT NoC archi-

tecture goes directly against conventional NoC design wisdom. These differences can be attributed to two fundamental CONNECT router design decisions, which are summarized below:

- **Single pipeline stage.** Instead of the typical three to five stage pipeline found in most contemporary VC-based router designs, CONNECT employs a single stage router pipeline, leading to lower hardware cost, lower latency and opportunities for simpler flow control and more efficient buffer usage, due to the reduced round-trip time between routers.
- **Tightly-Coupled Routers.** Instead of treating the NoC as a collection of decoupled routers connected through narrow links, CONNECT tries to maximize wire usage, by using wider interfaces, leading to tighter coupling between routers. This includes carrying flit control information (that would traditionally be carried in separate header flits) on additional wires that run along the data wires. This decoupling is also the driving idea behind CONNECT's "peek" flow control mechanism, that allows routers to directly peek at the buffer occupancy information of their downstream receiving routers.

4.1 CONNECT Router Architecture

Driven by the special characteristics of FPGAs, we developed a simple router architecture to serve as the basic building block for composing CONNECT networks. Our router design was implemented using Bluespec System Verilog (BSV) [3], which allowed us to maintain a flexible parameterizable design. CONNECT routers are heavily configurable and among other parameters, they support:

- Variable number of input and output ports
- Variable Number of virtual channels (VCs)
- Variable flit width
- Variable flit buffer depth
- Two flow control mechanisms
- Flexible user-specified routing
- Four allocation algorithms

Router Datapath. Figure 1 shows the architectural block diagram of a CONNECT router. Communication with other routers happens through input and output port interfaces, which can vary in number depending on the router configuration. Each input or output port interface consists of two channels; one channel for sending or receiving data and one side channel running in the opposite direction used to handle flow control. Input and output interfaces are either connected to network endpoints or are used to form links with other routers in the network.

CONNECT routers are organized as a single-stage pipeline to minimize hardware costs, minimize latency and simplify flow control. During each clock cycle a router receives and stores new flits from its input ports and forwards previously received flits through its output ports. Upon entering the router, each incoming flit is first processed by the routing logic to be tagged with the proper output port and is then stored in flit buffers that are organized per input and per virtual channel. To determine which flits will be scheduled to depart from the router, arbitration logic considers flit buffer occupancy and credit availability to decide which flits will traverse the switch and be forwarded through the switch to the output ports. In addition to scheduling flits, the arbitration logic is also responsible for respecting VC

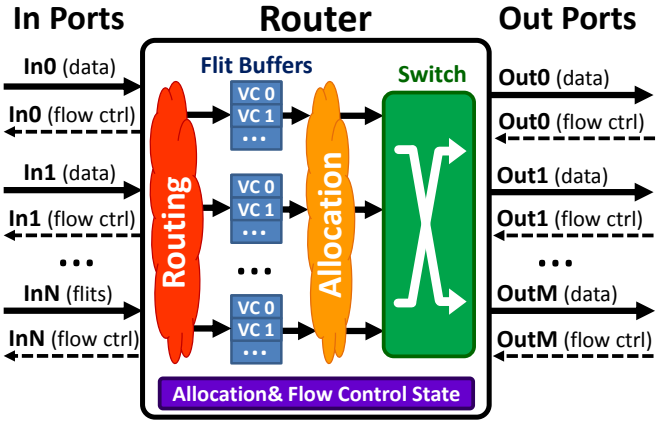

Figure 1: CONNECT Router Architecture

and port priorities, as well as preventing flits from different multi-flit packets from being interleaved on the same virtual channel (when virtual links are enabled).

Each individual router component is optimized to make the most efficient use of FPGA resources. Below we discuss implementation details about some specific router components of interest.

Packet Routing. Routing in CONNECT routers is handled by look-up tables that hold output ports for each possible destination in the network. Look-up based routing provides flexibility to construct arbitrary networks with custom routing. Even though routing look-up tables can grow to require a large number of entries for large networks, CONNECT implements them in an efficient manner by exploiting the geometry of FPGA Distributed RAMs. Each Distributed RAM is typically a single-bit wide memory element with 16 to 64 entries, depending on the specific FPGA family. Since routing tables tend to have many entries (one entry per network node), each being 2-3 bits wide (wide enough to encode a router output port), they map very efficiently to Distributed RAM and in almost all cases they occupy less than 10 LUTs. In many regular topologies, such as mesh or torus, these look-up tables could be easily replaced by topology-specific routing functions implemented in logic.

Flit Buffers. Flit Buffers in CONNECT are organized per input. CONNECT efficiently implements flit buffers using Distributed RAM by implementing multiple logical FIFOs, one per VC, in each single-read single-write Distributed RAM. Each Distributed RAM is split into fixed regions, and each VC-specific FIFO is implemented as a circular buffer within one of these regions. The head and tail pointers required to provide a logical FIFO abstraction occupy minimal area and are stored in discrete registers. This careful buffer space management, allows CONNECT to efficiently scale to large numbers of VCs.

Buffer Allocation. CONNECT supports four variations of separable input-output allocation algorithms [8]. The allocation module consists of two submodules; one that handles input arbitration and one for output arbitration. During each clock cycle the two input and output allocation submodules are triggered in sequence and the results of one submodule are fed to the other in order to produce a valid matching of eligible inputs with available outputs.

4.2 Highlights and Discussion

Below, we focus on some of the most interesting features of the CONNECT NoC Architecture.

Topology-agnostic. A major benefit of allowing any number of input or output ports and being flexible with respect to the routing algorithm is the ability to support arbitrary topologies. As long as flit widths match, CONNECT routers can be hooked to each other and form custom topologies that can better serve the needs of the application at hand. Similarly, all CONNECT networks that connect the same number of endpoints are interchangeable, which can greatly accelerate design space exploration.

Virtual Channels. In order to meet the diverse communication requirements of various applications, CONNECT has support for multiple VCs[1] , which, as explained earlier, are implemented in a very FPGA-efficient manner. Multiple VCs are fundamental for ensuring deadlock freedom, implementing protocols that require traffic isolation between different message classes (e.g., memory requests and responses) and can also be used to increase network performance by reducing the effects of head-of-line blocking [22].

Virtual Links. In order to ease the implementation of receive endpoints in NoCs that use multi-flit packets and employ multiple VCs, CONNECT offers a feature called "Virtual Links". When enabled, this feature guarantees contiguous transmission and delivery of multi-flit packets. In other words, this guarantees that once a packet starts being delivered it will finish before any other packet is delivered. Enabling virtual links can cause a slight increase in hardware cost, but, in return, can significantly reduce the reassembly buffering and logic requirements at the receive endpoints.

Peek Flow Control. In addition to offering traditional credit-based flow control, CONNECT also supports another flow control mechanism, which we call "peek" flow control. Although effective, credit-based flow control can be inefficient in the context of CONNECT, as credit-based flow control is designed to: (1) tolerate long round-trip delays, caused by multi-cycle link latencies and deep router pipelines and (2) minimize the number of required wires between neighboring routers by multiplexing flow-control information pertaining to different VCs over the same set of wires.

In peek flow control, instead of having routers exchange credits, routers effectively expose the occupancy information of all of their buffers to its upstream sending routers. This way, instead of maintaining credits and using them as a proxy to determine how much buffer space is available at the downstream receiving routers, sending routers can directly observe the buffer availability. The peek flow control scheme reduces storage requirements by eliminating the multiple credit counters that are normally maintained for each output and VC pair.

CONNECT currently employs a single-bit version of peek flow control, which is very similar to stop-and-go queuing [14] or simple XON/XOFF flow control schemes [13]. The flow control information exposed by each router corresponds to a single bit per buffer that indicates if the specific buffer is full or not. If round-trip communication delay is high, such a simplistic scheme can severely under-utilize the available buffer space. This is not an issue for CONNECT routers which only introduce a single cycle of delay.

[1]In addition to user-exposed VCs (a.k.a. message classes), NoCs often also employ a number of internal VCs within each router to improve network performance. Such VCs are typically only visible and allocated within the network and are not exposed to the network clients. To reduce hardware cost, CONNECT exposes all VC handling to the network clients. As a result, applications seeking to use additional VCs for performance improvements need to manually handle VC allocation at the NoC endpoints.

5. EVALUATION AND RESULTS

To demonstrate the effectiveness of CONNECT's FPGA-centric design choices, we first compare FPGA synthesis results and network performance of two instances of a CONNECT-based NoC against a high-quality state-of-the-art ASIC-oriented NoC [29], both before and after modifying their ASIC-style RTL for efficient FPGA synthesis while maintaining bit and cycle accuracy to their original RTL. To further evaluate the CONNECT NoC architecture and highlight its flexibility and extensive design space coverage, we examine multiple router configurations, as well as entire CONNECT networks and report FPGA synthesis results and network performance results. Synthesis results include FPGA resource usage and clock frequency estimates for a moderately sized Xilinx Virtex-6 LX240T FPGA (part xc6vlx240t, speed grade -1) and a large Xilinx Virtex-6 LX760 FPGA (part xc6vlx760, speed grade -2). To assess network performance, we drive the NoCs with various traffic patterns and show the resulting load-delay curves.

5.1 Methodology

FPGA synthesis results are obtained using Xilinx XST 13.1i. Network performance results are collected through cycle-accurate RTL-level simulations. Each load-delay curve is generated through multiple simulations that sweep a range of different network loads. For each simulation, a traffic generator feeds traffic traces through the NoC endpoints and collects statistics as the packets are drained from the network. For each experiment the simulator is initially warmed up for 100,000 cycles, after which delay measurements are collected for 1,000,000 cycles. The duration of warmup and measurement periods were empirically set to be long enough to ensure that the reported metrics had stabilized.

5.2 Comparing to ASIC State-Of-The-Art

To put the FPGA hardware cost and network performance of CONNECT into perspective, we compare it against publicly available RTL of a high-quality state-of-the-art VC-based router [29], which we will refer to as SOTA. This router is written in highly-parameterized Verilog and is modeled after the VC-based router described in [8]. It employs a 3-stage pipeline and supports many advanced features, that are not present or not applicable in CONNECT, such as a larger collection of allocators or adaptive routing. The router supports single or multi-dimensional mesh and torus topologies, as well as the flattened butterfly topology [18].

In the presentation below, we first compare FPGA synthesis results for different configurations of a single router in isolation. We then compare a 4x4 mesh network built using CONNECT routers against a similarly configured 4x4 mesh composed of SOTA routers. Our comparison includes FPGA synthesis results, as well as network performance results under synthetic traffic patterns.

Router comparison. Since the original SOTA router RTL is ASIC-oriented and was thus not optimized for FPGA synthesis, to make the comparison more fair, we modified the SOTA router RTL by applying RTL coding discipline suitable for FPGA synthesis. In particular, our changes only affect storage elements; we ensured that all register files properly mapped to Distributed RAM, instead of discrete registers or Block RAM.

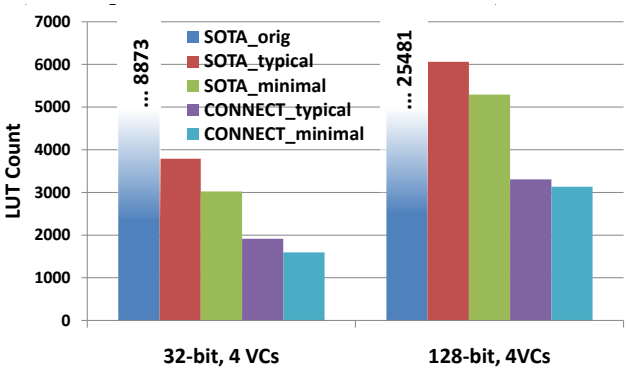

Figure 2: FPGA cost of SOTA and CONNECT 5-port router variants.

The bar graph in Figure 2 shows the difference in FPGA hardware cost in terms of the number of LUTs for a 32-bit 4-VC and a 128-bit 4-VC CONNECT and SOTA router. Both routers have 5 input and output ports ,which corresponds to the typical configuration used in mesh or torus topologies. In all cases we configure the CONNECT and SOTA routers to be as similar as possible, including buffer depths, link widths, allocator type and number of VCs. For each design we examine two router variations: 1)typical, which uses SOTA's default parameter settings (for any remaining parameters) and 2) minimal, which picks those parameter settings that absolutely minimize hardware cost. Even though the minimal configurations do not necessarily constitute realistic design points, they provide a sense of the absolute lower-bound in terms of SOTA's hardware cost. We also include the results for the original RTL (SOTA_orig), before applying the above-mentioned FPGA coding style changes.

In the case of CONNECT, the only change between typical and minimal is the allocator choice; in minimal we use a variation of a separable output-first allocator that minimizes LUT count at the cost of lower router performance. In the case of the SOTA router, for the minimal configuration, we performed a sweep of all router parameters and picked the combination that yielded the lowest FPGA hardware cost.

First of all, it is interesting to note the vast reduction in the SOTA hardware cost from just applying proper RTL coding discipline, which ranges from 57% to 76%. After correcting for FPGA-coding style, the SOTA routers are still almost twice as costly compared to the equivalent CONNECT routers in terms of LUT usage. For typical configurations, CONNECT routers use between 40% to 50% fewer LUTs, for typical and minimal configurations respectively. It is worth mentioning that a 128-bit wide CONNECT router uses approximately the same amount or even fewer FPGA resources than its 32-bit SOTA counterpart. As we will see later, these "saved" FPGA resources can be used to build a more aggressive CONNECT NoC that uses aproximately the same FPGA resources as a SOTA NoC, but can offer 3-4x higher network performance.

Mesh Network Comparison. To compare the two designs at the network level we use CONNECT and SOTA routers to build three 4x4 mesh networks with 4 VCs, 8-entry flit buffers and seperable allocators. Table 1 shows synthesis results for the resulting networks targeting Xilinx Virtex-6 LX240T and LX760 FPGAs. When configured with the same 32-bit flit width (SOTA and CONNECT_32), the SOTA network is more than twice as costly in terms of LUT usage, but can achieve approximately 50% higher clock frequency. The potential performance loss due to the maximum clock frequency difference can be easily regained by adapting other CONNECT NoC parameters, such as flit width. To demonstrate this, we also include results for a 128-bit wide version of the CONNECT mesh NoC (CONNECT_128), that uses aproximately the same FPGA resources as its SOTA 32-bit counterpart, but offers three to four times higher network performance.

We should point out that the endpoints in a SOTA network are required to precompute routing information for each packet injected into the network. This incurs a small additional hardware cost that is, however, excluded from our reported results, since it affects the network endpoints and not the network itself.

	Xilinx LX240T		Xilinx LX760	
4x4 Mesh w/ 4VCs	**%LUTs**	**MHz**	**%LUTs**	**MHz**
SOTA (32-bit)	36%	158	12%	181
CONNECT_32 (32-bit)	15%	101	5%	113
CONNECT_128 (128-bit)	36%	98	12%	113

Table 1: Synthesis Results for CONNECT and SOTA Mesh Network.

To compare the example CONNECT and SOTA NoCs in terms of network performance, we examine the load-delay behavior of the networks under uniform random traffic, where the destination for each packet is randomly selected, and an instance of the unbalanced traffic pattern, where a fraction of the generated packets determined by the *Unbalance Factor* are local and are sent to neighboring nodes. In our experiments we set the *Unbalance Factor* to 90%, which represents a system where nodes communicate heavily with their neighbors and occasionally also send packets to other randomly chosen nodes in the system. We size packets to half the flit buffer depth, which corresponds to 4 flits, and pick the VC randomly for each injected packet.

Since NoCs are typically used within larger systems hosted on an FPGA, their clock frequency is oftentimes dictated by other components and constraints in the system. This is especially true in FPGA environments, where the clock frequency gains of running each component at its maximum frequency are likely to be outweighed by the added synchronization latency increase and hardware cost. To properly capture this potential frequency disparity, we report network performance results for both 1) assuming the studied NoCs are all running at a common clock frequency of 100MHz, possibly dictated by some other system component, and 2) assuming each NoC is running in isolation and can be precisely clocked at its maximum frequency, which provides an upper bound for performance.

All packets in the SOTA network require an additional header flit that carries control information, which brings the total number of flits per packet to five; one header flit and four data flits. CONNECT does not require this extra header flit; instead it carries flit control information "on the side" using wider links. Since the header overhead can change depending on the specific packet size, we also report the SOTA_raw curve, which eliminates SOTA's header overhead and captures raw flit throughput, providing an upper bound for the fully amortized performance of SOTA.

Figures 3 and 4 present load-delay curves for the CONNECT and SOTA networks all running at the same frequency of 100MHz under the two traffic patterns introduced above. Interestingly, when operating at the same frequency, even CONNECT_32, which shares the same 32 bit flit width

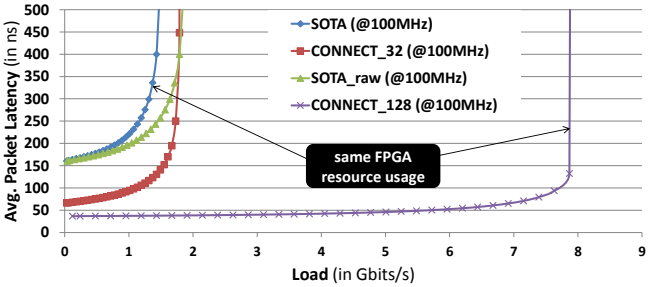

Figure 3: Load-Delay Curves for SOTA & CONNECT @ 100MHz with Unif. Random Traffic.

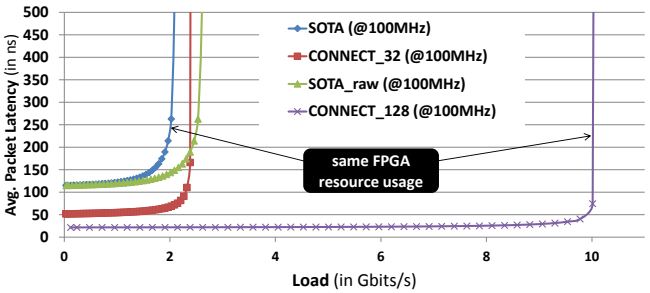

Figure 4: Load-Delay Curves for SOTA & CONNECT @ 100MHz with Unbalanced 90% Traffic.

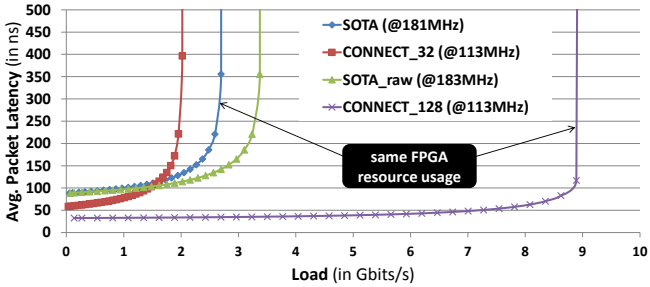

Figure 5: Load-Delay Curves for SOTA & CONNECT @ Max. Freq. with Unif. Random Traffic.

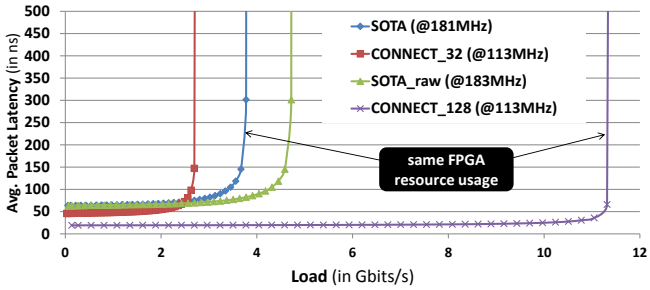

Figure 6: Load-Delay Curves for SOTA & CONNECT @ Max. Freq. with Unbalanced 90% Traffic.

with SOTA and occupies about half the FPGA resources, yields better network performance, both in terms of latency and saturation throughput. This is due to the additional header flit overhead on the SOTA network. When compared to SOTA_raw, which excludes the header overhead, CONNECT's performance is comparable to SOTA.

Figures 5 and 6 show the equivalent results when each network is running at its maximum clock frequency. In this case CONNECT_32 still offers significantly lower latency (59% lower) for the majority of operating loads, because of its reduced pipeline stages. At higher loads, as the load approaches the saturation point, SOTA outperforms CONNECT_32, due to its higher clock frequency.

However, notice that in all cases CONNECT_128, which occupies about the same FPGA resources as SOTA, can easily outperform all other networks by a wide margin across all traffic patterns and regardless of frequency adjustments; it consistently offers three to four times higher saturation throughput and more than three times lower latency.

Overall, for comparable configurations, CONNECT can offer similar network performance to SOTA with consistently lower latency at approximately half the FPGA resource usage. Alternatively, for the same FPGA resource budget, CONNECT can offer much higher performance than SOTA – three to four times higher saturation throughput and more than three times lower latency. In all cases the unbalanced traffic pattern, which consists of mostly local traffic, increases the saturation throughput across all networks, which is expected for the mesh topology that performs better under increased neighbor-to-neighbor traffic.

5.3 CONNECT Router Synthesis Results

As mentioned earlier, in addition to the mesh topology studied above, CONNECT supports a variety of different router and network configurations to better suit the diverse communication needs of emerging SoCs. To get a better feel for the cost and performance of different CONNECT-based routers, Table 2 shows FPGA resource usage and clock frequency synthesis results for a range of different CONNECT router configurations targeting a Xilinx Virtex-6 LX760 FPGA. All reported results are for a single router to be used in a 64-node network. As expected, increasing the number of router ports, VCs, flit width or buffer depth all contribute to higher LUT counts and negatively impact clock frequency.

The number of router ports has the largest impact in hardware cost, followed by the number of VCs. Both of these parameters influence the buffering requirements, as well as the allocation and flow control logic. Changes in flit width and buffer depth only affect buffering requirements and as such have a lower relative impact. It is interesting to note that buffer depth affects LUT count in a more unpredictable manner due to quantization effects of Distributed RAMs; intuitively, wider memory arrays scale smoothly in terms of LUT cost, while taller memory arrays scale in a more abrupt step-wise manner.

5.4 CONNECT Network Synthesis Results

In this section, to demonstrate the flexibility and extensive design space coverage of CONNECT, we examine a few different examples of CONNECT-based networks in terms of hardware cost and network performance. Table 3 lists the selected network configurations, which range from a low-cost low-performance ring network (Ring16) all the way to a high-performance fully-connected network (HighRadix16), as well as an indirect multistage network (FatTree16). The number next to each network name indicates the number of supported network endpoints. The HighRadix16 network corresponds to a network with eight fully connected routers, where each router is shared by two network endpoints, i.e. with a concentration factor of two.

Table 4 shows synthesis results for these eight sample network configurations targeting a moderately sized Xilinx Virtex-6 LX240T FPGA, as well as a larger Xilinx Virtex-6

Flit Width		32 bits								128 bits							
Num. VCs		2 VCs				4 VCs				2 VCs				4 VCs			
Buf. Depth		4	8	16	32	4	8	16	32	4	8	16	32	4	8	16	32
2 In/Out	LUTs	242	292	340	485	373	427	564	952	562	612	659	936	693	739	1020	1861
Ports	MHz	306	284	247	218	260	240	217	195	306	283	245	221	260	232	219	193
4 In/Out	LUTs	688	813	893	1236	938	1137	1454	2408	1424	1550	1629	2230	1672	1872	2460	4310
Ports	MHz	180	183	154	143	169	167	147	139	180	183	154	143	166	165	147	139
6 In/Out	LUTs	1893	2005	2161	2812	2000	2351	2861	4399	3705	3839	4018	4987	4055	4442	5364	8439
Ports	MHz	150	142	130	126	122	123	115	109	149	140	127	122	122	124	117	110
8 In/Out	LUTs	3012	3171	3439	4149	3767	3953	4849	6565	5368	5544	5780	7035	6134	6322	7753	11280
Ports	MHz	117	114	103	101	107	102	103	95	117	114	102	101	107	102	103	94

Table 2: Synthesis results for various CONNECT router configurations.

Network	Routers	Ports/Router	VCs	Width
Ring64	64	2	4	128
DoubleRing16	16	3	4	32
DoubleRing32	32	3	2	32
FatTree16	20	4	2	32
Mesh16	16	5	4	32
Torus16	16	5	2	64
HighRadix8	8	8	2	32
HighRadix16	8	9	2	32

Table 3: Sample Network Configurations.

LX760 FPGA. For each network we report the LUT usage as a percentage of the total amount of LUTs on the respective FPGA, as well as synthesis clock frequency.

The synthesis results indicate that all networks easily fit within both FPGAs, with plenty of room to spare for placing many other pieces of user logic. In fact, when targeting the LX760 FPGA, all networks occupy less than 10% of the available LUTs. Finally, it is also worth mentioning that CONNECT networks do not occupy even a single Block RAM, which leaves a great amount of on-chip storage available to other FPGA-resident components.

	Xilinx LX240T		Xilinx LX760	
Network	%LUTs	MHz	%LUTs	MHz
Ring64	30%	175	9%	200
DoubleRing16	9%	139	3%	158
DoubleRing32	11%	146	4%	169
FatTree16	12%	117	4%	143
Mesh16	15%	101	5%	113
Torus16	25%	91	8%	100
HighRadix8	20%	73	5%	76
HighRadix16	28%	67	9%	75

Table 4: Synthesis results for sample networks.

5.5 CONNECT Network Performance

In this section, we focus on a subset of four networks (DoubleRing16, Mesh16, FatTree16 and HighRadix16), that all support 16 network clients, and as such would be interchangable when used as the interconnect within an FPGA-based system. To study network performance we use the same two traffic patterns described earlier, uniform random and unbalanced (with an UnbalanceFactor of 90%), which can be thought of as corresponding to two different classes of FPGA applications, each with different degrees of local communication. Once again, we size packets to half the flit buffer depth, which corresponds to 4 flits, and pick the VC randomly for each injected packet.

Figure 7 shows the load-delay curves for the four selected networks under uniform random traffic. Given the low bisection bandwidth of the double ring topology, the DoubleRing16 network is the first to saturate at a load of approximately 30%. The Mesh16 and FatTree16 networks can sustain much higher loads before they saturate at roughly 55% load. This can be both attributed to the higher connectivity and bisection bandwidth of the mesh and fat tree topology, as well as the higher number of VCs in the case of the Mesh16 network (4 instead of 2). Finally, as expected, the HighRadix16 network achieves the highest performance, offering lower latency across all loads and saturating at a load of 70%. This should come as no surprise, given that the HighRadix16 network is fully-connected (maintains single-hop point-to-point links between all routers in the network), which means that the only source for loss of performance is output contention [8].

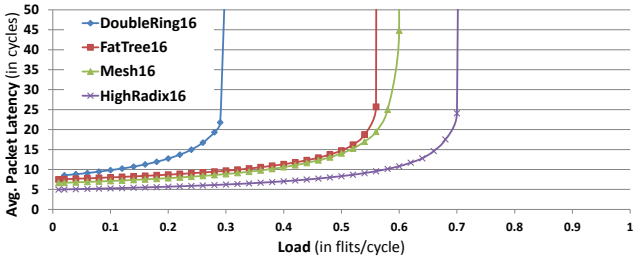

Figure 7: Load-Delay Curves for CONNECT Networks with Uniform Random Traffic.

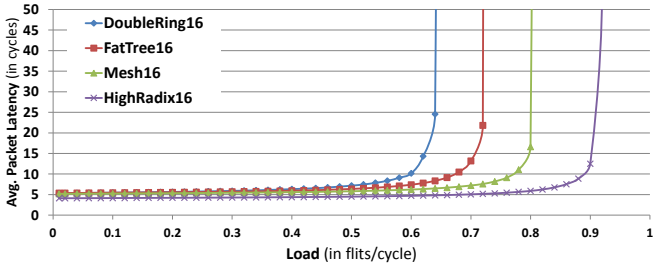

Figure 8: Load-Delay Curves for CONNECT Networks with Unbalanced 90% Traffic.

Figure 8 shows the equivalent load-delay curves under the unbalanced traffic pattern, which favors mostly neighbor-to-neighbor communication. As expected, the increased locality allows all networks to perform better, with the DoubleRing16 network experiencing the largest relative performance gains. In fact, under unbalanced traffic the DoubleRing16 network outperfoms the more FPGA resource intensive Mesh16 and FatTree16 networks.

Even though these results are mainly presented to demonstrate the flexibility of CONNECT and, as such, are not ex-

haustive or might omit other implementation details, such as frequency-related constraints, they do show that NoC performance can be highly dependent on network topology and configuration, but more importantly on the specific traffic patterns and requirements of an application. This observation is especially important in the context of FPGAs, where NoC topology and configuration can be easily adapted to suite the requirements of the given application.

6. RELATED WORK

Although there has been extensive previous work that combines FPGAs and NoCs, a large part of this work either examines the use of FPGAs for efficient NoC modeling [25, 31] or simply presents a larger FPGA-based design that also happens to include an application-specific ad-hoc NoC [30]. There is only a limited amount of previous studies that focus on FPGA-tailored NoC architectures to support SoC emulation or other FPGA applications.

In the context of FPGA-oriented NoC architectures, NoCem [11, 12] presents a very simple router block that can be used to compose larger networks on FPGAs. Compared to CONNECT it lacks more advanced features, such as support for virtual channels or selectable allocation and flow control schemes. More importantly, for equivalent networks, compared to CONNECT, it appears to incur a much higher cost after a rough normalization for the differeneces in the FPGAs used. (A more exact quantitiatve comparision is hard because the NoCem synthesis results were obtained on the much older Virtex-2 FPGAs.)

PNoC [15] is an interesting proposal for building lightweight networks to support FPGA-based applications. Even though PNoC can also yield low-cost FPGA-friendly networks, the fundamental difference compared to CONNECT is that it can only be used to create circuit-switched networks, instead of packet-based. Circuit-switched networks can be useful for the classes of FPGA applications that have structured non-conflicting traffic patterns and are willing to tolerate the additional setup and tear-down delay and connection management associated with circuit-switched networks.

In the context of FPGA-related NoC studies, previous work has developed analytical models for predicting NoC performance on FPGAs [21], as well as examined the effect of various NoC parameters, such as topology and number of nodes, on the performance of an FPGA-resident multiprocessor system [20]. Morever, previous work has also studied the trade-offs between FPGA implementations of packet-switched and time-multiplexed networks [17].

Previous work has also looked at leveraging or modifying the FPGA configuration circuitry to build efficient FPGA-based NoCs. Metawire [28] overlays a communication network on top of the configuration network in a Virtex-4 FPGA. However, such an approach yields lower performance and is inevitably tied to the specific FPGA architecture and vendor. Francis et al. [10] propose replacing the statically configured FPGA wiring with time-division multiplexed wiring that can enable the implementation of efficient low-overhead NoCs for future FPGAs.

Finally, there is also a large body of commercial interconnect approaches, such as Spidergon STNoC [6], ARM's AMBA [2] or even FPGA-specific approaches, such as the CoreConnect [16] PLB and OPB buses, commonly found in Xilinx FPGAs, or Altera's Qsys [1]. CONNECT offers a more lightweight, fine-grain and flexible FPGA-tailored solution for building soft NoCs, that can synergistically coexist with the above approaches to cover the diverse communication needs of emerging SOCs.

7. FUTURE DIRECTIONS

As FPGAs continue to gain more traction as computing and SoC platforms, the role of NoCs will inevitably become more central in future FPGA-based systems. This can happen both in the form of efficient flexible architectures tailored for soft-logic implementations, such as the one presented in this paper, but can potentially also trigger the transition to future FPGA devices with fixed hardened NoCs.

Our immediate plan is to release a current version of CONNECT in the form of a web-based flexible RTL NoC generator.i Moreover, we are interested in experimenting with a 2-stage pipeline router design that will yield improved clock frequency while still keeping FPGA resource usage at a minimum. Another interesting future direction is to study how we can apply the FPGA-oriented design guidelines and disciplines used in CONNECT to other common FPGA components in order to improve their efficiency.

As a longer term goal, we are interested in examining the form of future FPGA devices and their underlying switching fabric. Other researches have suggested that future FPGAs will consist of islands of reconfigurable logic connected through a dedicated high-performance NoC [27], which raises a few fundamental interesting questions: What will this NoC look like and how will its architecture and implementation be affected by the use of silicon interposers [23] in modern FPGAs? Which parts does it make sense to implement in hard-logic and which parts should be left to be implemented in soft-logic?

8. CONCLUSION

In this paper, we presented CONNECT, a flexible and efficient approach for building NoCs for FPGA-based systems. CONNECT embodies a set of design guidelines and disciplines that try to make the most efficient use of the FPGA substrate and in many cases go against ASIC-driven conventional wisdom in NoC design. We compare a high-quality state-of-the-art NoC design against our design both in terms of FPGA cost, as well as network performance for a similarly configured 4x4 mesh NoC. Across a wide range of configuration parameters, we find that CONNECT consistently offers lower latencies and can achieve comparable network performance at one-half the FPGA resource cost; or alternatively, three to four times higher network performance at approximately the same FPGA resource cost. Moreover, to demonstrate the flexibility and extensive design space coverage of CONNECT we report synthesis and network performance results for a wide range of router configurations and a variety of diverse CONNECT-based networks.

9. ACKNOWLEDGMENTS

Funding for this work was provided by NSF CCF-0811702 and NSF CCF-1012851. We thank the anonymous reviewers and Carl Ebeling for their useful comments that helped shape this paper. We thank the members of the Computer Architecture Lab at Carnegie Mellon (CALCM) and Daniel Becker from the Stanford CVA group for their comments and feedback. We thank Xilinx for their FPGA and tool donations and Bluespec for their tool donations and support.

Finally, we also thank Derek Chiou for organizing the 2011 MEMOCODE Contest that sparked the inspiration for this work.

10. REFERENCES

[1] Altera. Qsys System Integration Tool. `http://www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qts-qsys.html`.

[2] ARM. AMBA Open Specifications. `http://www.arm.com/products/solutions/AMBAHomePage.html`.

[3] Bluespec, Inc. Bluespec System Verilog. `http://www.bluespec.com/products/bsc.htm`.

[4] E. S. Chung, J. C. Hoe, and K. Mai. CoRAM: An In-Fabric Memory Abstraction for FPGA-based Computing. In *Nineteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2011.

[5] E. S. Chung, M. K. Papamichael, E. Nurvitadhi, J. C. Hoe, and K. Mai. ProtoFlex: Towards Scalable, FullSystem Multiprocessor Simulations Using FPGAs. *ACM Transactions on Reconfigurable Technology and Systems*, 2009.

[6] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra. Spidergon: A Novel On-Chip Communication Network. In *International Symposium on System-on-Chip*, 2004.

[7] W. Dally and C. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, 1987.

[8] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[9] P. Del valle, D. Atienza, I. Magan, J. Flores, E. Perez, J. Mendias, L. Benini, and G. Micheli. A Complete Multi-Processor System-on-Chip FPGA-Based Emulation Framework. In *IFIP International Conference on Very Large Scale Integration*, 2006.

[10] R. Francis, S. Moore, and R. Mullins. A Network of Time-Division Multiplexed Wiring for FPGAs. In *Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2008.

[11] G. Schelle and D. Grunwald. Onchip Interconnect Exploration for Multicore Processors Utilizing FPGAs. In *2nd Workshop on Architecture Research using FPGA Platforms (WARFP)*, 2006.

[12] G. Schelle and D. Grunwald. Exploring FPGA Network on Chip Implementations Across Various Application and Network Loads. In *International Conference on Field Programmable Logic and Applications (FPL)*, 2008.

[13] M. Gerla and L. Kleinrock. Flow Control: A Comparative Survey. *IEEE Transactions on Communications*, 1980.

[14] S. J. Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of the ACM symposium on Communications architectures & protocols*, SIGCOMM, 1990.

[15] C. Hilton and B. Nelson. PNoC: A Flexible Circuit-Switched NoC for FPGA-based Systems. *IEE Proceedings Computers and Digital Techniques*, 2006.

[16] IBM. The Coreconnect Bus Architecture. `https://www-01.ibm.com/chips/techlib/techlib.nsf/products/CoreConnect_Bus_Architecture`, 1999.

[17] N. Kapre, N. Mehta, M. deLorimier, R. Rubin, H. Barnor, M. Wilson, M. Wrighton, and A. DeHon. Packet Switched vs. Time Multiplexed FPGA Overlay Networks. In *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2006.

[18] J. Kim, J. Balfour, and W. Dally. Flattened Butterfly Topology for On-Chip Networks. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2007.

[19] I. Kuon and J. Rose. Measuring the Gap Between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007.

[20] J. Lee and L. Shannon. The Effect of Node Size, Heterogeneity, and Network Size on FPGA based NoCs. In *International Conference on Field-Programmable Technology (FPT)*, 2009.

[21] J. Lee and L. Shannon. Predicting the Performance of Application-Specific NoCs Implemented on FPGAs. In *Nineteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2011.

[22] M. Karo; M. Hluchyj; S. Morgan. Input Versus Output Queuing on a Space-Division Packet Switch. In *IEEE Transactions on Communications*, 1987.

[23] M. Matsuo, N. Hayasaka, K. Okumura, E. Hosomi, and C. Takubo. Silicon interposer technology for high-density package. In *50th Electronic Components and Technology Conference*, 2000.

[24] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and P. Maji. An analytical performance model for the Spidergon NoC with virtual channels. *Journal of Systems Architecture*, 2010.

[25] M. K. Papamichael, J. C. Hoe, and O. Mutlu. FIST: A Fast, Lightweight, FPGA-Friendly Packet Latency Estimator for NoC Modeling in Full-System Simulations. In *Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, 2011.

[26] M. Saldana, L. Shannon, J. S. Yue, S. Bian, J. Craig, and P. Chow. Routability of Network Topologies in FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2007.

[27] H. Schmit. Extra-dimensional Island-Style FPGAs. In *International Conference on Field Programmable Logic and Applications (FPL)*, 2003.

[28] Shelburne, M.; Patterson, C.; Athanas, P.; Jones, M.; Martin, B.; Fong, R. MetaWire: Using FPGA Configuration Circuitry to Emulate a Network-on-Chip. In *International Conference on Field Programmable Logic and Applications (FPL)*, 2008.

[29] Stanford Concurrent VLSI Architecture Group. Open Source Network-on-Chip Router RTL. `https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router`.

[30] L. Thuan. NoC Prototyping on FPGAs: A Case Study Using an Image Processing Benchmark. In *International Conference on Electro/Information Technology*, 2009.

[31] D. Wang, N. Jerger, and J. Steffan. DART: A programmable architecture for NoC simulation on FPGAs. In *Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, 2011.