On-Chip Network Exploration and Synthesis

by

Chia-Hsin Chen

B.S. in Electrical Engineering
National Taiwan University, 2007

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

Author .....
Department of Electrical Engineering and Computer Science
Dec 8, 2011

Certified by .....
Li-Shiuan Peh
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

On-Chip Network Exploration and Synthesis

by

Chia-Hsin Chen

Submitted to the Department of Electrical Engineering and Computer Science
on Dec 8, 2011, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

As CMOS technology improves, the trend of processor designs has gone towards multi-core architectures. Networks-on-Chips (NoCs) have become popular on-chip interconnect fabrics that connect the ever-increasing cores because of their ability to provide high-bandwidth. However, as the number of cores keeps increasing, the end-to-end packet latency and the total network power begin to pose tight constraints on NoC designs. In this thesis, we studied architecture proposals designed to tackle this latency and power budget issue. We also studied the impact of applying advanced circuit techniques to these architecture proposals and how to implement these techniques while realizing a NoC design.

The thesis begins with an evaluation of physical express topologies and the virtual express topologies that enable the bypassing of intermediate router pipelines. The bypassing of pipeline stages help reduce both end-to-end latency and power consumption since fewer resources are used. We observed that both topologies have similar low-traffic-load latencies and that virtual express topologies result in higher throughput and are more robust across traffic patterns. Physical express topologies, however, deliver a better throughput/watt and can leverage the low-swing link circuits to lower the latency and increase the throughput.

Next, then we identified that crossbars, in addition to links, can obtain benefit from the low-swing circuit techiques. We thus developed a layout generation tool for low-swing crossbars and links due to the inability of the existing tools for physical designs to generate these low-swing circuits automatically. The generated crossbars and links using our tool showed 50% energy saving compared to the full-swing synthesized counterpart. We also demonstrated a case study with a router synthesized with the generated crossbar and links.

Thesis Supervisor: Li-Shiuan Peh
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to express my sincere gratitute to my advisor, Prof. Li-Shiuan Peh. It has been an honor to be her student. Throughout the past three year, she has guided me to work on various projects and to collaborate with teams from various fields that has broadened my mind and learned about various aspects of research. The joy and enthusiasm she has on her research was contagious and influenced me. I thank her for her guidance and am looking forward to more fun years with her to pursue my PhD.

I would also like to thank my mentors, Niket Agarwal and Tushar Krishna, for spending their quality time patiently guiding me into this field and answering all my technical and non-technical questions. They are also co-authors in the projects described in the thesis. I would also like to acknowledge Sunghyun who was a co-author of one of the projects described in this thesis. I thank all my group colleages, including Bin, Kostas, Manos and Bhavya, for contributing to both my personal and professional time at MIT and Princeton.

Studying abroad to the US and being away from home are tough and lonely. I thank Ting-Jung, Yen-Kuan, Dawsen, Yin-Wen, Hung-Wen and Hsin-Jung for their accompany and help in forming a second home here in the US. I had so much fun with them.

Lastly, my deepest gratitude goes to my family for their full support throughout my life. I thank my parents for encouraging and motivating me to come to the US and pursue a MS-PhD. I also thank my brother for his constant support in discussing any questions and problems that I have and encounter.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To overcome the ever-increasing power consumption and diminishing returns in the performance of uni-core processor architectures, computer architects embraced multi-core architectures to utilize the growing number of transistors provided by each technology generation. Traditional electrical buses do not scale with the number of cores and fail to provide scalable bandwidth with large core counts. Hence, using a packet-switched networks-on-chip (NoC) as the communication fabric becomes the trend while designing next-generation processors. For example, IBM's Cell Broadband Engine [1] and Intel's Larrabee [2] use a ring topology, and Intel's 80-core [3] and 48-core [4] research prototypes use a mesh topology. Academic research chips such as MIT's RAW [5] and UT Austin's TRIPS [6] also use mesh topology as the on-chip communication fabric. Using a packet-switched network enables high bandwidth by sharing network links across multiple packet flows, but it comes with significant power and area overhead because it requires a complex router at every node. It also comes with significant delay since packets need to compete for resources on a hop-by-hop basis as they traverse through routers along the path.

1.1 Contribution

In this thesis, we explored two main classes of topologies designed to improve the packet-switched network performance while reducing the total network power consumption. We also developed a tool that is designed to fit circuits with low-swing

signaling techniques within the standard CAD flow so that the low-swing circuits can be readily used in large-scale designs.

1.1.1 Physical vs. Virtual Express Topologies with Low-Swing Links for Future Many-core NoCs

Selecting an appropriate topology is one of the most critical decisions in the design of on-chip networks; it impacts the zero-load latency and sustainable bandwidth, and also influences the power consumption of the network. Most existing on-chip networks utilize a 2-D mesh topology [7, 3, 8], as meshes have lower design complexity and map well to the 2-D chip substrate. However, they are energy inefficient because the high network diameter leads to extra router hops to reach the destination, and router energy is high relative to link energy [3]. This poses serious scalability concerns for such topologies as node count increases in the future. To tackle the scalability issues of 2-D mesh, various express topologies have been proposed in the literature. One set of proposals employ long physical links between non-local routers to reduce the effective network diameter [9, 10, 11]. We will refer to these topologies as *physical express topologies.* Another set of proposals use various techniques to opportunistically bypass router pipelines at intermediate network hops and thus save on network latency and power [12, 13, 14]. We will refer to these techniques as *virtual express topologies.*

Both physical and virtual express topologies have their advantages and disadvantages. Physical express topologies reduce the network diameter, thereby saving the latency and power due to bypassing of intermediate router hops. To accomplish the physical bypass paths, however, extra router ports, larger crossbars and extra physical channels are required. Extra router ports and larger crossbars lead to higher router area and power. While on-chip wires are relatively abundant, use of large number of dedicated point-to-point links leads to a large area footprint and low channel utilization. Virtual express topologies have the advantage that they do not use extra long physical links. However, the bypass of routers in such designs is opportunistic and not guaranteed. The virtual router bypass also differs from physical bypass in that the bypassing flits still have to multiplex through the router crossbar and output link, while in a physical bypass, all intermediate routers and links are bypassed completely.

The long links in the physical express topologies can also leverage some of the recent innovative low-swing interconnect proposals. Specifically, capacitively driven low-swing interconnects (CDLSI) [15] have the potential to save significant energy while also providing modest latency reductions. CDLSI links can be used as single-cycle multi-hop express links that provide further latency savings since long hops can be covered in a single cycle and the power consumption would also be less, owing to the low-swing nature of these links. The low power of the CDLSI links can also be exploited by all the topologies for connecting neighboring routers.

Although there are clear trade-offs between various physical and virtual express topologies, to the best of our knowledge, there has been no work comparing these various topology alternatives for many-core network on-chips. In this work, we compare a particular physical express topology (express physical channels (EPC) based on express cubes [9]), to a virtual express topology proposal (express virtual channels (EVC) [12]) for large node count on-chip networks. We present energy, area, latency and throughput results for a 256 node configuration with synthetic network traffic. We observe that both EPC and EVC help in lowering low-load latency. However, while the EVC network is more robust across traffic patterns, and offers a higher throughput than baseline for most cases, the EPC network is more traffic dependent and bisection bandwidth dependent. If we compare the performance-per-watt, however, EPC with CDLSI links turns out to be the best.

The main contributions of this work are as follows.

- We present a detailed characterization of how CDLSI links could be leveraged for long and short hop links to be used in various express topologies.
- We present a detailed comparison of physical and virtual express topologies for large node-count systems under different design constraints (bisection bandwidth, router area, and power).

1.1.2 A Low-Swing Crossbars and Links Generator for Low-Power Networks-on-Chip

A 1-bit $N \times M$ crossbar consists of $N \times M$ interconnected wires that are controlled by switches and enable any port to connect to any other port. The outputs of a

crossbar connect to links that then connect to an IP block or a router. The crossbar and links thus together form the datapath of a NoC. This datapath has been found to dominate the NoC power consumption. Fabricated chips from academia, such as MIT's RAW [5] and UT Austin's TRIPS [6], use RTL synthesis to generate the datapath, and the ratio of datapath power consumption and the total on-chip network power consumption are reported to be 69% and 64%, respectively. Intel TERAFLOPs [3] uses a custom-designed double-pumped crossbar with a location based channel driver to reduce the channel area and peak channel driver current [16] and is thus able to reduce datapath power to 32% of the total on-chip network power. Other circuit techniques that have been proposed to reduce this power consumption involve dividing the crossbar wires into multiple segments and partially activating selected segments [17, 18] based on the input and output ports. These circuit techniques present only the capacitance between the input and output port, and disable/reduce other capacitances. They are thus successful in reducing wasteful power consumption. However, they still require complete charging/discharging of the long wires from the input port to the output port and the core-core links, which are significant power consumers.

Low-swing signaling techniques can help mitigate the wire power consumption. The energy benefits of low-swing signaling have been demonstrated on-chip from 10mm equalized global wires [19], through 1-2mm core-to-core links [20], to less than 1mm within crossbars [21, 22, 23]. However, such low-swing signaling circuits, which can be viewed as analog circuits, require full custom design, resulting in substantial design time overhead. Circuit designers have to manually design schematic/netlists, optimize logic gates for each timing path, and size individual transistors. Moreover, layout engineers have to manually place all the transistors and route their nets with careful consideration of circuit symmetry and noise coupling. This custom design process leads to high development cost, long and uncertain verification timescales, and poor interface to other parts of a many-core chip, which are mostly RTL-based.

In the past, designers faced similar challenges while integrating low-power memory circuits with the VLSI CAD flow, with their sense amplifiers, self-timed circuits and dynamic circuits. Memory compilers, which are now commonplace, have solved the problem and enabled these sophisticated analog circuits to be automatically gen-

erated, subject to variable constraints specified by the users. This paper proposes to similarly automate and generate low-swing signaling circuits as part of the datapath (crossbar and links) of a NoC, thereby integrating such circuits within the CAD flow of many-core chips, enabling their broad adoption.

Since crossbars and links are such an essential component of on-chip networks, there have been efforts in the past to automate their generation. Sredojevic and Stojanovic [24] presented a framework for design-space exploration of equalized links, and a tool that generates an optimized transistor schematic. However, they rely on custom-design for the actual layout. ARM AMBA [25], STMicroelectronics STBus [26], Sonics MicroNetworks [27], and IBM CoreConnect [28] are examples of on-chip bus generators; DX-Gt [29] is a crossbar generator; and ×pipes [30] is a network interface, switch and link generator. These tools are aimed at application specific network-on-chip (NoC) component generation, but they all stop at the synthesizable HDL level, i.e. they generate RTL, and then rely on synthesis and place-and-route tools to generate the final design. This is not the most efficient way to design crossbars, as we show later in Section 3.3, highlighting that a synthesized crossbar design consumes significantly more power than a custom low-swing crossbar.

In this work we present a NoC datapath generator, which is the first to integrate low-swing links in an automated manner. It is also the first to generate a noise-robust layout at the same time, embedded within the synthesis flow of a NoC router. Our tool takes a low-swing driver as input and ensures (1) a crosstalk noise-robust routing, (2) supply noise-robust differential signaling, and (3) crosstalk-controlled full-shielded links, in the generated datapath.

The main contributions of this work are as follows.

- It is the first automated generation of noise-robust low-swing links within the crossbar, and between routers.
- It is the first automated layout generation of a crossbar for a user specified number of ports, channel-width, and target frequency.
- It is the first demonstration of a generated low-swing crossbar and link within a fully-synthesized NoC router.

- Our automatically generated low-swing crossbar achieves an energy savings of 50%, at the same targeted frequency of the synthesized crossbar, at 3-4 times the area overhead. Relative to the entire router, the larger footprint of the crossbar is manageable, at just 30% of the overall router area.

1.2 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 evaluates the physical and virtual express topologies and summerizes key insights. Next, Chapter 3 describes the design of the low-swing crossbars and links generators. Finally, Chapter 4 concludes the thesis and describes the future directions that I shall be looking into during my PhD.

Chapter 2

Physical vs. Virtual Express Topologies

In this Chapter, we evaluated two categories of topologies: physical express topology and virtual express topology. These topologies were proposed to mitigate the scalability issue of on-chip networks by bypassing part or all of the pipeline stages in the intermediate routers. This bypassing would lower the low-load latency and power consumption due to less resources used. Physical express topologies achieve deterministic bypassing by using extra links connecting distant routers, whereas virtual express topologies opportunistically bypass pipeline stages using flow control techniques. There are trade-offs between the two topologies. For example, physical express topologies bypass intermediate routers completely and hence eliminate the power consumed at those routers. However, using extra links requires a crossbar with higher radix in the router. On the other hand, virtual express topologies use flow control techniques to bypass part of the pipelines at intermediate routers and do not require extra links that would lead to higher-radix routers; hence, traversing the crossbar costs less power. However, the packets stil have to traverse all intermediate routers. In this Chapter, we compared power, area and performance of these topologies under different design constraints using synthetic network traffic. In addition, recently proposed link designs like capacitively driven low-swing interconnects (CDLSI) can help lower link power and latency and can favor these bypass designs. We also evaulated the impact of using this link design on both topologies.

The rest of the chapter is organized as follows. Section 2.1 provides circuit level details of CDLSI links and its latency, area, and energy properties. Section 2.2 discusses various physical and virtual express topologies proposed in the literature and motivates the need for comparison amongst them. Section 2.3 discusses the evaluation methodology and presents quantitative results, and Section 2.4 summarizes this chapter.

2.1 Capacitively Driven Low-Swing Interconnects

Conventional low-swing interconnects are attractive to NoC designs as they reduce the power consumption by reducing the voltage swing on the wires [31]. But this advantage is usually accompanied by a penalty in latency. Moreover, they usually require a second voltage power supply.

Capacitively Driven Low-Swing Interconnects (CDLSI) were proposed by Ho. et al. in [15]. The design involves driving wires through a series capacitor, and these interconnects have been shown to have excellent energy savings without the degradation of performance. The use of the coupling capacitor results in a low-swing wire without the need of a second power supply. It also extends the wire bandwidth by introducing pre-emphasizing effects and significantly reduces the driver energy consumption.

The differential and twisted wires enable the signals to be sent at a low voltage swing and eliminate the coupling noise. However, this comes at the cost of 2X wiring area. The link area model we used in our experiments can be formulated as $A_{Cu} = (2w_{flit} - 1)lw$ for normal non-differential Cu wires and $A_{CDLSI} = (4w_{flit} - 1)lw$ for CDLSI wires, where A_{Cu} stands for the area of a conventional Cu wire, A_{CDLSI} stands for the area of a CDLSI wire, w_{flit} is the flit-width, l is the link length and w is both the width of the link and the spacing between two wires. The spacing between the links is assumed to be the same as the link-width.

We applied a delay-power optimization that is similar to the approach in [32], to both normal Cu wires and CDLSI wires. Detailed analysis of this approach can be found in [33]. For normal Cu wires, more than half of the total power is consumed by the wire. On the other hand, for CDLSI wires, transceivers consume most of the power. As a result, for a given delay penalty, the impact on power saving in CDLSI

	length	1mm		2mm		4mm	
	BW	D(ps)	E(fJ)	D(ps)	E(fJ)	D(ps)	E(fJ)
CDLSI	1Gbps	108	22.4	215	27	430	61
	2Gbps	108	22.4	215	27	430	61
	3Gbps	108	22.4	194	30.4	430	61
	4Gbps	101	27.3	175	35.8	368	71.5
	5Gbps	94	35.2	162	43.6	330	87.2
Normal	1-10Gbps	108	133	215	265	430	531

Table 2.1: Delay(D) and energy(E) consumption of CDLSI wire and normal Cu wire for different lengths and bandwidths

wires is greater than that in normal Cu wires.

Table 2.1 shows the latency and energy consumption of a CDLSI wire as a function of different wire lengths and bandwidths with 192 nm wire width at 32 nm technology node. The wire width and the wire spacing are chosen to be four times the minimum wire width defined in ITRS [34] to meet the bandwidth requirements in our experiments. The table also shows the numbers for normal Cu wires. The energy consumption of a CDLSI wire is approximately 10x less than that of a normal Cu wire with the same wire length and bandwidth below 3 Gbps. It should be noted that CDLSI wires consume ultra-low energy even as the length of the wire increases. This property enables the use of CDLSI wires as express links for multi-hop communication. For example, the delay for a 4mm long CDLSI wire is 330 ps, that is, it is within two cycles for a 5GHz clock frequency, whereas the it takes three cycles to traverse a normal Cu wire of the same length.

2.2 Express Topologies

An ideal interconnection network between processing cores in CMPs would be point-to-point, where the latency of the messages is equal to just the link latency, and there are no additional penalties due to contention and corresponding arbitration. However, such a network is not scalable, requiring $N-1$ links per node in an N-node network, which will blow up the area footprint. This has led to researchers proposing packet-switched networks with intermediate routers to multiplex the available bandwidth [35]. The router microarchitecture for such a network is described next.

2.2.1 Baseline Router

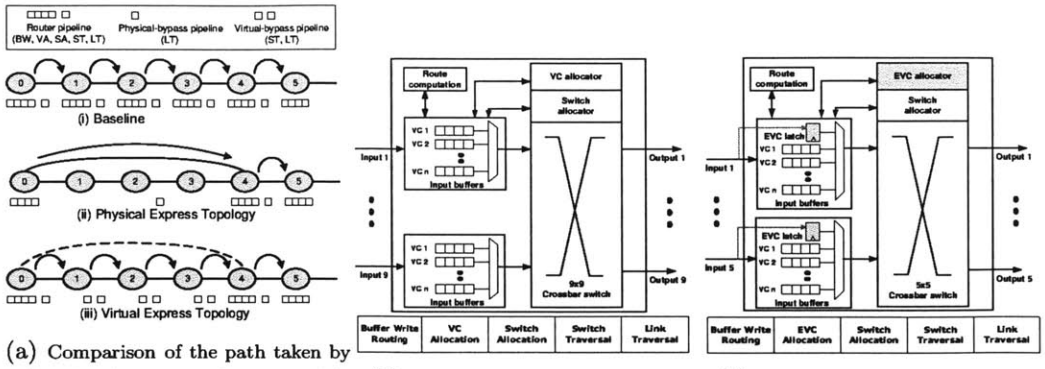
A NoC router typically has a 5-stage pipeline [36], with four stages in the router, and one in the link. In the first stage, the incoming flit gets buffered (BW-Buffer Write), and performs routing for determining the output port to go out from. In the second stage (VA-VC Allocation), the flits at all input ports arbitrate for virtual channels (VCs) at the next router. In the third stage (SA-Switch Allocation), the flits at all input ports that have been assigned output VCs now arbitrate for using the crossbar that will connect this input port to the required output port. In the fourth stage (ST-Switch Traversal), the flits that won the switch allocation traverse the crossbar. Finally, in the fifth stage (LT-Link Traversal), flits coming out of the switch traverse the links till the next routers. A packet can go through VA and SA multiple times depending on the contention, until it succeeds in obtaining an output VC and the switch. Thus each flit entering a router takes multiple cycles before it can make a hop to the next router. This design will be referred to as the *baseline* design in the rest of the chapter.

This solution of having hop-by-hop traversal through multiple intermediate routers is simple and effective. However, spending multiple arbitration cycles in intermediate routers would add unnecessary delay to packets traveling longer distances, especially in future many-core chips. Moreover, the buffering and arbitrations at the routers add to the power consumed. Thus, there have been many proposals that advocate for packets traveling longer distances to *bypass* intermediate routers. We characterize these into two categories.

- *physical express topologies:* adding physical channels to skip intermediate routers.

- *virtual express topologies:* having means to bypass arbitrations and buffering at the intermediate routers.

The basic ideas behind these techniques are shown in Figure 2-1(a), and described next.

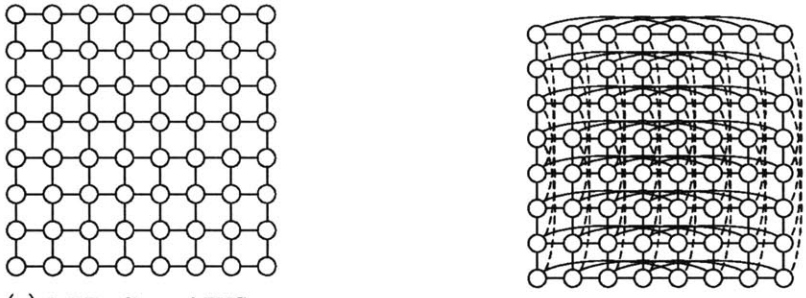


(a) Comparison of the path taken by a flit going from router 0 to router 5 in the Baseline, Physical Express Topologies and Virtual Express Topologies

(b) EPC Router Microarchitecture

(c) EVC Router Microarchitecture

Figure 2-1: Comparison of Express Topologies



(a) 8x8 Baseline and EVC network

(b) 8x8 EPC network

Figure 2-2: Network topology

2.2.2 Physical Express Topologies

Physical express topologies argue for physical express links between far away routers. Non-local packets can traverse these express links (i.e. LT) to avoid getting delayed at intermediate routers and also not add to contention at the ports there. Moreover, the use of express links eliminates the power that these packets would consume at the intermediate routers. The trade-off, however, is that each router now needs to have multiple ports, which in turn requires a bigger crossbar. The area and power of a crossbar increase as a square of the number of ports times the link width, and thus more ports is a potential concern. Moreover, the available metal area on-chip could add a limitation on the number of express links that can be added. Reducing the link width can address some of these concerns but would add serialization latency to the packets.

Express Cubes [9], Flattened Butterfly [10] and MECS [11] are examples of physical express topologies. Express Cubes was a proposal for the multi-chassis shared-memory multi-processor (SMP) domain for connecting nodes that are multiple hops away to reduce latency. Flattened butterfly is an on-chip network topology that uses routers with high port-counts to map a richly connected butterfly network onto a two-dimensional substrate using a two-level hierarchy. It reduces the network diameter to two, thus minimizing the impact of multiple router-hops. However, the total number of links required in each dimension grows quadratically with the number of nodes in the dimension. Multi-drop express channels (MECS) uses an one-to-many communication model enabling a high degree of connectivity in a bandwidth-efficient manner. It solves flattened butterfly's problem of requiring too many links by using an express multi-drop bus originating from each router to connect other routers along the dimension. However, the crossbar has only one port per direction like the baseline, and so all the express links entering a router from a particular direction need to multiplex on the same crossbar port, thereby sacrificing throughput.

2.2.3 Virtual Express Topologies

Virtual Express Topologies are a class of flow-control techniques and router microarchitecture designs that reduce router energy/delay overheads by creating dynamic fast-paths in a network. The basic idea is that flits try to bypass buffering and arbitrations at the router and proceed straight to the switch (ST) and link traversals (LT), thereby reducing the latency and power (as buffer reads/writes and VC/Switch arbitrations are skipped). The flits that fail to bypass get buffered and proceed through the conventional baseline pipeline described in Section 2.2.1. No extra physical links are required, and the crossbar is the same as the baseline crossbar. However, this means that the incoming flits from various ports (which wish to bypass the router) and the locally buffered flits all need to multiplex on the same output links. Virtual express topology proposals essentially describe efficient ways of performing this multiplexing such that most flits are able to bypass the routers along their path.

Express Virtual Channels (EVCs) [12], Token Flow Control (TFC) [13] and Prediction Router [14] are examples of virtual express topology proposals. Express Virtual Channels statically partitions all the VCs into 1-hop, 2-hop, ..., l_{max}-hop

express VCs (EVCs). Flits arbitrate for appropriate EVCs during VA depending on their routes. A flit that gets a k-hop EVC can bypass the k-1 intermediate routers along that dimension, by sending a lookahead one cycle in advance to pre-allocate the crossbar at the next router. EVCs uses deterministic XY routing and does not allow bypassing at turns to avoid contention among different lookaheads. Token Flow Control also enables router bypassing by obtaining and chaining together *tokens* (hints about buffers and VCs at neighboring routers) to form arbitrarily long bypass paths. It supports contention among lookaheads, thereby allowing adaptive routing and bypassing along turns. Prediction Router enables bypassing by predicting the output port and setting up the switch a-priori.

2.2.4 Chosen Topologies

We choose a 16x16 mesh as our baseline topology. Due to limited space, a smaller 8x8 mesh example is shown in Figure 2-2(a), with each router having 5 ports and implementing the 5-stage pipeline described in Section 2.2.1. The physical express topology we use for our evaluation studies is based on express cubes and uses 9-port routers with express links connecting each router to the router that is 4-hops away, in each direction, as shown in Figure 2-2(b). We refer to this topology as *Express Physical Channels (EPC).*

The virtual express topology we use for our experiments is EVC since it maps well to the EPC design by not allowing turning paths and restricting the bypassing to a maximum of l_{max} hops (which we choose to be four in our design)[1] at a time. The EVC topology uses a 16x16 mesh like the baseline.

EPC and EVC might not be the best possible physical and virtual express topologies for all scenarios in their individual domains, but they capture the essential properties of these design spaces that we wish to compare, namely the performance, power and area implications of long physical links and high crossbar ports versus opportunistic bypassing and multiplexing on the crossbar ports and links.

Figure 2-1 shows the microarchitecture of the EPC and EVC routers.

[1]Higher values of l_{max} complicate flow-control signaling and result in diminishing returns [12]

Figure 2-3: Equalized bisection bandwidth.

2.3 Evaluation

In this section, we present a detailed comparison of a physical express topology, Express Physical Channels (EPC), and a virtual express topology, Express Virtual Channels (EVC), using both conventional full-swing copper interconnects and CDLSI low-swing links. We study the configurations along three dimensions under which future NoCs might be constrained, namely, bisection bandwidth, chip area and power. The simulation infrastructure and the various experiments are described next.

2.3.1 Simulation Infrastructure

To compare the different configurations, we used a cycle-level on-chip network simulator, Garnet [37]. Garnet models a detailed 5-stage router pipeline as described in Section 2.2. We used Orion 2.0 [38], an architecture-level network energy and area model, to evaluate the power consumption and area footprint of routers. For the links, we use the numbers discussed in Section 2.1. All our experiments were done at 32nm technology node. In our results, we report only dynamic power and not leakage power. This is because at 32 nm technology node, leakage power is expected to be pretty high and aggressive leakage reduction techniques would be employed to reduce overall power consumption. Such techniques are not modeled in Orion 2.0, and thus we choose not to report leakage power. Evaluation with synthetic traffic uses packets which uniformly consist of either 8-byte control packets or 72-byte data packets. We assume three message classes out of which two always carry control packets and one is dedicated to data packets. This was done to closely match the modern cache coherence protocol requirements, as was done in [12]. Table 2.2 lists various network parameters that are common to all the configurations we evaluated for synthetic traffic. We ran experiments for uniform random, tornado and bit complement traffic. We do not report results for tornado traffic since the trends were very similar to uniform random traffic. Next, we describe various topology configurations that we evaluated.

Technology	32 nm
V_{dd}	1.0 V
$V_{threshold}$	0.7 V
Frequency	5 GHz
Topology	16-ary 2-mesh (The physical express topologies have additional express links)
Routing	Dimension ordered X-Y routing
Synthetic Traffic	Uniform Random, Tornado, Bit Complement

Table 2.2: Simulation parameters

2.3.2 Configurations

We evaluate two sets of configurations across all topologies. First we assume normal Cu wires for the links. We then perform experiments using CDLSI links for both local and express links in all topologies.

Configurations with normal Cu wires

For the EPC topology, we assume 1mm links between routers and 4mm express links. It takes three cycles to traverse these express 4mm paths as discussed in Section 2.1. For the EVC network, it is assumed that the EVC flits spend one cycle traversing the router (ST) and one cycle traversing the links (LT). The reverse signaling for buffer and VC flow-control is assumed to take one cycle for both EPC and EVC. The baseline design is the 5-stage router design described in Section 2.2.

Future NoC designs could be constrained along different dimensions depending on system requirements. Three constraints which we feel NoC designs might face are bisection bandwidth, chip area and power consumption. With this in mind, we perform experiments to compare the various topologies under the above constraints. After fixing a constraint, we perform design-space explorations to find out the best performing configuration for a particular topology. For the baseline design, we assume 128-bit links (flit-width). Varying the flit-width of the baseline will change the parameters of other topologies when we normalize designs. To study its effect, we performed all our experiments with varying baseline flit-width and found that the conclusions of our experiments do not change. Hence we report results with 128-bit flit-width as baseline. We will refer to this configuration as *Baseline*. With 128-bit

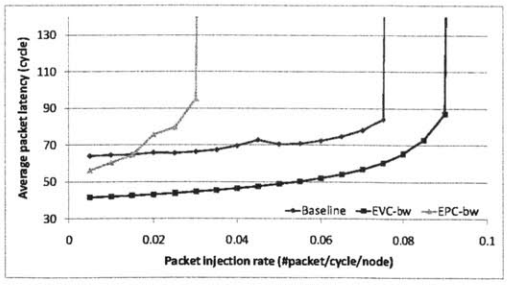
flit-width, a 72-byte data packet has five flits, while an 8-byte control packet forms one flit.

As mentioned before, we perform three sets of experiments: one in which bisection bandwidth across designs is equalized, another in which the router area is equalized, and finally we compare designs given the same power budget.

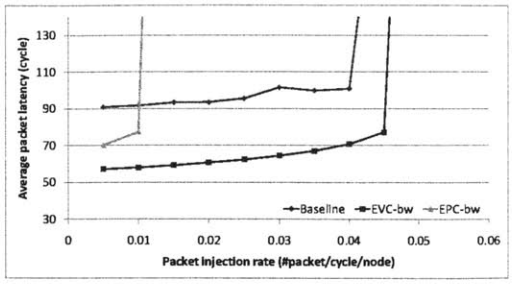
Configuration with equalized bisection bandwidth: In the first set of experiments, we keep the bisection bandwidth across all topologies constant. The EVC topology has the same number of ports at the *Baseline* topology and hence the flit-width remains the same as 128 bits. As with every configuration we choose the number of buffers and VCs that leads to the best performance for the EVC topology. We call this configuration *EVC-bw*. The EPC topology has five times as many channels at the bisection cut of the *Baseline* design. To equalize the bisection bandwidth, the flit-width of the EPC design thus becomes one-fifth of that of the baseline, which is 24 bits. This is shown in Figure 2-3. A 72-byte data packet thus consists of 24 flits, whereas an 8-byte control packet is composed of three flits. Now we vary the number of VCs and buffers and find out the best performing configuration. This will be called *EPC-bw* in the rest of the chapter.

Configuration with equalized router area: In the second experiment, we equalize the router area across all configurations. The crossbar and buffers are the major area consuming components in a router and hence we equalize them. The parameters of the router that impact crossbar and buffer area are flit-width, number of VCs and number of buffers. We sweep the design space for the EPC and EVC topologies, varying the above three parameters so that the total area is the same as that of *Baseline*. We report results for the best performing configurations and call them *EVC-area* and *EPC-area*.

Configuration with same power budget: We plot the power-performance curve for all the designs and configurations that we studied and compare the performance of the designs under the same power budget. We did not have to run additional configurations for this study.

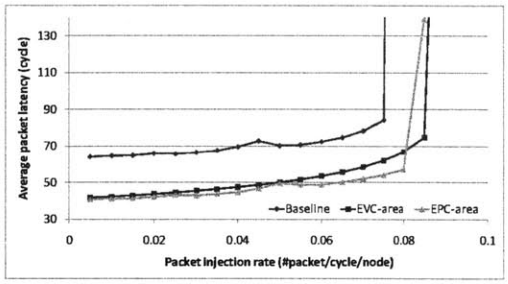


(a) Uniform Random Traffic

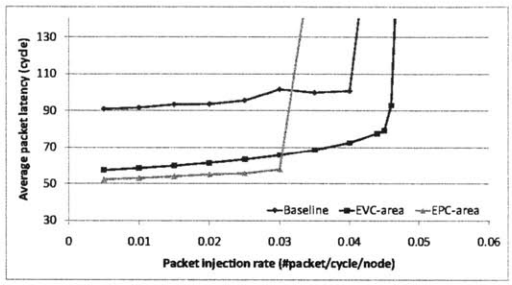


(b) Bit Complement Traffic

Figure 2-4: Network performance of various topologies with normal Cu wires under same bisection bandwidth



(a) Uniform Random Traffic



(b) Bit Complement Traffic

Figure 2-5: Network performance of various topologies with normal Cu wires under same router area

Configurations with CDLSI wires

We also evaluate all the topologies using CDLSI wires instead of conventional wires and all other parameters remaining the same as before. We again assume 1mm links between routers and 4mm express links. It takes 2 cycles to traverse these express paths using CDLSI wires. The configurations are called *Baseline-CDLSI, EVC-bw-CDLSI, EVC-area-CDLSI, EPC-bw-CDLSI,* and *EPC-area-CDLSI.* Table 2.3 shows all the configurations.

2.3.3 Results

We evaluate the above mentioned configurations with a 16x16 mesh topology. We first present the performance results for the various studies.

	Flit width (bit)	#VC/msg class	#Buffer/port
Baseline	128	6	36
Baseline-CDLSI	128	6	36
EVC-bw	128	12	84
EVC-area	128	6	36
EPC-bw	24	4	108
EPC-area	64	6	66
EVC-bw-CDLSI	128	12	84
EVC-area-CDLSI	128	6	36
EPC-bw-CDLSI	24	4	108
EPC-area-CDLSI	64	6	66

Table 2.3: Topology parameters

Normal Cu wires

The results with normal Cu wires are presented next.

Same bisection bandwidth: Figure 2-4 shows the relative performance (packet latency vs. injection throughput) of various topologies for different synthetic traffic, given the same bisection bandwidth. Note, we do not show tornado traffic as it had results similar to uniform random traffic.

For both uniform random and bit complement traffic, we observe that *Baseline* has the highest latency at low loads. The *EPC-bw* has the second highest latency and *EVC-bw* has the lowest low-load latency. The high hop-count of the baseline network leads to extra router and link hops resulting in higher low-load latencies. Both *EVC-bw* and *EPC-bw* bypass intermediate router pipelines and thus have lower low-load latencies. The maximum difference in low-load latencies is for the bit complement traffic. This is because in bit complement traffic, the sources and destinations are farthest away, leading to the best utilization of express paths (both virtual and physical).

Although *EPC-bw* completely bypasses the routers and *EVC-bw* bypasses only certain pipeline stages, the low-load latency for *EVC-bw* is lower. This is because of very low flit-widths in the *EPC-bw* networks. The *EVC-bw* network has a flit-width of 128 bits and equalizing the bisection bandwidth results in the *EPC-bw* network having a flit-width of 24 bits. A data packet (72 bytes) thus has 24 flits for *EPC-bw* and 6 flits for *EVC-bw*. The higher serialization latency of the physical express topology far outweighs the benefits of complete physical bypasses, leading to overall

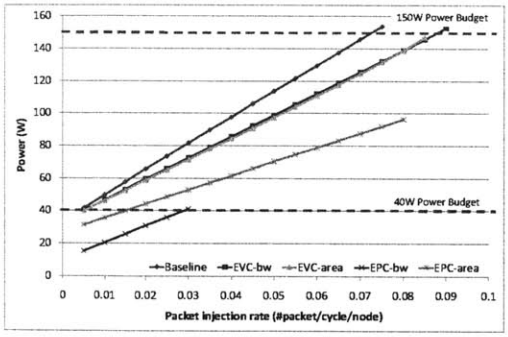


Figure 2-6: Power-Performance Curve with Normal Cu Wires

higher latency than *EVC-bw*.

In terms of throughput, the *Baseline* and the *EVC-bw* have much better saturation throughput than *EPC-bw* for all traffic patterns. This is because physical express topologies do not optimally utilize the available link bandwidth. *EVC-bw* outperforms *Baseline* due to router bypasses and pushes the saturation throughput towards the ideal.

In summary, given the same bisection bandwidth, *EPC-bw* performs worse than *EVC-bw* due to the serialization effects and poor bandwidth utilization.

Similar router area: Figure 2-5 shows the relative performance (packet latency vs. injection throughput) of various topologies for different synthetic traffic patterns, given the same router area budget. When comparing topologies with the router area normalized, we observe that the *Baseline* has the highest latency at low loads. The *EPC-area* network has the lowest low-load latency for both traffic patterns although it is only marginally better than *EVC-area* for uniform random traffic. Unlike the equalized bisection bandwidth case, (where *EPC-bw* loses out to *EVC-bw* due to higher serialization latency), the 64-bit *EPC-area* topology has lower serialization latencies leading to better low-load latencies.

In terms of throughput, *EVC-area* and *EPC-area* have better throughput than *Baseline* for uniform random traffic due to better utilization of resources and lesser contention. *EPC-area*, however, loses out on throughput to *EVC-area* because the partitioning of links leads to poorer utilization of physical links. For bit complement traffic, the throughput of *EPC-area* is even worse than that of the *Baseline*. Our implementation of EPC networks assumes a static routing algorithm and if a packet

wishes to use an express link, it keeps on trying for it even if the local links are idle. In bit complement traffic, most of the traffic goes across the chip and thus requires express links. The static nature of our routing therefore leads to poorer utilization of local links and hence the inferior throughput. Smarter adaptive routing algorithms are required to choose between the local and express links for such express topologies and that is beyond the scope of this study.

In summary, given the same router area budget, the performance of *EPC-area* is highly traffic dependent while *EVC-area* is more robust.

Similar power budget: Many-core CMPs are already facing the power wall and if we go by current trends, the primary design constraint that future NoCs will face might be power. An NoC architect might need to choose between a set of network topologies, given a network power constraint. The performance evaluations from the equal bisection bandwidth and equal router area concluded in favor of EVC networks due to higher throughput than EPC and comparable low-load latencies. However, the complete picture can be seen in Figure 2-6 which shows the network power consumption of all topologies with varying performance for uniform random traffic. Given this plot and a network power budget, one can choose the best performing network topology. For example, if the total on-chip network power budget is 40W, the *EPC-bw* network is the best performing one, even though in terms of absolute latency and throughput, it is the worst as seen before. *EPC-bw* has the lowest flit-width among all configurations and that leads to lower buffer, crossbar and link power consumption. However, if the desired throughput is higher than the saturation throughput of *EPC-bw*, the network power budget would have to be increased and *EPC-area* becomes a winner now since it consumes lower power than the EVC networks for the same throughput. But once EPC networks saturate, EVC networks perform the best at higher network budgets like 150W. The *Baseline* network always performs worse than EVC and EPC networks under a given power budget. Note that the observed power consumption of these topologies is much higher than what actual designs would allow in future, but there is a huge body of work that tackles the power consumption of routers and how to minimize that. The discussion of such works are beyond the scope of this work.

In summary, EPC topologies are more attractive than the EVC networks from a

performance/watt perspective as long as they meet the desired throughput.

CDLSI wires

The results with CDLSI wires are presented next.

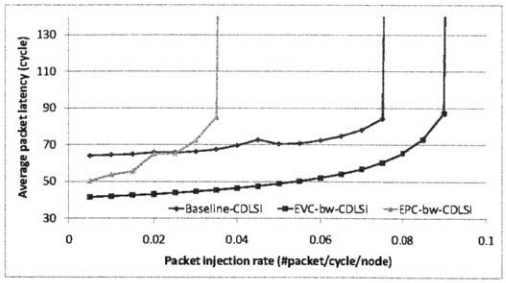
Same bisection bandwidth: Figure 2-7 shows the relative performance (packet latency vs injection throughput) of different topologies with various synthetic traffic and given the same bisection bandwidth. The addition of CDLSI links does not affect the relative trends that were observed in the comparisons without CDLSI links. We thus, do not discuss these results and refer the readers to Section 2.3.3.

Similar router area: Figure 2-8 shows the relative performance (packets injected per cycle) of various topologies for synthetic traffic and given the same router area. Adding CDLSI links to the EPC topology leads to lower long-hop latencies and the 4-hop links now take only two cycles to traverse, whereas it took three cycles with normal Cu wires. Lower long-hop latency speeds up packet delivery and eases contention and thus leads to better overall throughput. The effect of this is seen in the uniform random traffic case, where we see that *EPC-area-CDLSI* has a better throughput than *EVC-area-CDLSI*. The trend was opposite with normal Cu wires. Thus, CDLSI wires help EPC topologies outperform EVC topologies, making them an attractive design under certain scenarios. With bit complement traffic, the relative performance of various topologies is similar to what was seen with normal Cu wires in Section 2.3.3.

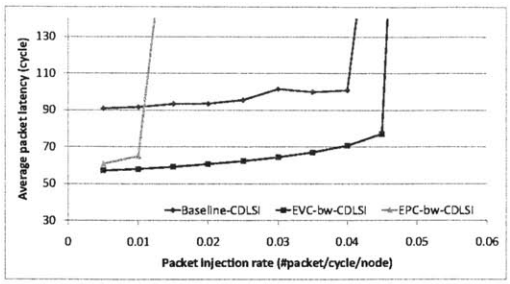
Similar power budget: CDLSI links do not affect the relative power consumption trends that we observed in Figure 2-6 with normal Cu links. We thus, do not discuss the details of the observation again.

Power breakdown

In order to understand the power consumption of various topologies, we study the power breakdown of various on-chip network components. Figure 2-9 shows the detailed network power breakdown normalized to the total power consumed by *Baseline*. As expected, the topologies with CDLSI links have lower power as compared to the same topology with Cu wires. This is due to the low-swing nature of CDLSI links. The link power of the CDLSI topologies goes down dramatically, thus lowering the overall

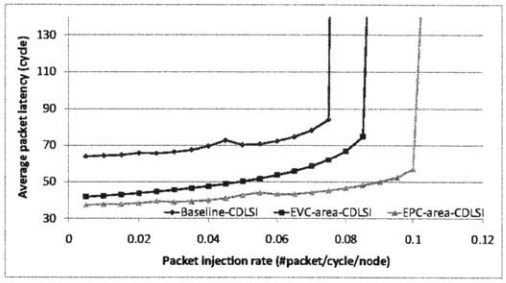


(a) Uniform Random Traffic

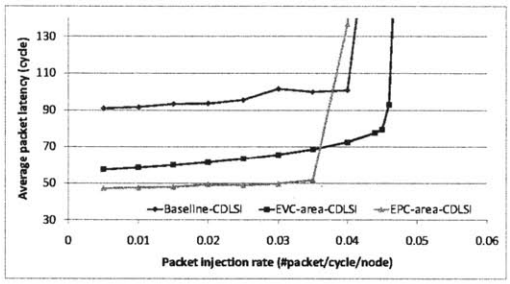


(b) Bit Complement Traffic

Figure 2-7: Network performance of various topologies with CDLSI wires under same bisection bandwidth



(a) Uniform Random Traffic



(b) Bit Complement Traffic

Figure 2-8: Network performance of various topologies with CDLSI wires under same router area

power consumption. Another observation is that the buffers, crossbar and clock are the major power consuming components in the router. This is as expected. The EPC networks have the lowest overall power, primarily due to the reduced flit-widths that these networks have. This reduces buffer and crossbar power. Interestingly, lower flit-width also eases the load on the clock distribution network thus lowering clock power in the router. The power consumption of the allocators is negligible.

2.3.4 Discussion

We summarize the findings of our experiments next.

- Virtual express topologies provide the best throughput configuration, given the same resources, and are more robust across all traffic patterns.
- The latency and throughput of physical express topologies degrades a lot with thinner links due to the serialization latency.

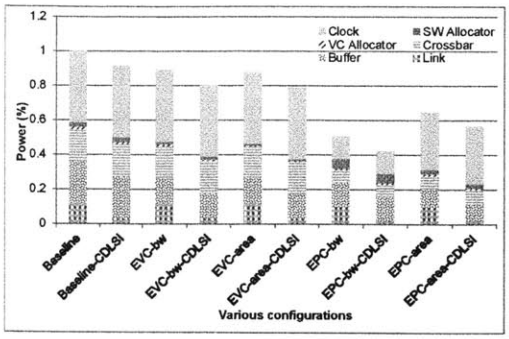


Figure 2-9: Network power breakdown for various topologies

- Given sufficient link width (bisection bandwidth), the performance of physical express topologies is highly traffic dependent. While uniform random traffic is able to extract the benefits of the express links and local links, traffic patterns like nearest neighbor would result in complete non-utilization of express links, while bit-complement needs to rely on an adaptive routing scheme to remove contention on the express links.

- When CDLSI links are used, the physical express topologies leverage the low latency physical express links to achieve the best low load latencies and also improve throughput.

- When performance/watt is considered, the physical express topologies are the better choice, until they saturate. After that express virtual topologies seem to perform best for a given power budget.

2.4 Chapter Summary

In this Chapter, we compared the effectiveness of physical and virtual express topologies for meshes with large node counts, under constraints of bisection bandwidth, router area and power consumption. We also evaluated the impact of the capacitively-driven low-swing interconnect (CDLSI) links on all designs. We observed that while both designs have similar low-load latencies, virtual express topologies give higher throughputs and are more robust across traffic patterns. Physical express topologies, however, deliver a better throughput/watt and can leverage CDLSI to lower the

latency and increase throughput.

Chapter 3

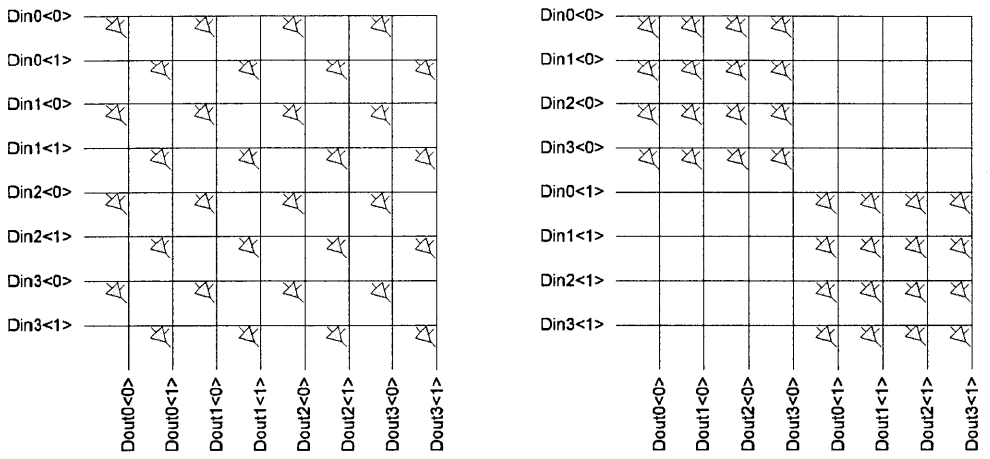
Low Power Crossbar Generator

In Chapter 2, low-swing signaling has been shown to be effective in reducing the power consumption in links. We noticed that crossbars also consume significant amount of power as compared to the total power consumed in NoC. Krishna et. al [23] showed that applying low-swing signaling techniques to crossbars is also beneficial to lowering its power consumption. However, current CAD tools do not support circuit designs with low-swing techniques; hence, designing a crossbar with low-swing technique support requires manual layout that is time-consuming and error-prone. In this Chapter, we applied the low-swing signaling techniques to the datapath that includes a crossbar and links and developed a generator that systematically generates low-swing datapaths given different architectural parameters and design preferences.

The rest of the chapter is organized as follows. Section 3.1 presents some background on crossbars and low-swing link design. Section 3.2 explains our low-swing crossbar and link generator. Section 3.3 provides some evaluation results for some datapaths generated using our tool, and Section 3.4 summarizes the chapter.

3.1 Background

In this section we present background on crossbars, low-swing links, and the limitations of the current synthesis flow.



(a) Port-sliced organization

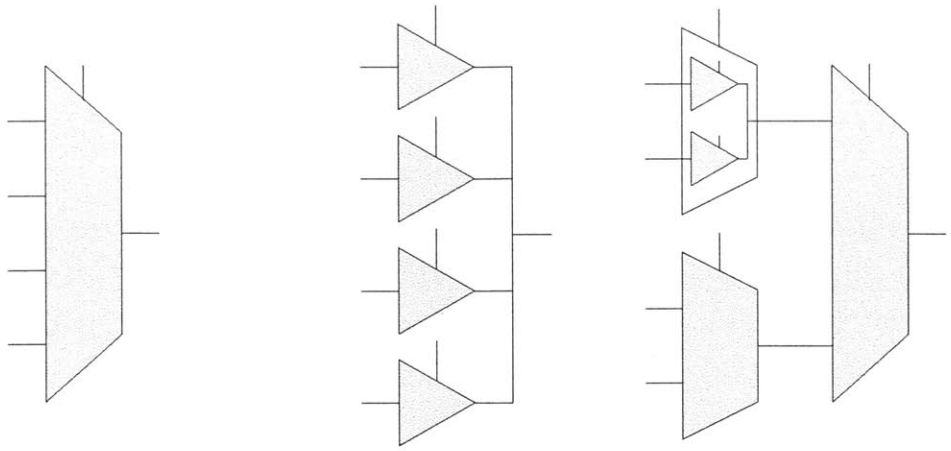
(b) Bit-sliced organization

Figure 3-1: 2-bit 4×4 crossbar schematic

3.1.1 Crossbar

A $N \times M$ crossbar connects N inputs to M outputs with no intermediate stages, where any inputs can send data to any non-busy outputs. Figure 3-1 shows the schematic of a 2-bit 4×4 crossbar. In effect, a 1-bit $N \times M$ crossbar consists of M $N : 1$ multiplexers, one for each output. The $N : 1$ multiplexer can be realized as one logic gate or cascaded smaller $N' : 1$ multiplexers, where $N' < N$, as shown in Figure 3-2. A custom-circuit designer often favors the former implementation due to the layout regularity, as it enables various optimization techniques. However, this implementation suffers from the fact that the intrinsic delay of the multiplexer grows with N. Synthesis tools usually use the latter approach that cascades smaller multiplexers to implement a $N : 1$ multiplexer with arbitrary N. By using this approach, the intrinsic delay grows with $\log N$ instead of N. However, it may lead to higher power consumption since more multiplexers are used.

Two gate organizations are possible for many-bit crossbars, as shown in Figure 3-1. One organization, port-slicing, groups all the bits of a port close to each other. The other organization, bit-slicing, groups all the gates of a bit together. The former approach eases routing (since all bits for an input/output port are grouped together), and minimizes the span of the control wires that operate the multiplexers for each input port. However, using the former approach leaves lot of blank spots that in-



(a) (b) (c)

Figure 3-2: Logical 4:1 multiplexer (a) and two realizations (b)(c)

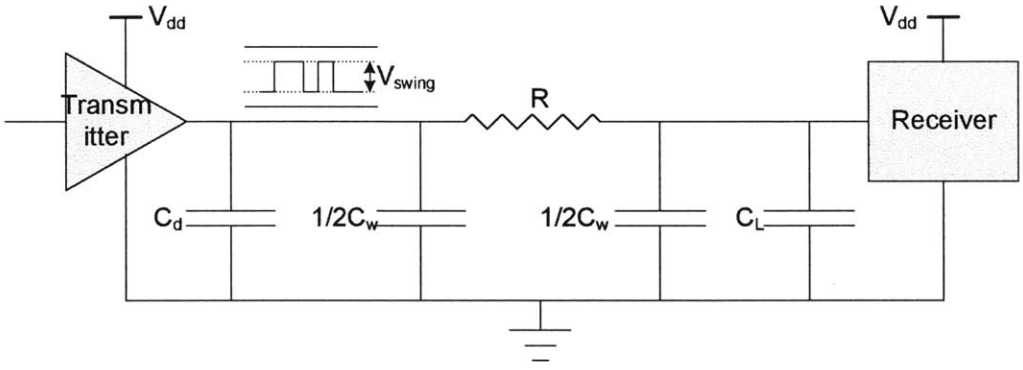


Figure 3-3: Simplified datapath

creases area, and folding the crossbar over itself to reduce area is non-trivial. The latter approach, on the other hand, minimizes the distance between the gates that contribute to the same output bit. This design is easier to optimize for area by placing all the bit-cells together and eliminating blank spaces, but requires more complicated routing to first spread out and then group all bits from a port.

In addition to a crossbar, links and receivers form a datapath. Different design decisions for these components would result in trade-offs in area, power and delay. From the perspective of sending a signal, a datapath can be simplified to three components connected together: a transmitter, a wire, and a receiver, as shown in Figure 3-3. The corresponding delay and energy consumption can be formulated as follows:

$$Energy = (C_d + C_w + C_L)V_{DD}V_{swing} \tag{3.1}$$

$$Delay = ((C_d + C_w + C_L)V_{swing}/I_{av}) \tag{3.2}$$

where C_d is the output capacitance of the transmitter, C_w is the wire capacitance, C_L is the input capacitance of the receiver. V_{DD} is the power supply of the circuit, and V_{swing} is the voltage swing on these capacitors. I_{av} is the average (dis)charge current. In general, lowering the capacitance, reducing the voltage swing, and increasing the (dis)charging current can help in reducing energy consumption and delay.

A transmitter with larger sized transistors would have larger (dis)charging current which would decrease the delay. But it has larger footprint and C_d. Greater wire spacing lowers the coupling capacitance between wires but it takes larger metal area. Increasing wire width could reduce the wire resistance but it also increases capacitance and metal area.

3.1.2 Low-swing signaling

Current on-chip network architectures require both long interconnects for the connection of processor cores, and small wire spacing for higher bandwidth. This trend has significantly increased wire capacitance and resistance. Unfortunately, physical properties of the on-chip interconnects are not scaling well with transistor sizes. To reduce the delay and power consumption caused by these RC-dominant wires, low-swing circuit techniques [39] are now in the spotlight of on-chip networks.

In low-swing interconnects, the propagation delay decreases linearly with the signal swing, under the condition that the charge/discharge current is not affected by the reduction in voltage swing. Furthermore, the lower signal swing carries the major benefit of reducing dynamic power consumption, which can be substantial when the interconnect load capacitance is large. The low-swing signaling schemes, however, give rise to noise concerns.

Some of the noise concerns in low-swing designs can be mitigated by sending data differentially, which helps eliminate common-mode interference. However, this takes up two wires which doubles the capacitance and area. Adding shielding wires also helps reduce crosstalk and could potentially lower voltage-swing, but it also

adds coupling capacitance and area. Increasing the sensitivity of the receiver helps lower voltage-swing on the wires, but it often needs a larger sized transistor or more sophisticated receiver design that has larger footprint and capacitance.

Thus low-swing links offer tremendous advantages in terms of power and latency, but require careful design to ensure robust performance.

3.1.3 Limitations to current synthesis flow

Given a hardware description of a crossbar, the existing synthesis flow, like the one shown in Figure 3-4, with a standard cell library could synthesize and realize a crossbar circuit. Unfortunately, the existing synthesis flow and standard cell libraries are designed for full voltage-swing digital circuits. New features in certain CAD tools enable low power designs by supporting multiple power domains and power shutdown techniques. However, none of them support analysis and layout for low voltage swing operations. Moreover, place-and-route tools are often too general and cannot take full advantage of the regularity of a crossbar and fail to generate an area-efficient layout. Therefore, a system designer needs to custom-design a low-swing crossbar, which is time-consuming and error-prone.

3.2 Datapath Generator

In this section we present our crossbar and link generator for low-swing datapaths. The low-swing property is enabled by replacing the cross-points of a crossbar with low-swing transmitters and adding receivers at the end of the links to convert low-swing signals back to full-swing signals. The data links that connect transmitters and receivers are equipped with shielding wires to improve signal integrity. As shown in Table 3.1, our proposed datapath generator takes architectural parameters (e.g. the number of inputs and outputs, data width per port, link length), user layout preferences (e.g. port locations, link width and spacing), and technology files (e.g. standard cell library, targeted metal layers, TX and RX cells), and generates a crossbar and link layout that meets specified user preferences and system design constraints: area, power, and delay. The output files of our proposed datapath generator are fed directly into a conventional synthesis tool flow, which is similar to how we use a memory

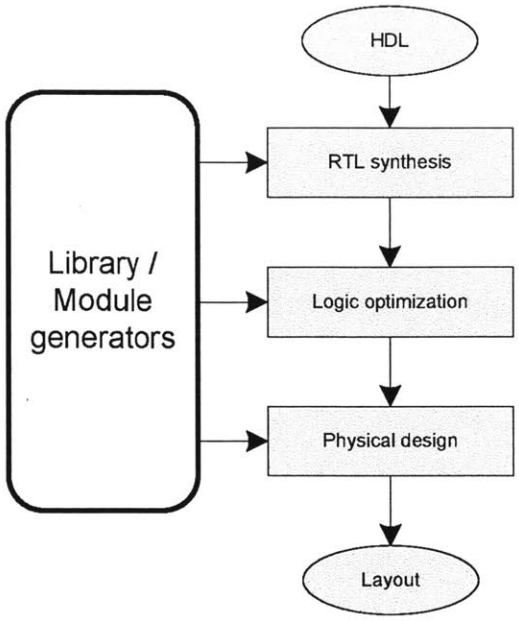


Figure 3-4: Standard synthesis flow

compiler. Figure 3-5 shows the proposed datapath generation flow. The generation involves two phases, library generation and selection. In the library generation phase, the program takes a suite of custom-designed transmitters and receivers, architectural parameters that users are interested in, and technology files as inputs; Then, it pre-characterizes the custom circuits. Next, the tool generates the layout of all possible combinations and simulates them to get post-layout timing, power, and area. This forms the library of components for the selection phase. In the selection phase, the generator takes architectural parameters and user preferences as inputs to find the most suitable design from the results generated in the library generation phase, and outputs the files needed for the synthesis flow.

In the following subsections, we walk through a detailed example of generating a datapath with a 64-bit 6×6 crossbar, 1mm links, and receivers in a 45nm SOI HVT technology.

3.2.1 Building block pre-characterization

We treat the 1-bit transmitters and receivers as atomic building blocks of the generator, thus giving users the flexibility of using different kinds of transmitter and receiver

Table 3.1: Inputs to proposed datapath generator

Type	Proposed datapath generator
Architectural parameters	Number of input ports (N) Number of output ports (M) Data width in bits (W) Link length (L)
User preferences	Input port location Output port location Link wire width and spacing
Technology related information	Standard cell library Metal layer information Transmitter and receiver design Second power supply level (if needed)
System design constraints	Target frequency, power, area

Table 3.2: Pre-characterization results

	Transmitter	Receiver
Average current (μA)	2.6	11.0
Input cap (fF)	1.52 (select) 2.87 (data)	1.05 (clk) 0.4 (data)

designs. Given the transmitter and receiver designs, the generator first performs pre-characterization using Spice-level simulators (we used Cadence UltraSim) to obtain average current and input capacitances. The average current is later used to determine the power wire width, while the input capacitances are used to determine the size of the buffers that drive these building blocks.

For example, Figure 3-6 depicts the schematic of a low-swing transmitter design and a receiver design we chose as inputs to the generator. The experiments in both this section and Section 3.3 are performed using the IBM 45 SOI HVT technology, and the pre-characterization results are shown in Table 3.2.

3.2.2 Layout generation

In this step, the generator tiles the transmitters and receivers to form the datapath, taking various aspects into consideration, such as building block restrictions, floorplanning, routing, and link design. This section details each of these aspects.

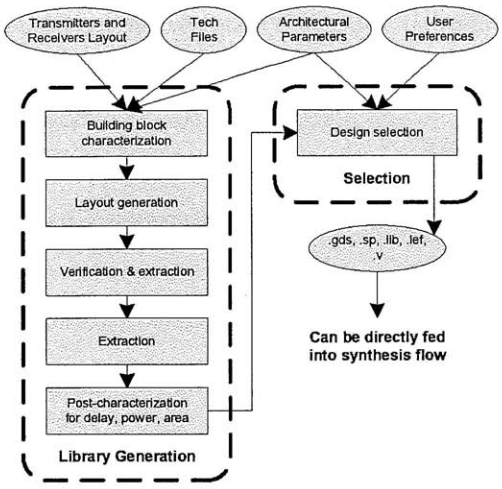
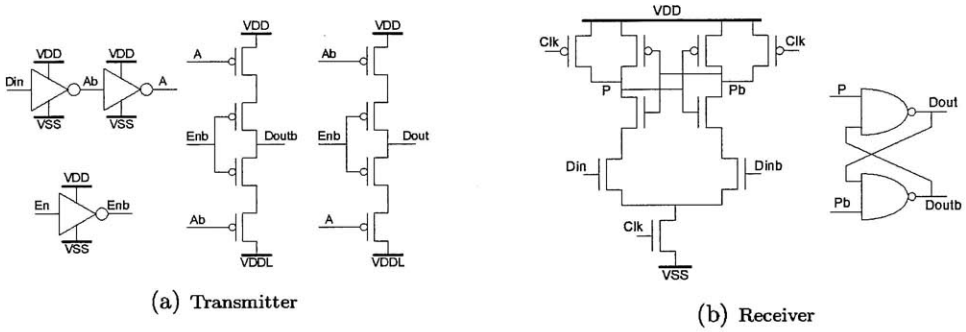


Figure 3-5: Proposed Datapath Generator's Tool flow



(a) Transmitter (b) Receiver

Figure 3-6: Schematic of transmitter and receiver

Building block restrictions

We applied constraints to the transmitters' and receivers' pin locations. The reason is twofold. First, the gates of the transistors for low-swing operations are more sensitive to coupling from full-swing wires. Therefore, some constraints on transmitters' and receivers' pin location are helpful to avoid routing low-layer full-swing signal wires over these transistors. Second, constraints on pin locations make the transmitter/receiver cells more easily tile-able. Without loss of generality, we chose one specific pin layout, restricted as shown in Figure 3-7. The power and ground pins' locations are chosen to be the same as the corresponding pins in standard cells. All other pins are placed relative to the transmitter's core, which contains noise-sensitive transistors. For example, the Select pin is on the left of the core, the Data-in pin

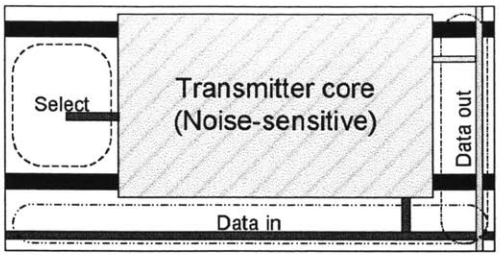


Figure 3-7: Transmitter abstract layout

is at the bottom, and the Data-out pin is on the right. Similar constraints are also applied to the receiver cell design.

Floorplanning

To achieve higher transmitter cell area density, we chose the bit-sliced organization, which was shown earlier in Figure 3-1(b). The tool first generates a 1-bit $N \times M$ crossbar as shown in Figure 3-8. The transmitters are placed at the cross-points of input horizontal wires and output vertical wires. The tool then places W 1-bit crossbars in a 2-dimensional array to form a W-bit $N \times M$ crossbar, as shown in Figure 3-9. The number of 1-bit crossbars on each side is calculated to square the crossbar layout area so as to minimize the average length of the wires each bit needs to traverse. Receivers are placed so that the routing area from the links to the receiver inputs is minimal.

Although a port-sliced organization is also effective, it requires a more sophisticated wire routing algorithm to achieve the same cell area density as a bit-sliced organization. A naive approach, as shown in Figure 3-1(a), would result in low-transistor density and a W^2 bit-to-area relationship, instead of W which can be readily achieved by using the bit-sliced organization.

Routing

For each 1-bit crossbar, the number of metal layers needed to route the power and signals is kept minimal to maximize the number of available metal layers for output wire routing. No wiring is allowed above noise-sensitive transistors in lower metal layers. While this increases the total crossbar area, it lowers the wiring complexity for

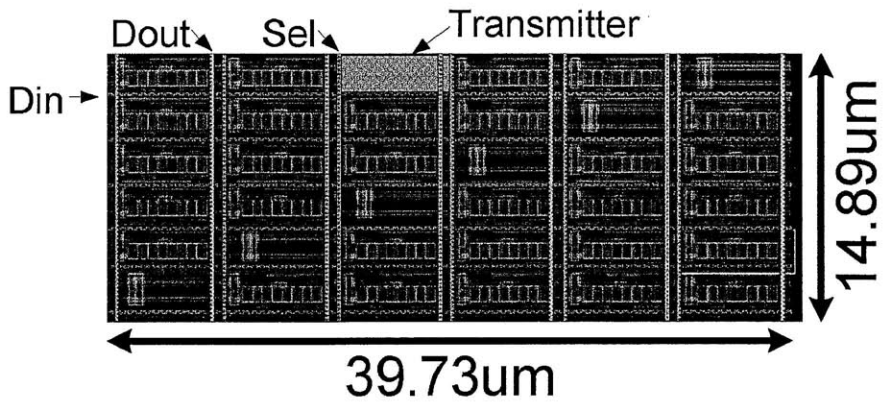


Figure 3-8: Example single-bit crossbar layout with 6 inputs and 6 outputs

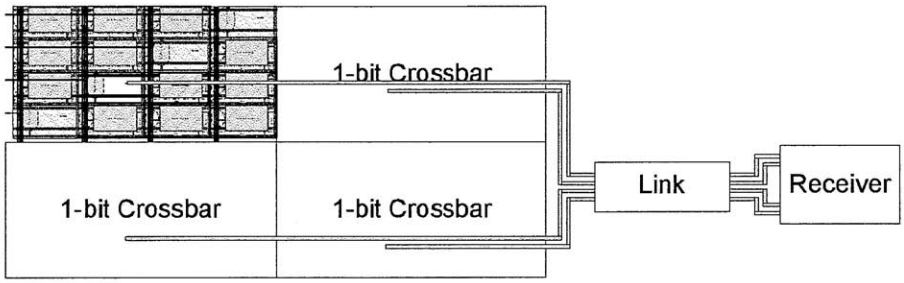


Figure 3-9: 4-bit crossbar abstract layout with 1 port connecting to the link

Data-out wires from each 1-bit crossbar to crossbar outputs. Since we employed the bit-sliced organization, the Data-out wires are distributed across the entire crossbar. Two metal layers are used to route the Data-out wires to the edge of the crossbar: one is used for outputs in horizontal direction, while the other is used for the vertical direction. Since the same metal layer is used to route all wires in a particular direction, the crossbar area is limited by the wire pitch if the transmitter's cell area is small. Otherwise, it is limited by transmitter cell area. As shown in Figure 3-9, Data-out wires coming out from the edge of the 1-bit crossbar array are routed to the inputs of links. We carefully designed the routing algorithm so that it takes minimal wiring area to connect the outputs of a crossbar to the links.

A structured layout of the power distribution network is applied. A power ring that surrounds the whole crossbar, one that surrounds the whole receiver block, and power stripes, are all automatically generated. The widths of the power wires are

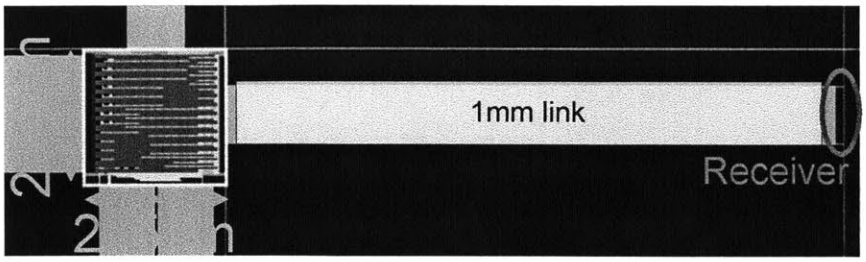


Figure 3-10: Example 6x6 64-bit datapath layout with one link shown

calculated based on the average current so that the current density is less than 1 mA/um the results from the pre-characterization, we used both 0.8μm-wide and 0.7μm-wide power wire for crossbar and receiver respectively.

Link Design

Link parameters such as link wire length, width, and spacing are specified as the inputs of the generator. Since the links are running at low-swing, they are more vulnerable to noise. We thus add shielding wires to improve the noise immunity. We chose the shielding wire organization that is shown in Figure 3-11, where a shielding wire is placed on the same layer as link between two different bits and two shielding wires are placed right below the differential wires. This is chosen as it minimizes low-swing noise from other links and full-swing logic from lower metal layers.

Typically the wire length is set based on the distance between the crossbar and the components this crossbar is connected to. Different choices of wire width and spacing would affect the timing and energy consumption of transmitting a signal. For example, one could reduce the delay by doubling the wire pitch but it requires larger wiring area. Table 3.3 shows this trade-offs between link area and link performance, where the wire width is normalized to the minimum wire width and the wire spacing is normalized to the minimum spacing. The performance was simulated by transmitting a full-swing signal on the link.

A layout of the example datapath generated is shown in Figure 3-10.

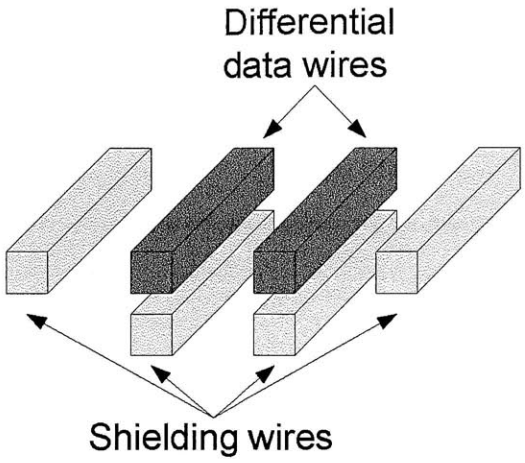


Figure 3-11: Selected wire shielding topology

Table 3.3: Performance of 1mm^2 link of two organizations

Wire width	Wire spacing	Delay (ps)	Energy/bit (fJ)	Link area (mm^2)
1	2	70.0	35.0	0.093
2	4	33.7	30.5	0.176

3.2.3 Verification and Extraction

We use Calibre from Mentor Graphics to check if the generated circuit obeys the design rules, and to perform layout versus schematic (LVS) verification. A schematic netlist is generated for LVS. In order to get a more accurate delay of the circuit, RC extraction is done for the post-characterization of the generated design.

3.2.4 Post-characterization and selection

Post-characterization is performed to determine the actual frequency, power, and area the crossbar can achieve. The selection step chooses the suitable datapath design based on the results from the post-characterization step, and outputs the files needed for the standard synthesis flow.

The Table 3.4 shows the simulation results for the walk-through examples. At the selection step, for example, if the criteria is to achieve high frequency and have little constraint on the area, the design with doubled link pitch is returned.

Table 3.4: Example generated datapaths

Link wire width	Link wire spacing	Max freq (GHz)	Crossbar area (mm^2)	Energy/bit (fJ)
1	2	2.5	0.053	46.4
2	4	2.7	0.084	48.3

3.2.5 Discussion

The proposed generator enables the layout generation of low-swing datapaths for a given set of architectural parameters and design constraints and outputs files for integration with the standard synthesis flow. The generator removes additional design effort necessary for a fully custom design by automatically and systematically tiling transceivers and routing wires. When compared to synthesizing a full-swing datapath using commercial tools, our generator adds SPICE simulation overheads to characterize the properties of the generated datapaths such as delay and power during the library generation step.

To use the generator with a different technology, the user needs to provide the technology related information as shown in Table 3.1 and to provide the layout of the transmitters and receivers designed for that specific process.

3.3 Evaluation

In this section, we first evaluate the crossbars generated by our proposed tool, against the synthesized crossbars. We then present a case study of a 5-port NoC virtual channel router that is integrated through the standard synthesis flow with the low-swing datapath generated by our tool.

In all our experiments, we used Cadence Ultrasim to evaluate the performance and power consumption of the RC extracted netlists.

3.3.1 Generated vs. synthesized datapath

Using the transmitter and the receiver design we describe in Section 3.2, we generated low-swing datapaths across a range of architectural parameters and compared the simulation results with datapaths generated by standard CAD tools using only standard cells. We will refer to the crossbar/datapaths generated by our tool as *generated* crossbars/datapaths, and those generated by standard CAD tools using

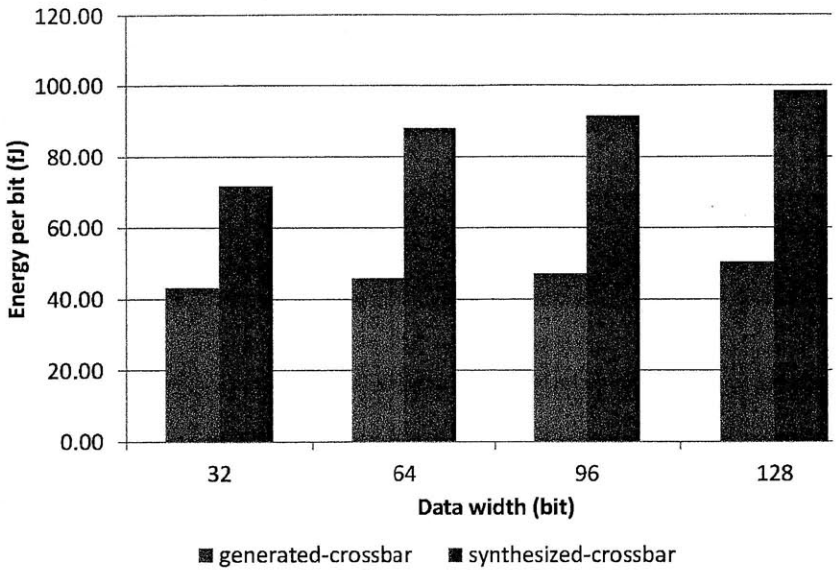


Figure 3-12: Energy per bit sent of 6-port datapaths with different data width

standard cells as *synthesized* crossbars/datapaths. Evaluating generated datapaths with different transmitter and receiver designs can be done but is equivalent to evaluating the effectiveness of different low-swing techniques, which is beyond the scope of this work. In our experiments, we assumed a link length of 1mm and specified a delay constraint of 0.6ns from the input of the crossbar to the output of the link for synthesized datapaths so that the datapath can be run at 1.5GHz.

Energy per bit. We simulated the datapaths (crossbar and link) at 1.5GHz and report the results for varying data widths and varying number of ports in Figure 3-12 and Figure 3-13, respectively. As shown in Figure 3-12, for both crossbars, as the data width increases, the energy per bit sent also increases because an increase in the data width leads to an increase in the area of the crossbar. This increase results in longer distance (on average) for a bit to travel from an input port to an output port. Longer distance translates to higher energy consumption. The energy per bit sent also increases with the number of ports, because a bit needs to drive more transmitters. Overall, our simulations showed that a generated datapath, as in our design, results in 50% energy savings (on average per bit sent) compared to a synthesized datapath.

Area. Figure 3-14 shows the area of the generated vs. synthesized crossbars. Due to the bit-sliced organization and larger transmitter size, the generated crossbar area

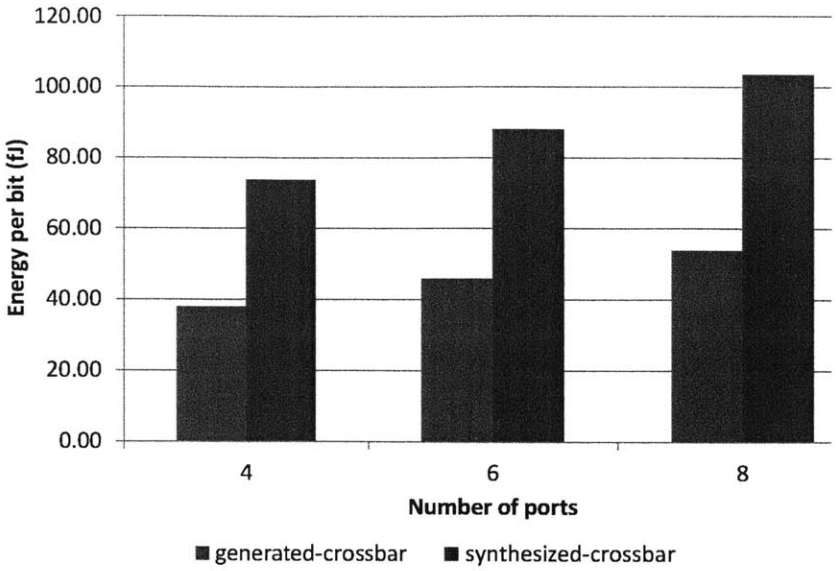


Figure 3-13: Energy per bit sent of 64-bit datapaths with different number of ports

is dominated by the transmitter area. Using this organization results in its crossbar area growing linearly with the data width and quadratically with the number of ports, as captured in Figure 3-14. On the other hand, as Figure 3-14 indicates, a synthesized crossbar has a smaller area footprint because the transmitter design we are simulating is differential, and our wire routing is conservative to achieve high immunity to noise. Both of these factors result in increased area footprint, a trade-off for better latency and lower power of low-swing datapath. In addition, the design of low-swing transceivers for on-chip is in its infancy. Future designs may be more area-efficient, and continue to be pluggable into our generator toolchain.

3.3.2 Case Study

We synthesized a typical NoC router of a mesh topology integrated with a low-swing datapath using the files generated by our tool. The router is a 3-stage pipelined input buffered virtual channel (VC) router with five inputs and five outputs [36], and with a 64-bit data path. As shown in Figure 3-15, one input and one output port are connected to the local processing unit, while the remaining ports are connected to neighboring routers. We assumed that the local processing unit resides next to

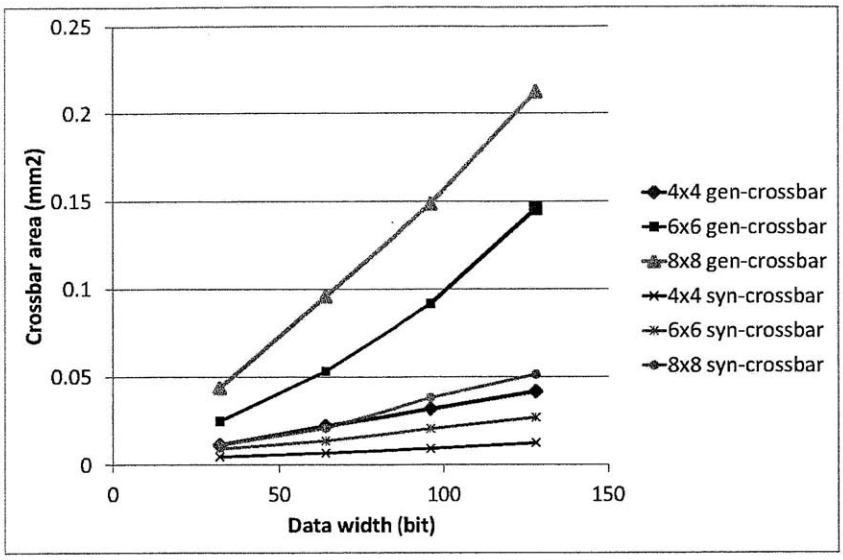


Figure 3-14: Crossbar area with various architectural parameters

the router, the distance between routers is 1mm, and the target working frequency is 1GHz. Table 3.5 shows the detailed router specifications.

We used the same synthesis flow shown in Figure 3-4 to realize the router design from RTL to layout. Figure 3-16 shows the final layout of the router with the generated datapath. The black region in the figure is assumed to be occupied by processing units. The low-swing crossbar occupies about 30% of the total router area. The delay of the low-swing datapath is 630ps. The power consumed in the generated datapath is 18% of the total power consumed by the router[1]. The power consumption was obtained from UltraSim simulations by feeding a traffic trace through all the ports of the router. The traffic trace was generated from RTL simulations of a 4x4 NoC; every node injects one message every cycle destined to a random node. The final synthesized router with the generated low-swing crossbar and links consists of 286,079 transistors.

[1]It should be pointed out that this is a textbook NoC router. With a bypassing NoC router, such as that in [23], the NoC power will be largely that of the datapath, since most packets need not be buffered and can go straight from the input port through the crossbar to the output port and link.

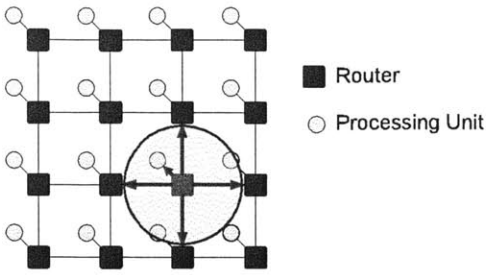


Figure 3-15: Five-port router in a mesh network

Table 3.5: Router specifications

# of input ports	5
# of output ports	5
Data width	64
# of buffers per port	16 (1k bits)
Flow control	Wormhole with VC
Buffer management	On/Off
Working frequency	1 GHz

3.4 Chapter Summary

In this chapter, we present a low-swing NoC datapath generator that automatically creates layouts of crossbar and link circuits at low voltage swings, and enables the ready integration of such interconnects in the regular CAD flow of many-core chips. Our case study demonstrates our generated datapath embedded within the synthesis flow of a 5-port NoC mesh router, leading to 50% savings in energy-per-bit. While our case study leverages a specific low-swing transmitter and receiver circuit, our generator can work with any TX/RX building block. We hope this will pave the way for low-swing signaling techniques to be incorporated within mainstream VLSI design, realizing low-power NoCs and enabling many-core chips.

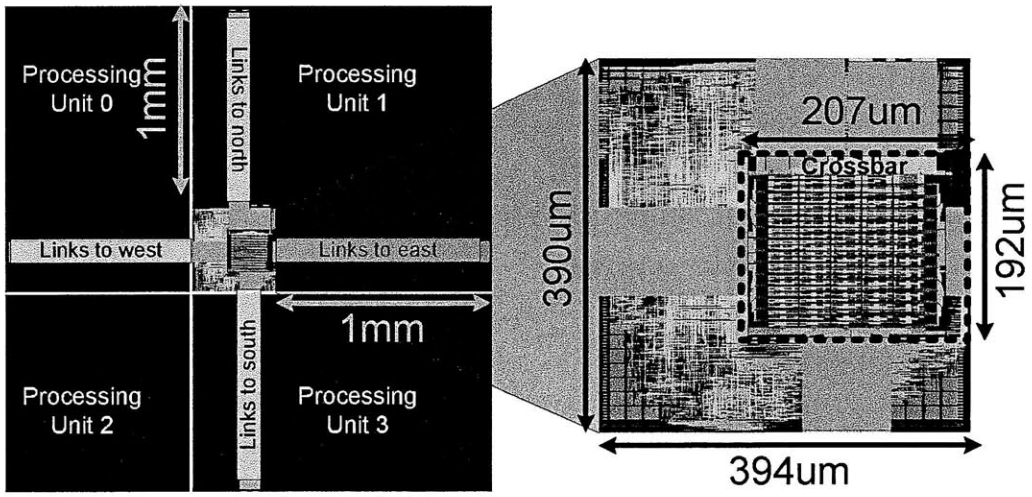


Figure 3-16: Synthesized router with generated low-swing datapath

Chapter 4

Conclusions

Designing high-performance and low-power networks-on-chips (NoCs) has become more important for future processors consisting of multiple cores on the same die or even the multiple such processors on the same substrate. This thesis has been aimed to explore some proposed NoC designs targeting both high-performance and low-power and to explore the opportunities to leverage the low-swing circuit techniques to furthur reduce the total power consumption of NoCs.

4.1 Thesis Summary

In Chapter 2, we explored the space of physical and virtual express topologies and evaluated the impact of using one low-swing circuit technique, CDLSI, on the performance and of these topologies. We evaluated both classes of the topologies under realistic system constraints, such as bisection bandwidth, router area, and power budget. We concluded that both express topologies help reduce low-load latencies. Moreover, virtual express topologies help improve throughput, whereas the physical express topologies give better performance-per-watt and can leverage CDLSI to lower the latency and to increase throughput.

We then identified that the crossbar in a router is another major source of power consumption. In Chapter 3, we applied the low-swing signaling techniques to both crossbars and links, and developed a low-swing crossbar and link layout generator. The generator takes architectural parameters as inputs and generates the layout and

other library files for integration to standard synthesis flow. The generated datapath showed 50% energy savings, and we also showed a case study of a NoC router synthesized with the generated datapath.

4.2 Future Work

As CMOS technology scales, the core counts will continue to grow to hundreds or even a thousand, posing ever stricter requirements on bandwidth and latency. As presented in this thesis, using express topologies with low-swing interconnects can help mitigate this requirement and push the network performance toward the ideal electrical fabric; however, electircal wires fundamentally do not scale with the CMOS technology and this makes the electrical fabrics difficult to meet the performance requirements needed for the systems with ever increasing core counts.

Photonics is an emerging interconnect technology with the potential to offer unparalleled channel capacity and energy efficiency compared to electrical interconnects. Silicon-based photonics enables integration with CMOS electronics, so that benefits of optical links can be enjoyed at chip-scale granularity. Several NoC designs have been proposed that leverage this emerging technolgy [40, 41, 42, 43]. However, due to the immaturity of photonic technology, many device parameters are still far from ideal and hence there is a wide range of values that we can assume for these parameters. I plan to compare photonic-inspired NoC designs with the sophisticated electrical NoC designs, and to propose and realize new opto-electronic NoC designs that have better performance and power than electrical NoCs.

Bibliography

[1] Michael Kistler, Michael Perrone, and Fabrizio Petrini. Cell multiprocessor communication network: Built for speed. *IEEE Micro*, 26:10–23, May 2006.

[2] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan. Larrabee: a many-core x86 architecture for visual computing. In *ACM SIGGRAPH*, pages 18:1–18:15, 2008.

[3] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, September 2007.

[4] J. Howard et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *IEEE Intl. Solid-State Circuits Conference (ISSCC)*, Febrary 2010.

[5] Michael Bedford Taylor, Walter Lee, Jason Miller, David Wentzlaff, Ian Bratt, Ben Greenwald, Henry Hoffmann, Paul Johnson, Jason Kim, James Psota, Arvind Saraf, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2004.

[6] Karthikeyan Sankaralingam, Ramadass Nagarajan, Haiming Liu, Changkyu Kim, Jaehyuk Huh, Doug Burger, Stephen W. Keckler, and Charles R. Moore. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2003.

[7] D. Wentzlaff et al. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5), 2007.

[8] P. Gratz, C. Kim, R. McDonald, S.W. Keckler, and D.C. Burger. Implementation and evaluation of on-chip network architectures. In *Proc. Intl. Conference on Computer Design (ICCD)*, October 2006.

[9] William Dally. Express cubes: Improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 40(9), 1991.

[10] John Kim, James Balfour, and William Dally. Flattened butterfly topology for on-chip networks. In *Proc. Intl. Symp. Microarchitecture (MICRO)*, December 2007.

[11] Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu. Express cube topologies for on-chip interconnects. In *Intl. Symp. on High Performance Computer Architecture*, February 2009.

[12] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express virtual channels: Towards the ideal interconnection fabric. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2007.

[13] Amit Kumar, Li-Shiuan Peh, and Niraj K Jha. Token flow control. In *Proc. Intl. Symp. Microarchitecture (MICRO)*, November 2008.

[14] Hiroki Matsutani, Michihiro Koibuchi, Hideharu Amano, and Tsutomu Yoshinaga. Prediction router: Yet another low latency on-chip router architecture. In *Proceedings of International Symposium of High Performance Computer Architecture*, February 2009.

[15] R. Ho, T. Ono, F. Liu, R. Hopkins, A. Chow, J. Schauer, and R. Drost. High-speed and low-energy capacitively-driven on-chip wires. *IEEE Intl. Solid-State Circuits Conference (ISSCC)*, pages 412–413, February 2007.

[16] Sriram Vangal, Nitin Borkar, and Atila Alvandpour. A six-port 57gb/s double-pumped nonblocking router core. In *Symp. VLSI Circuits*, pages 268–269, June 2005.

[17] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proc. Intl. Symp. Microarchitecture (MICRO)*, 2003.

[18] Kangmin Lee, Se-Joong Lee, Sung-Eun Kim, Hye-Mi Choi, Donghyun Kim, Sunyoung Kim, Min-Wuk Lee, and Hoi-Jun Yoo. A 51mw 1.6ghz on-chip network for low-power heterogeneous SoC platform. In *IEEE Intl. Solid-State Circuits Conference (ISSCC)*, February 2004.

[19] B. Kim and V. Stojanovic. A 4gb/s/ch 356fj/b 10mm equalized on-chip interconnect with nonlinear charge-injecting transmit filter and transimpedance receiver in 90nm cmos. *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pages 66–68, February 2009.

[20] D. Schinkel, E. Mensink, E.A.M. Klumperink, A.J.M. van Tuijl, and B. auta. Low-power, high-speed transceivers for network-on-chip communication. *IEEE Trans. on Very Large Scale Integration Systems*, 17(1):12–21, January 2009.

[21] Manoj Sinha and Wayne Burleson. Current-sensing for crossbars. In *IEEE Intl. ASIC/SOC Conference*, September 2001.

[22] Panduka Wijetunga. High-performance crossbar design for system-on-chip. *Intl. Workshop on System-on-Chip for Real-Time Applications*, 0:138, 2003.

[23] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P.Chiang. SWIFT: A SWing-reduced Interconnect For a Token-based Network-on-Chip in 90nm CMOS. In *Proc. Intl. Conference on Computer Design (ICCD)*, October 2010.

[24] Ranko Sredojevic and Vladimir Stojanovic. Optimization-based framework for simultaneous circuit-and-system design-space exploration: A high-speed link example. In *Intl. Conference on Computer-Aided Design*, volume 0, pages 314–321, 2008.

[25] ARM AMBA. `http://www.arm.com/products/system-ip/amba`.

[26] Stbus communication system: Concepts and definitions. `http://www.st.com/stonline/books/pdf/docs/14178.pdf`.

[27] Drew Wingard. Micronetwork-based integration for SoCs. In *Proc. Design Automation Conference (DAC)*, pages 673–677, May 2001.

[28] IBM CoreConnect. `https://www-01.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture`.

[29] Mohamed A. Shalan, Eung S. Shin, and Vincent J. Mooney III. DX-GT: Memory management and crossbar switch generator for multiprocessor system-on-a-chip. In *Proc. Workshop on Synthesis and System Integration of MIxed Technologies*, pages 357–364, April 2003.

[30] Matteo Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi, and Luca Benini. xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proc. Intl. Conference on Computer Design (ICCD)*, October 2003.

[31] H. Zhang, V. George, and J. M. Rabaey. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Trans. on Very Large Scale Integration Systems*, 8(3):265–272, June 2000.

[32] P. Kapur, G. Chandra, and K. C. Saraswat. Power estimation in global interconnects and its reduction using a novel repeater optimization methodology. *Proc. Design Automation Conference (DAC)*, pages 461–466, June 2002.

[33] K.-H. Koo, P. Kapur, J. Park, H. Hoh, S. Wong, and K. C. Saraswat. Performance comparison between capacitively driven low swing and conventional interconnects for cu and carbon nanotube wire technologies. *Intl. Interconnect Technology Conference*, 2009.

[34] International technology roadmap for semiconductors. http://www.itrs.net.

[35] W. J. Dally and B. Towles. Route packets not wires: On-chip interconnection networks. In *Proc. Design Automation Conference (DAC)*, June 2001.

[36] William J. Dally and Brian Towles. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann Publishers, 2004.

[37] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj Kumar Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *IEEE Intl. Symp. on Performance Analysis of Systems and Software*, April 2009.

[38] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proc. Design, Automation and Test in Europe (DATE)*, February 2009.

[39] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits: A Design Perspective, second edition.* Prentice Hall, 2003.

[40] Dana Vantrease, Robert Schreiber, Matteo Monchiero, Moray Mclaren, Marco Fiorentino, Al Davis, Nathan Binkert, Raymond G. Beausoleil, and Jung Ho Ahn. Corona: System implications of emerging nanophotonic technology. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, 2008.

[41] Yan Pan, Prabhat Kumar, John Kim, Gokhan Memik, Yu Zhang, and Alok Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, 2009.

[42] Scott Beamer adn Chen Sun, Yong-Jin Kwon, Ajay Joshi, Christopher Batten, Vladimir Stojanovic, and Krste Asanovic. Re-architecting dram memory systems with monolithically integrated silicon photonics. In *Proc. Intl. Symp. Computer Architecture (ISCA)*, June 2010.

[43] George Kurian, Jason E. Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C. Kimerling, and Anant Agarwal. Atac: a 1000-core cache-coherent processor with on-chip optical network. In *Intl. Conference on Parallel Architectures and Compilation Techniques*, 2010.