High-Radix Scalable Modular Crossbar Switches

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Cagla Cakir

B.S., Electronics Engineering, Sabanci University
M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA, USA

May 2016

Abstract

As process technologies have scaled, the increasing number of processor cores and memories on a single die has also driven the need for more complex on-chip interconnection networks. Crossbar switches are primary building blocks in such networks-on-chip, as they can be used as fast single-stage networks or as the core of the router switch in multi-stage networks. While crossbars offer non-blocking, single-hop, all-to-all communication, they tend to scale poorly with the number of nodes due to the latency and energy of the long wires and high-radix multiplexor structures needed. In this work, we investigate how to improve crossbar performance, energy-efficiency, and scalability.

To better understand the design space and scaling limitations, we have developed an on chip switch modeling tool calibrated using circuit-level simulations. The tool enables a design space exploration showing how area, power, and performance vary across radix, data width, wire parameters, and circuit implementation. In addition to conventional design options, we examined capacitively coupled low-swing signaling to improve to energy consumption of the I/O wires. This exploration shows that the main bottlenecks are the long I/O wires and the key to improving the performance and efficiency is to minimize the area. Using these insights, we present modular crossbar switches that can perform better at high radices than the monolithic designs. The modular sub-blocks are arranged in a controlled flow-through, pipelined scheme to eliminate global connections and maintain linear performance scaling and high throughput. Modularity also enables energy savings via deactivation of unused I/O wires.

To evaluate our design, we implemented a prototype radix-64 modular crossbar switch testcip in 40nm CMOS bulk process. The testchip operates at 2.38GHz at 1V nominal supply voltage and consumes 1.2W power. It offers 2.2X better throughput and 2.4X better energy-efficiency than published state of the art designs. We further evaluated modular crossbar networks with the proposed crossbar switches using BookSim2, a network on chip

evaluation tool. The proposed design achieves more than 90% saturation throughput with an internal speed up of 1.5, supports high data line rates, and offers lower average network latency compared to conventional crossbars. Evaluation results show that modular crossbars are scalable to high-radices while still offering high-performance, energy-efficiency and one-hop simplicity.

Acknowledgements

This thesis wouldn't have been possible without many wonderful people that I have interacted with and learned from during my time at CMU. They didn't only contribute greatly to my thesis and professional growth, but also made Pittsburgh truly my second home.

Ken Mai has been the most amazing and caring advisor that I could ever ask for. His vast technical knowledge, insightfulness, and approachability have made him an irreplaceable mentor, confidant, and a role model. He taught me how to conduct research properly, ask the right questions, and tackle challenges efficiently and creatively. My time as a graduate student has always been a blast, and I owe it to his guidance and continuous support that kept me motivated and feel accomplished.

Oracle labs offered their financial support, a safe haven to escape hot and humid Pittsburgh summers, and a unique opportunity to work with great mentors. I am indebted to Ron Ho for giving me this chance to collaborate closely with him and learn from him. His never ending cool ideas and ability to quickly identify and solve challenges always inspired me to learn more and become a better researcher. Jon Lexau has also been great mentor who made sure I was well taken care of and contributed generously to my testchip design.

My committee members James Hoe and Michael Papamichael were always gracious enough to share their knowledge and extend their support during my time at CMU. Their feedback and guidelines helped me explore new research areas and made this thesis more complete.

CMU was a great place to meet and learn from many wonderful teachers. Special thanks to Larry Pileggi for many interesting lectures as well as his continuous support and mentorship.

Judy Bandola, Shelley Phellps, Elaine Lawrence, Samantha Goldstein, and Nathan Snizasky, made sure I always had the best administrative support, encouragement and delightful conversations.

It was a pleasure to work with many great people in our research group including Mark McCartney, Mudit Bhargava, Yu Cai, Eric Menendez, Craig Teegarden, Weinan Ma, Yun Du, Burak Erbagci, Rachel Dondero, Nail Etkin Can Akkaya, Raymond Carley, Onur Kibar, and Prashant Mohan. Their invaluable contributions, help, and feedback for many tapeouts, papers, presentations, and this thesis are greatly appreciated. Special thanks to Mudit Bhargava, who has been generous enough to always guide me, Craig Teegarden, whose camaraderie made my rookie phase in grad school much more enjoyable, Rachel Dondero, without whom work and all-nighters wouldn't be as much fun at all, and Nail Etkin Can Akkaya for always sharing his tea with me.

I have also been lucky enough to have a great circle of close friends at work that might have lengthen our lunch times and increased the frequency of the coffee breaks. Ekin Sümbül, Berkin Akın, and Soner Yaldız have been great friends as well as great colleagues that made my time at CMU priceless. Also deserving a special note are Ömer Özdemir, Meriç İşgenç, Shadi Saberi, Gökçe Keskin, Emre Karagözler, Qiuling Zhu, Mats Forssell, Da-Cheng Juan for making work much more enjoyable.

Many great friends helped me push through graduate school and feel at home in Pittsburgh. Tuğçe Yüksel and Bekir Bediz were the best company during the harshest nights doing research as well as many fun nights. I was also extremely lucky to have Sercan Yıldız by my side during the final steps of thesis writing and job hunting, despite the times we tried to sabotage each other to keep at the same pace. Also notably Kevin Balko, Bilge Yümer, Ersin Yümer, Bülent Arda Gözen, Burçin Şimşek, Loubna El-Abbadi, Mert Terzi, Günay

Orbay, Onur Albayrak, and İdil Ulengin made this journey much more memorable. Last but not least, Melis Hazar, my partner-in-crime and 24-hour support mechanism, enriched my life in every possible way and I wouldn't have come out of this journey as who I am today without her.

Finally, I owe everything I accomplished to my family. I am the luckiest person in the world to have such an amazing, loving, and fun family. My parents, Tülay and Zeki Çakır, gave me everything I needed to be where I am today and always encouraged me to be anything I wanted to be. They are the best parents, best friends, and role-models I can ever ask for. My aunt, my older sister, my best friend, Nuray Bahadır, has always been my favorite teacher and always believed in me unconditionally. My grandparents, Endam and Kazım Bahadır, have been my most dedicated cheerleaders. My aunts, uncles and cousins, Güzelay and Ezgi Özdemir, and Ali, Eray, Emel, Özlem, Beliz, Ozan, and Berin Bahadır are my support system and favorite people. None of this would have been possible without my family, so I humbly dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Advances in process technologies continue to enhance the performance and available number of transistors on a single die with every new generation as predicted by Moore's Law [36, 55]. Historically, chip designers have focused on delivering smaller, faster, and more efficient single core processors aided by higher clock frequencies and sophisticated micro-architectures advancing single-thread performance. However, rising power density with scaling [63] has resulted in diminishing returns for further increasing the clock frequency with the end of Dennard Scaling [17, 18]. Moreover, single-threaded performance reached its limits with increasing overheads from micro-architectural complexities such as extensive pipelining of instruction streams. Thus, these limitations encouraged designers to explore system-on-chips with multiple cores to meet the growing demand for higher throughput and lower power.

Systems-on-chip exploit parallelism from multiple processing and storage units to improve system performance. With every new generation and increasing transistor budgets, the number of cores on a single die increases instead of the size and complexity of a single core, thus, further advancing the compute throughput from process scaling. However, as the

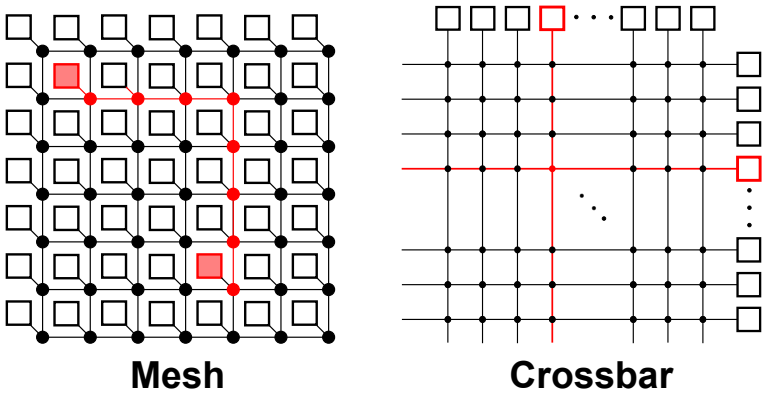


Figure 1.1: Mesh and Crossbar Topologies.

number of components in the system increases, designing the interconnection infrastructure of these systems becomes challenging [10]. In combination with growing wire delay with process scaling, communication has become one of the major factors that determines the system performance and power consumption.

Traditional buses have failed to meet the performance requirements of recent systems-on-chip with large number of cores. In spite of its low-cost and simplicity, shared bus connections cannot scale efficiently due to serial utilization. Networks on chip (NoC) emerged as an alternative by offering highly scalable, reliable, and modular communication platforms.

They are constructed from point-to-point data links (channels) and routers [4, 6, 16, 26] that transfer data packets from source to destination ports. Multi-hop networks like mesh (Figure 1.1) are easy to design and scalable to high-radices due to small router sizes and short channels. However, in these networks, data travels through multiple routers, where every router point computes the next-hop route and arbitrates multiple data packets. Hence, they have nonuniform latency and significant programming complexity. In contrast, crossbars are a single-hop, non-blocking network that offer all-to-all communication, with non-uniform latency, programming simplicity, and deterministic fairness [41, 53]. However, it is harder to design due to the large router switch that connects every I/O port.

Although crossbars have been quite popular for system-on-chip designs with low-tomedium

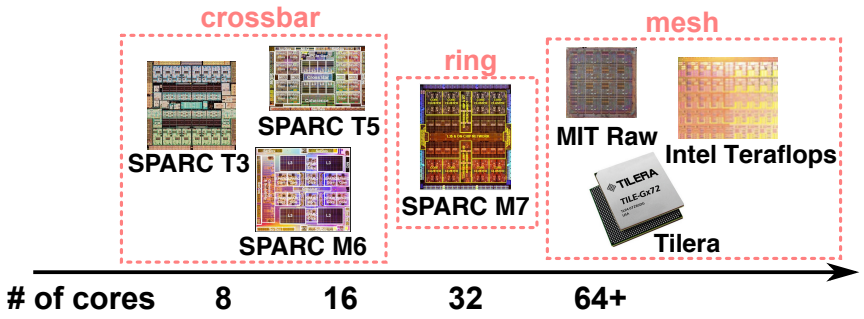


Figure 1.2: Networks-on-Chip in Recent Multi-Core Processors.

number of cores due to their simplicity, recent multi-core processor have replaced crossbars with multi-stage networks such as rings and meshes as seen in Figure 1.2 [3, 7, 47]. Designing high-radix crossbars is a challenge due to the quadratic cost scaling of the router. The crossbar interconnection switch, memory resources, and allocation logic typically scale poorly in area, power, and latency/throughput with increasing radix.

Improving crossbar scalability benefits many communication platforms ranging from system-on-chip and large-scale system networks to high-end Internet routers as the crossbar switch fabric is the main building block. High-radix crossbars can be used as fast, single-stage networks for large system-on-chips. As the use-case for such high-radix crossbars typically involves physically disparate input and output nodes (e.g., multiple processors communicating with multiple L3 cache blocks), these structures are usually built as a compact, centralized crossbar connected to the input and output nodes via dedicated point-to-point data links [41] as seen in Figure 1.3.

High-performance Internet routers and switches also commonly use one-hop high-radix crossbars as the switching fabric that determines the scalability and performance of the Internet. Further, high-radix crossbars can also be used as high-radix routers for large-scale networks for larger systems like super computers and datacenters. High-radix routers became critical in building large-scale networks with the emerging high-speed signaling technologies [23, 30] and increased available off-chip bandwidth [14]. Previous work [27] has shown that advancement in available off-chip bandwidth is best exploited by using high-

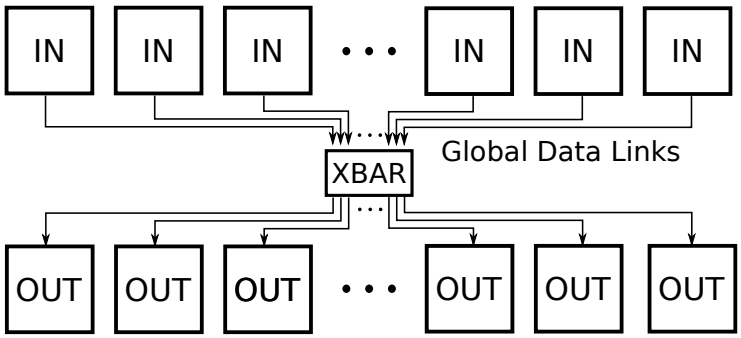


Figure 1.3: System view of a crossbar network.

radix routers with thin channels, rather than making ports wider. Networks built with high-radix routers reduce the hop count, leads to a lower network latency and cost. Therefore, there is significant effort to improve the scalability, performance, and energy-efficiency of crossbars.

1.2 High-Radix Crossbars

The crossbar network consists of the crossbar switch, I/O buffers, and allocators as seen in 1.4. The switch is the core of the network, offering all-to-all connectivity for I/O ports. The buffers are required to store the incoming and outgoing data in cases of conflicts, and allocators schedule and manage the network resources such as buffer availability and the switch traversal.

The main challenges in designing high-radix crossbars are the complex allocation, increased number of memory resources required, and the physical design of the crossbar switch itself. Previous work has shown that the scheduling of high-radix crossbars is a surmountable challenge [12, 27, 41, 64]. Further, as the network data packet size is usually larger than the physical data width of the crossbar, the scheduling decision latency can be amortized over the multiple cycles required to transfer the entire network packet across the crossbar.

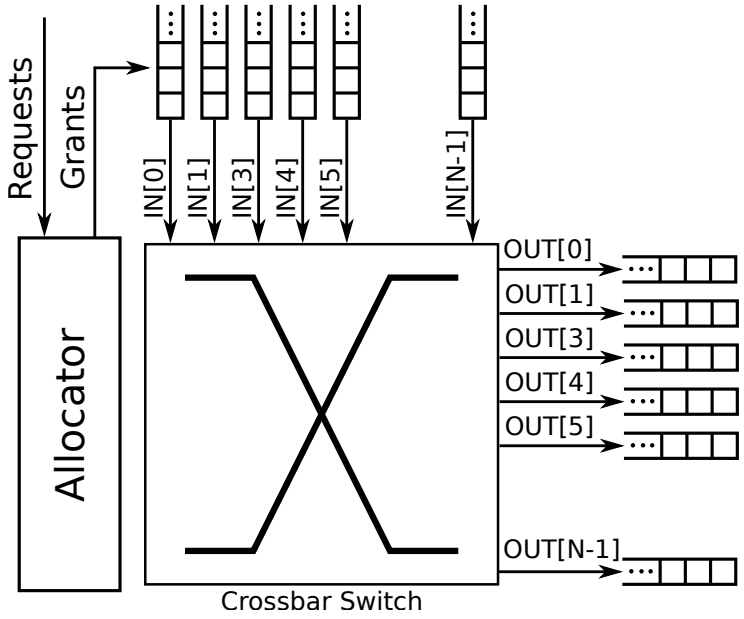


Figure 1.4: Crossbar Architecture.

On the other hand, crossbar switch performance plays a crucial role in high-radix crossbar network performance. Internal switch speedup simplifies allocation and enables high network saturation capacity with low memory cost. Therefore, high-performance switches can enable building efficient high-radix crossbar networks. However, designing a high-radix switch is a challenge. As number of inputs and outputs (radix) of the switch scales, the area grows quadratically (Figure 1.5), hence increasing the latency and energy consumption significantly.

In this thesis, we present a detailed switch design space exploration using a crossbar modeling tool to understand the scaling limitations better. We also discuss circuit and microarchitecture techniques to improve performance and energy-efficiency of high-radix crossbar switches. More specifically, we present a low-swing crossbar switch with high energy-efficiency, and a modular crossbar switch that can achieve linear performance scaling. Further, we present a scalable, high-radix modular crossbar that can offer high saturation throughput and low network latency using the high-performance modular switch.

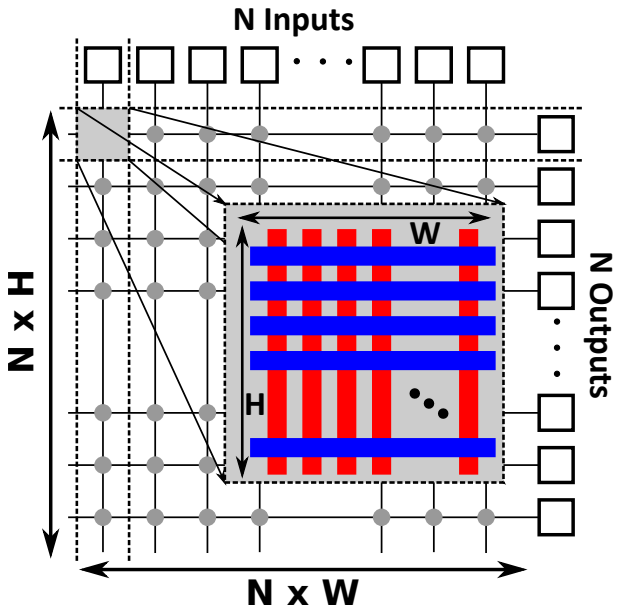


Figure 1.5: Crossbar switch area depends on radix (quadratically) and the I/O intersection width and height.

1.3 Dissertation Organization

The crossbar switch architecture is discussed in detail to understand the scaling limitations in Chapter 2. We present a crossbar switch modeling tool for fast design space exploration. We then discuss the trends for various design parameters at system, circuit, and layout level, and present guidelines on how to optimize design parameters for a high-performance, energy-efficient switch.

Chapter 3 explores low-swing signaling to further improve energy-efficiency of crossbar switches. We present a low-swing crossbar switch that uses capacitively coupled I/O wires and multiplexers. We evaluate performance, energy consumption, and area and discuss scaling of the proposed crossbar switch.

Chapter 4 introduces a modular crossbar switch that offers linear performance scaling and energy savings. We build high-performance and energy-efficient switch blocks using the design space exploration results. We then use these blocks to build modular high-radix crossbars that can perform better than monolithic switches. Details of the switch architec-

ture and evaluation are discussed in this chapter.

To validate improvements of our design, we designed and implemented a prototype modular crossbar switch testchip. Chapter 5 describes the testchip in detail and discusses the layout extracted simulation results.

Finally, the modular crossbar network architecture is presented in Chapter 6. The modular architecture consists of the modular switch, centralized allocators, and I/O buffers. The network performance is evaluated using a cycle-accurate NoC evaluation tool. The modular crossbar network offers high saturation throughput and lower network latency with the internal switch speedup.

Chapter 2

Design Space Exploration

In this chapter, we discuss details of the crossbar switch architecture and define the system, circuit, and layout level design parameters. Crossbar switch characteristics (area, power, performance) depend heavily on the floorplanning and layout, and hence the most accurate evaluation is achieved by post-layout or post-route simulations. This makes design optimization via iteration highly time and resource intensive, especially for full custom ASIC designs. Although standard cell designs have more flexibility to optimize system level parameters such as number of modules and data width, these designs lack low level circuit and routing optimizations due to EDA tool limitations.

To better understand the design space, we have developed an on-chip crossbar modeling tool based on analytical models calibrated using circuit-level simulation results in 40nm CMOS. We present a design space exploration showing how crossbar area, power, and performance vary across input/output node number, data width, wire parameters, and circuit implementation.

We also use the crossbar modeling tool (with added modifications) to explore our low-swing and modular crossbar switch ideas to improve energy-efficiency and throughput respectively in the later chapters. The tool allows fast design exploration for novel switch ideas with

different parameter constraints. It also allows evaluating the proposed designs at various radices to understand scalability and limitations.

2.1 Related Work

The related work most similar to that present here is from Passas *et al.*[41, 43] who present a cost analysis and modeling of crossbar switch area, delay, and power. Their main focus is the crossbar topology and the performance optimization for high radices across different radices and data width. They use standard cell design flow to build multiplexer tree based crossbars for the analysis and report post-route and simulation based results. Using standard cell design flow gives them flexibility to run an automated flow for post-route simulations, but their experiments are limited to multiplexer tree based designs and limited by electronic design automation (EDA) tool routing options. In contrast, our analytical crossbar model covers both system level (number of modules and data width) and circuit level (multiplexer circuits, wire pitches, etc.) design parameters to achieve design optimization as close to full custom design as possible. We further take into account custom layout optimizations that are not possible in the previous standard cell synthesis based modeling work.

As NoCs are in widespread use, other researchers have developed a number of on-chip network modeling tools for design exploration [4, 39, 40, 60, 65, 66, 67]. These tools contain relatively simple crossbar models, as crossbars are typically at the core of routers used in multi-hop networks. Further, these crossbars are typically of low-radix as they only connect the local network node to the rest of the multi-hop network. The crossbar modeling is part of the router modeling and is based on the basic X-Y crossbar design where the inputs and outputs run perpendicular to each other (usually using minimum sized wires) and connected via tri-state buffers at each intersection point. These studies focus on the modeling and optimization of the entire multi-hop network itself. Although they provide good system level modeling and exploration, the crossbar design options are not explored, the modeling

is at a relatively high level, and the radix of the crossbars modeled is relatively low.

Finally, in the circuit design space, researchers have proposed improvement techniques for X-Y topology crossbar switches [11, 45, 51, 52, 53, 54, 58, 69]. However, these are isolated point designs and there has not been significant exploration of the design space at the circuit-level owing to the iteration difficulties mentioned earlier. As such, our work allows for incorporation of new circuit techniques into our overall modeling and optimization framework. Indeed, the impetus for our modeling effort was our interest in designing high radix crossbar implementations, and our realization that there was not a modeling/optimization framework with sufficient level of detail and design options.

2.2 Crossbar Architecture

An N x N crossbar is a non-blocking switch that connects N input nodes to N output nodes, where each node has a data width (DW). The crossbar architecture is shown in Figure 2.1, where the connection of every input data to an output forms an N:1 multiplexer.

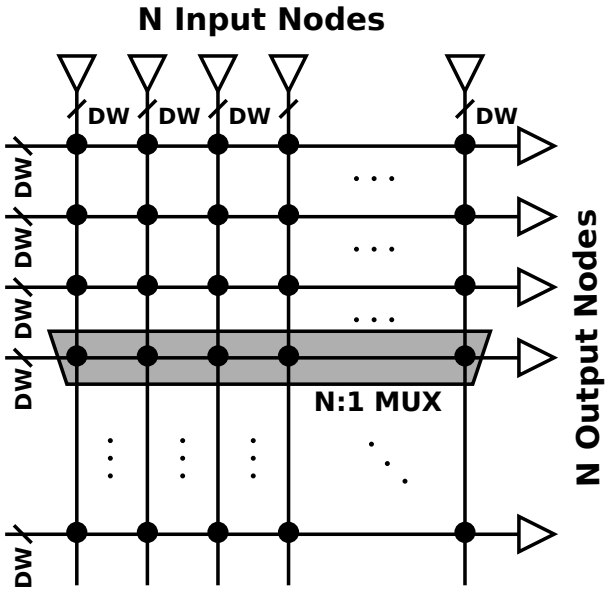


Figure 2.1: *N*x*N* crossbar architecture, where the connection of every input data to an output forms an *N*:1 multiplexer.

The crossbar consists of N x DW input data bits, N x DW output data bits, and N x DW multiplexers, N:1 each. As N and DW increases, floorplanning and the total design area become significant parts of the performance due to large number of long I/O wires, multiplexers, and challenging routing requirements. Physical design details such as floorplanning and area, as well as circuit details of multiplexers and I/O wires are discussed below.

2.2.1 Floorplanning

One challenge in crossbar design is to arrange I/O data wires for easier access to/from multiplexer gates from/to edge of the design. Floorplanning options are port-slicing and bit-slicing as can be seen in Figure 2.2. Grouping the data bits together for each module (port) is called port-slicing. The modules (module [0], module[1],...,module[N]) are placed next to each other where each module has DW data bits. This requires the output multiplexers to span the whole design width to get inputs from each module. On the other hand, grouping together a specific data from each module is called bit-slicing. In this case, data bits (data[0], data[1],...,data[DW]) are placed next to each other where each group has N data bits (one data bit from each module). This enables building smaller and faster multiplexers as inputs of each multiplexer are already grouped.

However, it is typical to assume that the I/O data bits are grouped together for each node and routed to/from the edge of the crossbar design. Therefore, for bit-slicing, a rearrangement of the I/O data bits at the edge of the design is required. For large number of I/O data bits, this bit scrambling requires a large area, and extra clock cycles [41]. Therefore, port-slicing offers better performance overall.

2.2.2 Area

Large number of wires, multiplexers, and the floorplanning details determine the area of the design, which is critical for the electrical parameters of the long I/O wires. Matrix-

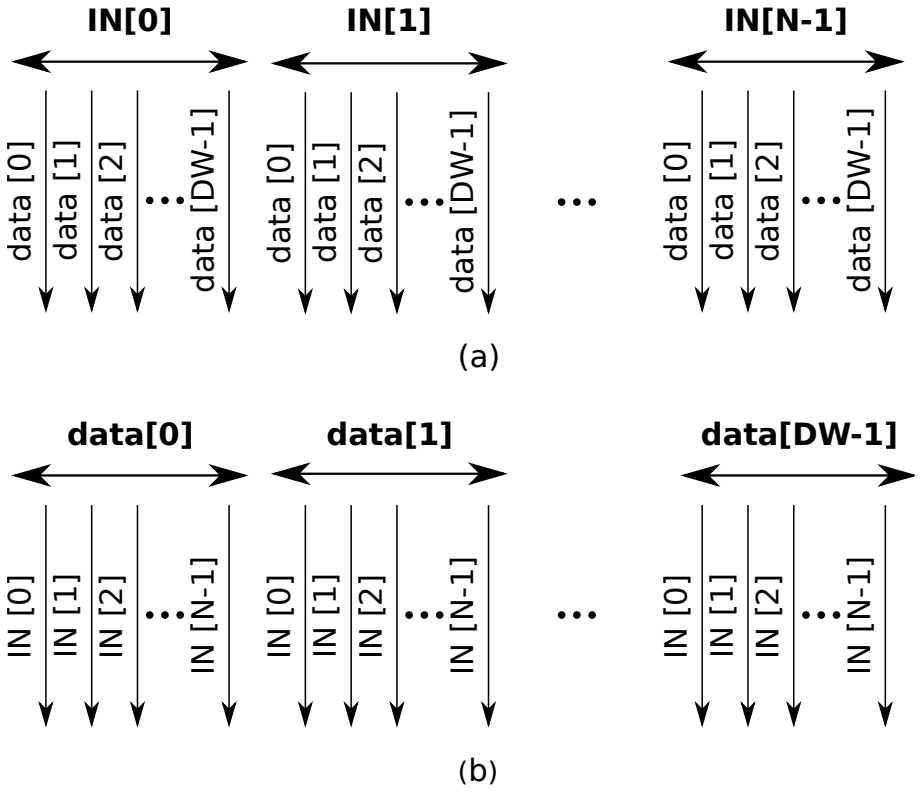


Figure 2.2: *N*x*N* crossbar with *DW* data bits implemented using port-slicing (a) and bit-slicing (b) floorplans.

style wiring, where input and output wires run perpendicular to each other, maximizes area utilization. Optimum size multiplexers usually fit underneath the minimum pitch I/O wires for smaller radix designs, whereas for high radices, using larger wire pitches can improve the performance by increasing the design area for the multiplexers (for better drive strength).

2.2.3 Input Wires

Each input bit should be distributed to every output multiplexer that means the input wires should span the whole height of the crossbar. In addition to quadratic latency growth of long input wires, fan out of the input bits increase as the design radix increases. Every multiplexer behaves as a constant load on input wires. Input drivers have more design flexibility as they can be stacked physically without a significant contribution to the overall core area and repeaters can be inserted when needed. However, the input wire delay is still a substantial portion of the overall delay at high radices.

2.2.4 Multiplexers and Output Wires

Switch circuits are multiplexers that drive the long output wires. Since we adopted port slicing, each multiplexer spans the whole design width to access all input nodes and drives long output wires. Internal wires of the multiplexers are not negligible, and this increases the delay by adding intrinsic wire delay and extra load capacitance from the wires. Further, as they are placed underneath the I/O wires to have a smaller crossbar area, the multiplexer area, hence the drive strength, is limited.

Architectural Implementation Options

Designing a N:1 multiplexer, where N is large, is a challenge due to parasitic loading from the non-active inputs. A popular solution is to build the large radix multiplexer using

smaller radix sub-multiplexers in a tree-structure to improve the performance. In a flat multiplexer implementation, a single horizontal wire track is dedicated for every output bit. But if the multiplexer is built as a tree, the number of horizontal wire tracks is increased to $\log_m N$ (the depth of the tree where m is sub-multiplexer radix) as seen in Figure 2.3. The length of the internal wires between the sub-multiplexers depend on the physical implementation choices as will be explained below. Further, increasing the depth of the multiplexer increases the design area, hence effecting the length and latency performance of I/O wires.

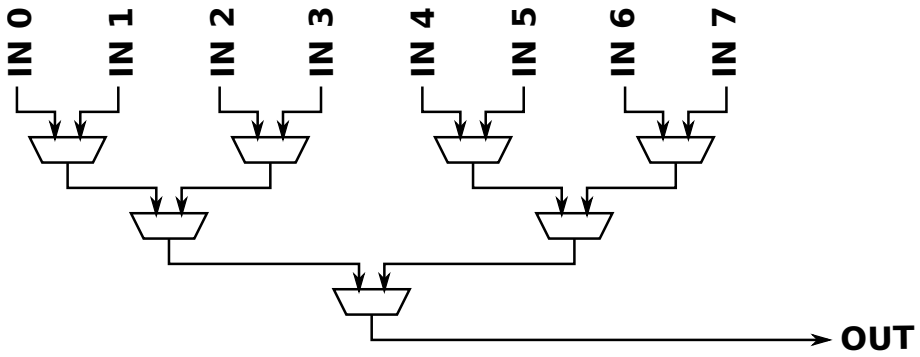


Figure 2.3: 8:1 multiplexer built using 2:1 sub-multiplexers.

Physical Implementation Options

As the inputs of the multiplexers span a large area, there are two main physical multiplexer implementation options: the distributed and centralized styles. The centralized style routes all the inputs to the multiplexer located in the middle and drives the output wire from that location, whereas in the distributed approach, the output wire spans all the input locations and the inputs tap-on to the output. Figure 2.4 and 2.5 show the centralized and distributed style implementations of an 8:1 multiplexer respectively. The centralized style minimizes the internal wires of the multiplexer, but it requires more horizontal wires dedicated to the input routing of multiplexers and suffers from larger area at higher radices. On the other hand, the distributed style has longer internal wires and cannot use unidirectional repeaters on the internal and output wires and suffers from quadratic wire delay scaling.

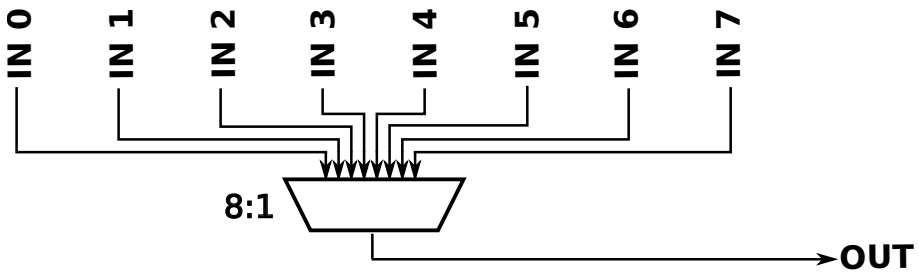


Figure 2.4: (a) Centralized physical implementation of a 8:1 multiplexer.

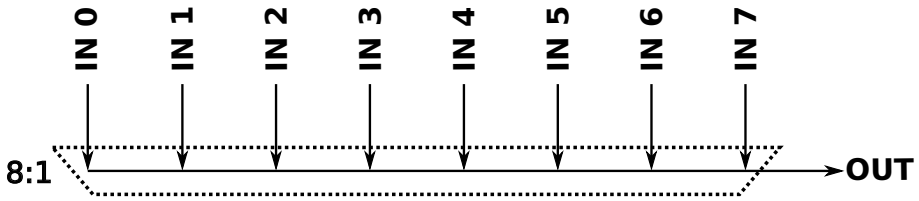


Figure 2.5: (b) Distributed physical implementation of a 8:1 multiplexer.

Circuit Implementation Options

The critical path of the crossbar is the data propagation through the longest input wire to the longest output wire, which means the switch circuit connects two high resistive paths and has high susceptibility to noise. Therefore static tri-state inverters are used to implement the multiplexers as they offer good performance due to low resistive paths to ground/power. The crossbar multiplexer is built by wiring together the tri-state inverters for each input module (different options of architecture and physical implementation are discussed above). Every input module has a select signal that is activated if the input is granted priority. Therefore, only the chosen input module's tri-state inverter drives the output wire.

We also explored using dynamic tri-state buffers for crossbar multiplexers. For hierarchical multiplexer implementations, precharging internal wires (sub-multiplexer outputs) are not feasible due to high switching energy, and extra wire routing for the precharge signal for each stage. But, it is an appealing option for flat multiplexer implementations since only the output wire is precharged and it avoids the large PMOS pull-up devices. The PMOS transistor used for the precharge can be sized smaller, since the unselected tri-state inverters

shares the same precharge signal and will also pull-up the output wires.

Figure 2.6 shows a static tri-state inverter (left) and a dynamic tri-state inverter. The static tri-state inverter drives the output wires when the input module is granted priority (SEL = 1). The dynamic tri-state inverter output is initially precharged high (PC = 0), and when in operating mode (PC = 1), the output is pulled-down if the input module is granted priority and the data is high.

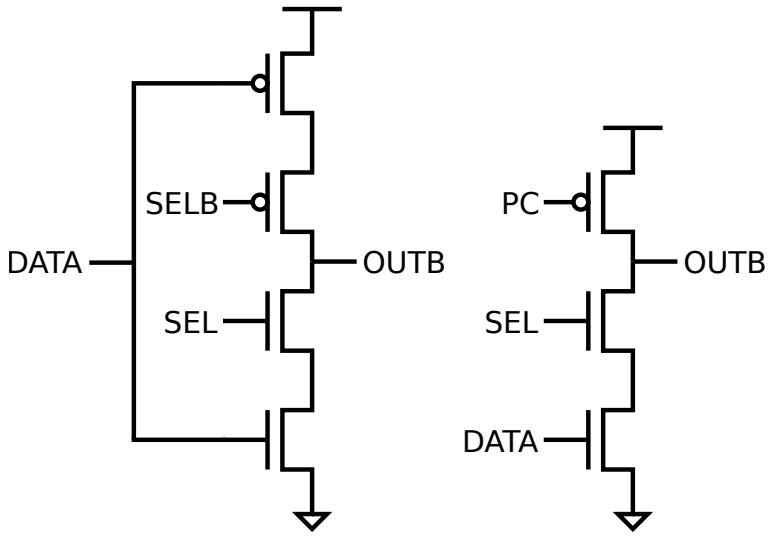


Figure 2.6: A static (left) and a dynamic (right) tri-state inverters.

2.3 Crossbar Modeling

We built an analytical model of the crossbar using design options explained in Section 2.2 as design parameters in Matlab. Models for the evaluation metrics (area, delay/throughput, and energy consumption) are calibrated using circuit-level, post-layout simulation results in 40nm CMOS bulk process (tt corner). Using the modeling framework, we present guidelines on how to optimize each parameter for minimum energy-delay product.

2.3.1 Design Parameters

The design parameters are listed below.

Number of I/O nodes (N): Radix of the crossbar.

Data width (DW): Number of data bits for each I/O node.

Input and output wire pitches: Sum of the wire width and spacing. Input and output wire pitches are independent from each other and should be optimized considering their electrical wire parameters as well as how they contribute to the total design area.

Number of metal layers: Number of metal layers dedicated for both input and output wires.

Sub-multiplexer Radix (m): Radix of the sub-multiplexers that are used to build the N:1 switch multiplexer.

Multiplexer physical implementation: Options are the centralized or the distributed styles as described above.

2.3.2 Evaluation Metrics

Evaluation metrics are area, delay/throughput, and energy consumption. Area is calculated using design parameters. Delay and energy consumption is modeled with design parameters as well as wire characteristics determined by the area and the floorplan.

Area

Floorplan of the crossbar design using port-slicing and matrix-style wiring can be seen in Figure 2.7. The area is dominated by the I/O wires and the switch circuits can be placed

underneath the wires. Details of a single switch block for an output port is shown as well.

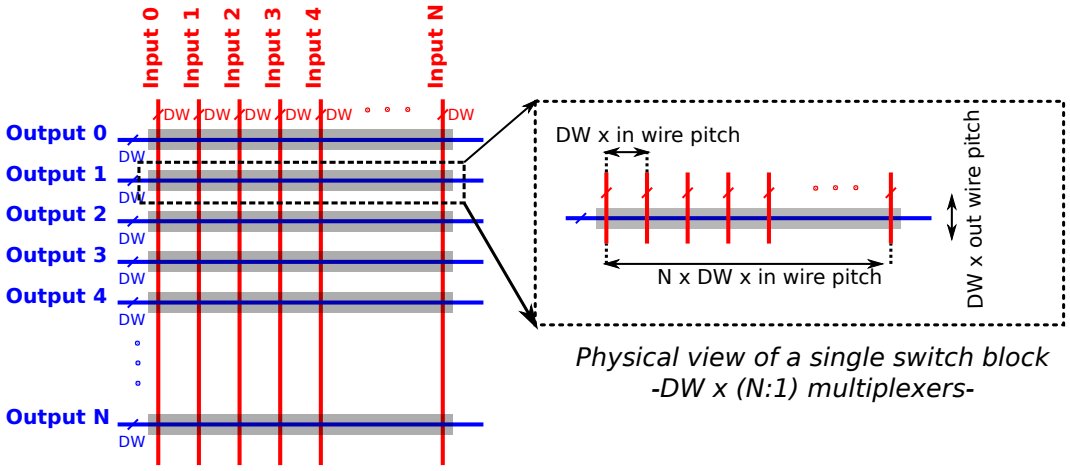


Figure 2.7: Floorplan of an *N*x*N* crossbar.

I/O wire length and design area are calculated as in Equation 2.1 and 2.2. Area is calculated using radix, datawidth, I/O wire pitches, and number of metal layers. Increasing the I/O wire width would decrease the wire resistance, however would increase the wire capacitance and design area. Input wire pitch determines the output wire length and vice versa. Therefore, it is important to understand the trade-offs for crossbar wire engineering, as the matrix style wire structure in the crossbar creates an interdependence of wire pitches and wire lengths. Further, area can be decreased by using more metal layers. However, smaller area also limits the size of switch circuits, hence limits the drive strength.

$$wire\,length_{i/o} = \frac{wire\,pitch_{o/i} \times DW \times N}{number\,of\,metal\,layers} \tag{2.1}$$

$$area = wire\,length_i \times wire\,length_o \tag{2.2}$$

Delay and Throughput

I/O wires play a substantial role in both delay and energy consumption. Wires are modeled as distributed RC lines [21]. Wire capacitance (C_{wire}, sum of four parallel-plate capacitances plus a fringe capacitance) and wire resistance (R_{wire}) are calculated using wire length, width, and spacing.

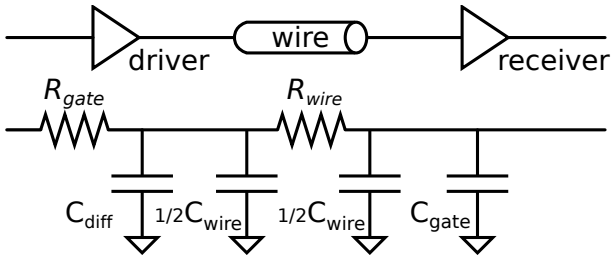


Figure 2.8: RC modeling of the driver, wire, and receiver for delay calculations.

Delay is modeled by RC formulation as seen in Equation 2.3. Drivers are modeled as resistors (R_{gate}) with parasitic loads (C_{diff}), and receivers present capacitive loads (C_{gate}). Delay for Figure 2.8 is calculated as in Equation 2.3.

$$\tau = R_{gate} \times (C_{diff} + C_{wire}) + R_{wire} \times (1/2C_{wire} + C_{gate}) \tag{2.3}$$

Input Wire Delay: Input data is distributed to every output multiplexer which behaves as a constant receiver load. Since receivers are distributed throughout the wire, parasitic loads contribute to the delay similarly as the distributed wire capacitance.

$$\tau_{input} = R_{gate} \times (C_{diff} + C_{wire} + N_o C_{gate}) + R_{wire} \times (1/2C_{wire} + 1/2N_o C_{gate}) \tag{2.4}$$

Multiplexer and Output Wire Delay: The multiplexers drive the long output wires and the delay is modeled for different physical implementation styles as shown below. If built in a tree-structure, each sub-multiplexer drives a wire since the internal wires are not negligible

and the final sub-multiplexer drives the output wire. The delay is the sum of all sub-multiplexer delays as in Equation 2.5 and 2.6.

For the centralized style, switch drivers (tri-state buffers) of the multiplexer are the wire drivers, and the driver parasitic load is not only the C_{diff} of the driver gate, but also the sum of all the unselected switch driver parasitic loads. If the input of the multiplexer (m) is larger than 4 (can be anything from 2 to N), we build it as quad-tree multiplexer using 4:1 multiplexers, since it is the most optimal way [61], and calculate the overall delay as the sum of the delay of each stage (as in Equation 2.5) .

$$\tau_{centralized} = R_{gate} \times (mC_{diff} + C_{wire} + C_{gate}) + R_{wire} \times (1/2C_{wire} + C_{gate}) \quad (2.5)$$

For the distributed style, switch drivers of the multiplexer are distributed throughout the output wire; therefore the parasitic load from the unselected drivers can be modeled as extra distributed parasitic loading on the selected driver as well as the wire resistance.

$$\tau_{distributed} = R_{gate} \times (mC_{diff} + C_{wire} + C_{gate}) + R_{wire} \times (1/2C_{wire} + 1/2mC_{diff} + C_{gate}) \quad (2.6)$$

Size of the network packet is usually larger than the DW. In order find the optimum DW, the throughput (TP) of the design is also evaluated.

$$TP = DW \times \frac{1}{delay} \quad (2.7)$$

Energy

Our energy modeling focuses on the switch capacitive current and energy consumption is modeled as in Equation 2.8 where C_{wire} is the sum of input, output and internal switching wire capacitances and C_{mux} is the sum of the gate and diffusion capacitances of the switching devices of the multiplexers for a single data bit.

$$energy = N \times DW \times (C_{wire} VDD^2 + C_{mux} VDD^2) \tag{2.8}$$

2.3.3 Methodology

For a given N, we calculate area, delay, and energy using the models explained above in a flow shown below. In order to find the optimum design point, the flow is repeated for every possible set of design parameters.

1. Input N
2. Set the design parameters : DW, number of metal layers, input and output wire pitch, sub-multiplexer radix, multiplexer physical implementation style
3. Calculate I/O wire lengths and design area
4. Placement of multiplexers: Multiplexer tree is structured and each sub-multiplexer is placed.
5. Calculate wire parasitics for I/O wires and the internal wires for the multiplexers: Length of the internal wires are calculated from locations of the current and the next sub-multiplexers.
6. Size the multiplexers:
    - If built as a tree, sizing starts from the last sub-multiplexer.

- Multiplexer is sized according to optimal fan out (FO) sizing [61] where load is gate capacitance of the next multiplexer and the wire capacitances.
- Each multiplexer in the tree has an allowed area that is decided by design area and number of multiplexers in the tree.
- If the sized multiplexer does not fit in that area, FO is increased until the multiplexer is small enough.

7. Calculate delay and energy for multiplexers and output wires: Sum of delay and energy of all multiplexer stages.
8. Calculate delay and energy for input wires: Repeaters are inserted if necessary.
9. Calculate the total delay and energy

2.4 Evaluation

We run this flow for different sets of design parameters to understand the behavior of each parameter and the optimal design points. While sweeping a specific parameter, the rest of the parameters are set to the most optimal solution for that specific design point. First, we explore sub-multiplexer radix and the physical implementation style that lead us to an optimum design option. Using this design point, we look into the trends for the wire pitch and the number of metal layers. Further, we examine the effects of DW on performance and changes with radix scaling. Finally, we highlight a design point optimized for minimum energy-delay product and compared its performance to previously published results.

2.4.1 Sub-multiplexer Radix

The sub-multiplexer radix (m) is swept from 2 (binary tree) to N (flat) for both the centralized and distributed implementation styles to find the optimum design point. Figure 2.9,

2.10, and 2.11 show comparisons of delay, energy, and area of centralized and distributed styles with different sub-multiplexer radices.

For the centralized style, the most optimal implementation is to use a binary-tree multiplexer (m =2). The multiplexer is located in the middle where all the distributed inputs are driven to, therefore every stage of multiplexers have $m/2$ extra horizontal wire tracks dedicated for input wires. This results in a larger area, higher energy consumption, and delay with increasing m.

On the other hand, for the distributed style, the best performance can be achieved using a flat multiplexer ($m = N$). This implementation does not have extra wire tracks for the inputs that limits the m, and since the distributed inputs are connected via long wires, the parasitic loading from the other inputs are negligible.

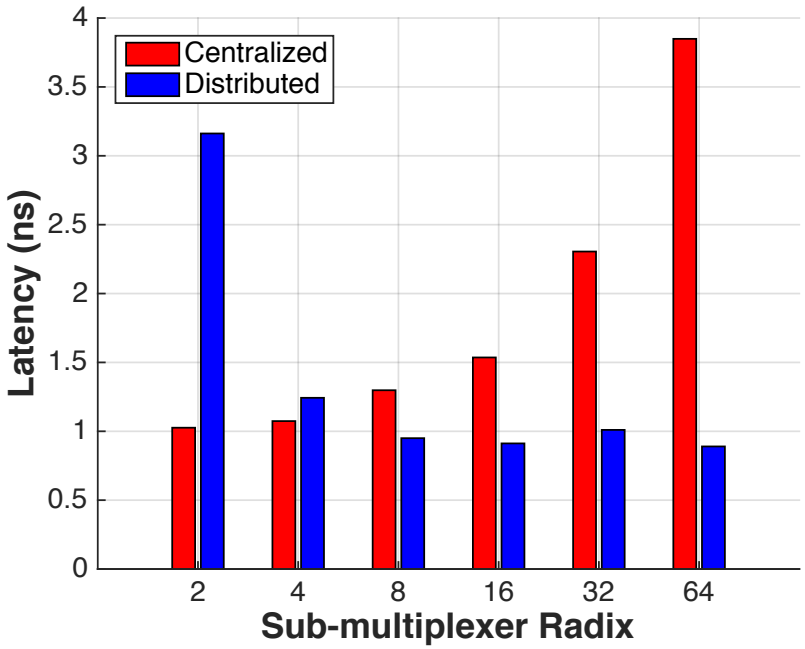


Figure 2.9: Latency of centralized and distributed styles with different sub-multiplexer radices.

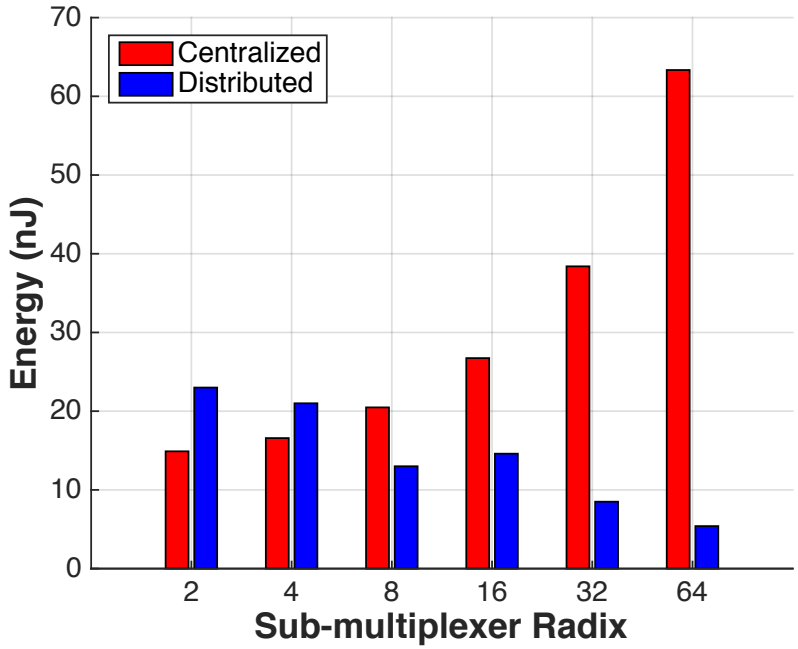


Figure 2.10: Energy of centralized and distributed styles with different sub-multiplexer radices.

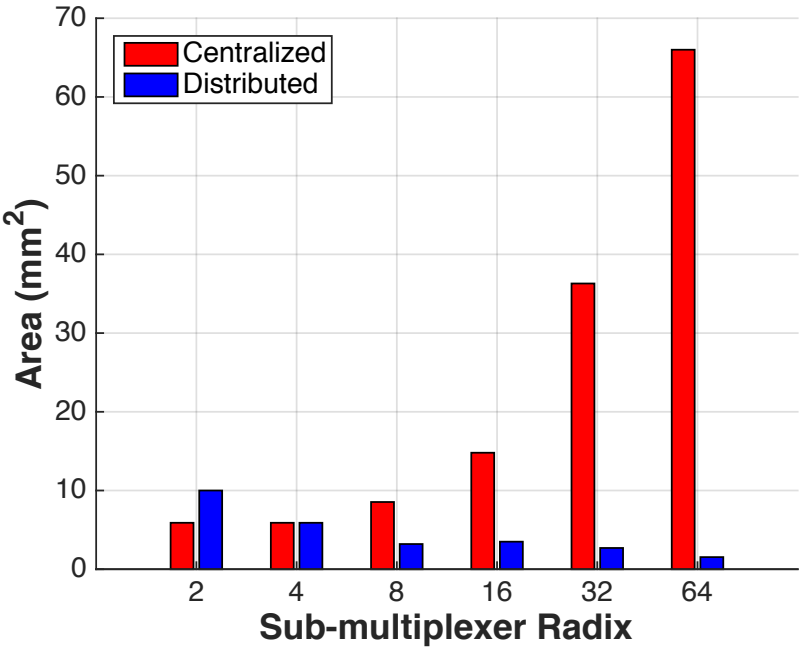


Figure 2.11: Area of centralized and distributed styles with different sub-multiplexer radices.

2.4.2 Physical Implementation Style

We compared the performance of different physical implementation styles. The centralized style is implemented using a binary tree multiplexer, whereas the distributed style is implemented using a flat distributed multiplexer.

For the distributed, flat multiplexer, there is a single output wire that all inputs are connected to. As mentioned in the previous section, this makes it possible to use dynamic logic to build the multiplexer. The switch connections of the inputs are built as pull down circuits, and pre-charge pull up devices are distributed on the output wire (number of pull up devices depend on the wire capacitance as well as the timing requirements). This style decreases the circuit area significantly by removing large PMOS devices in the tri-state buffers, however introduces additional timing requirements because of the pre-charge scheme.

Figure 2.12, 2.13, 2.14 show comparison of delay, energy, and area of centralized, distributed, and distributed-dynamic styles for a radix-64 crossbar with 64b of data. As can be seen the distributed-dynamic style offers the best performance in terms of delay, energy, and area.

Figure 2.15 shows and example of an 2x2 1b crossbar with distributed-dynamic tri-state buffers. Figure 2.16 shows the circuit details and timing of the 2x2 crossbar. Inputs are distributed on the high-phase of the clock while outputs are precharged and the outputs are evaluated on the low-phase of the clock.

2.4.3 Wire Pitch

Increasing the input and out wire pitches increases the area as well as the energy consumption. Increasing the design area improves the switch driver strength, hence performance, but also increases the wire capacitance that potentially hurts the performance. The delay for different input and output wire width and spacing can be seen in Figures 2.17 and 2.18. Both the input and output performances are more sensitive to wire width because increas-

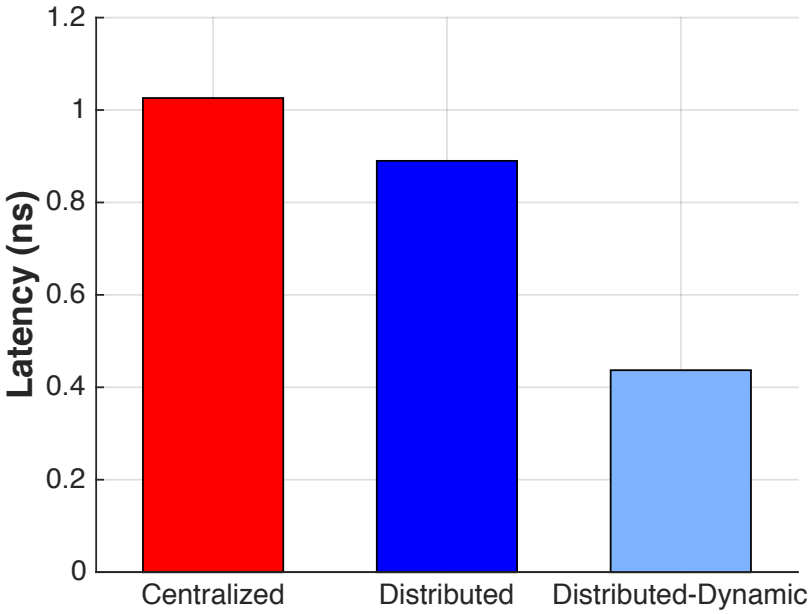


Figure 2.12: Latency comparison of of centralized, distributed, and distributed-dynamic styles.

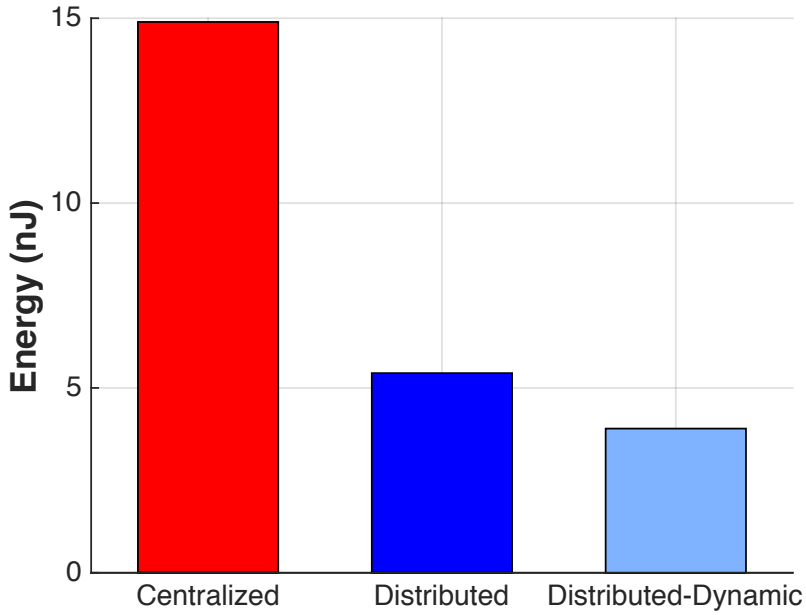


Figure 2.13: Energy comparison of of centralized, distributed, and distributed-dynamic styles.

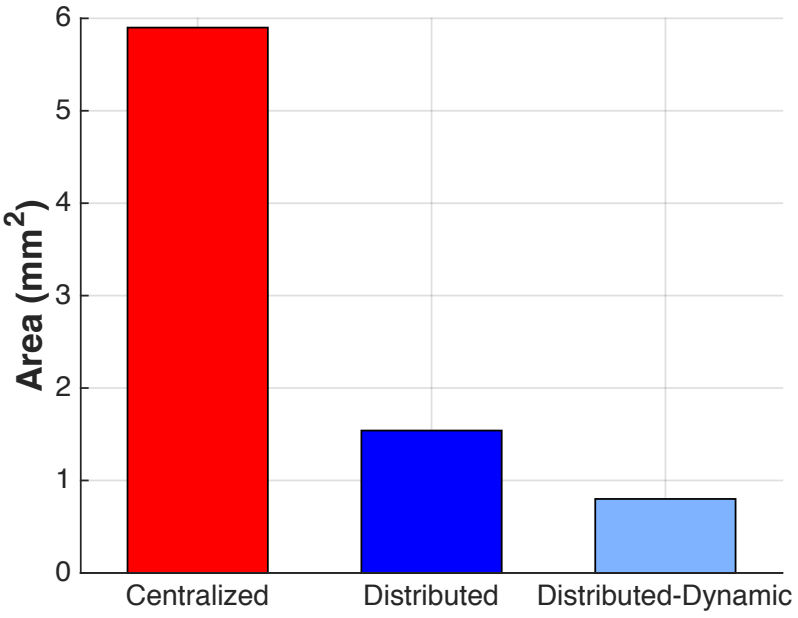


Figure 2.14: Area comparison of of centralized, distributed, and distributed-dynamic styles.

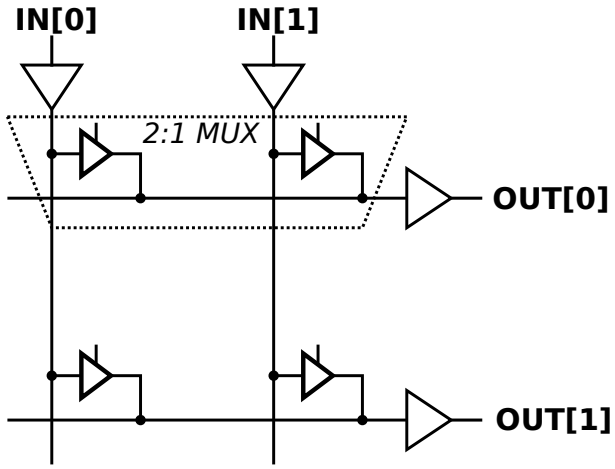


Figure 2.15: 2 x 2 1b crossbar architecture with distributed-dynamic multiplexers.

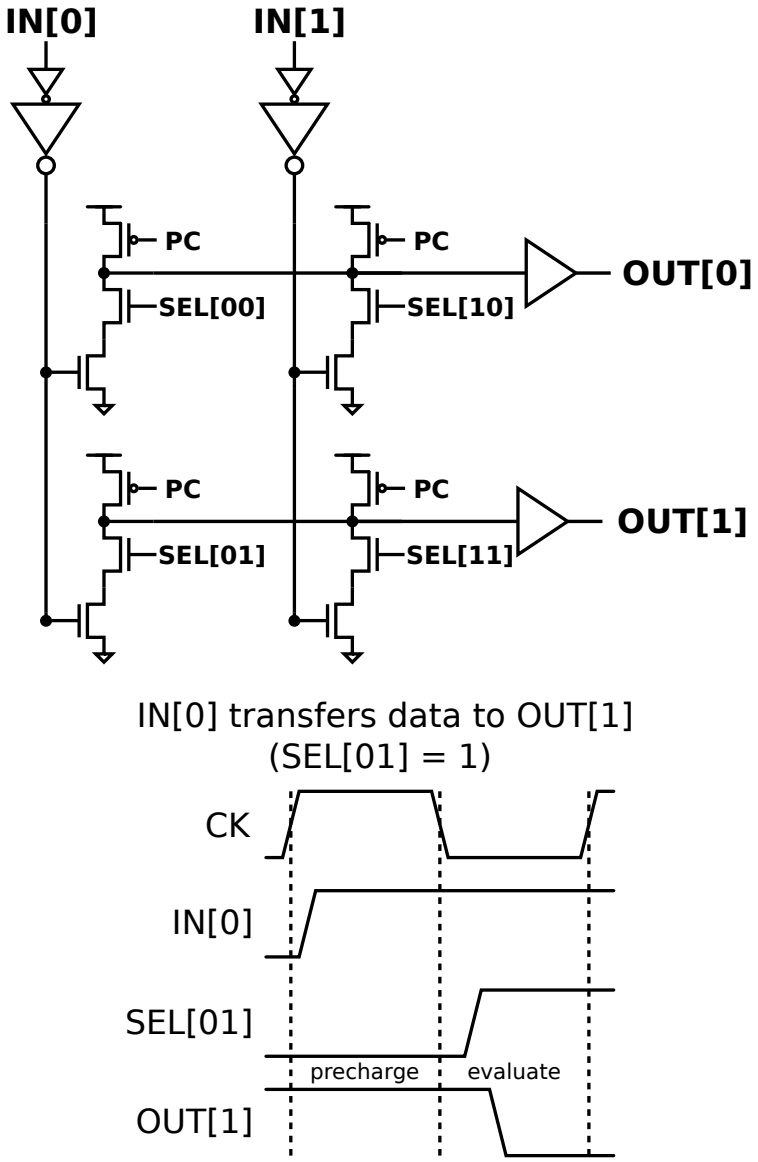


Figure 2.16: 2 x 2 1b crossbar circuits.

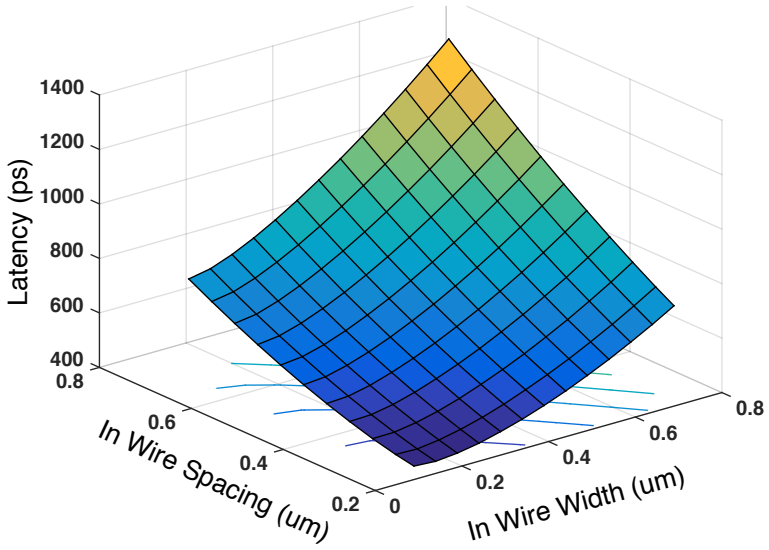


Figure 2.17: Delay for different input wire width and spacing.

ing the wire width decreases resistance of that wire, improves the switch driver strength, but increases the capacitance of the corresponding wire as well as the opposite wire. On the other hand, increasing the spacing improves the driver strength, and only increases the opposite wire capacitance.

The optimum input wire width and spacing are the minimum values. The optimum output wire spacing is the minimum as well, however the optimum output wire spacing is ~ 6 times more than the minimum spacing. Increasing the output wire spacing increases the output driver area while increasing the input wire capacitance. But since the input drivers have more flexibility in terms of area and drive strength, the overall delay performance is improved with increasing output spacing.

2.4.4 Number of Metal Layers

Increasing the number of metal layers dedicated for input and output wires decreases design area linearly. But it also decreases switch driver area; therefore performance improvement saturates when drive strength of the switch circuits degrades significantly and it becomes physically impossible to fit minimum size transistors. Figure 2.19 shows energy-delay prod-

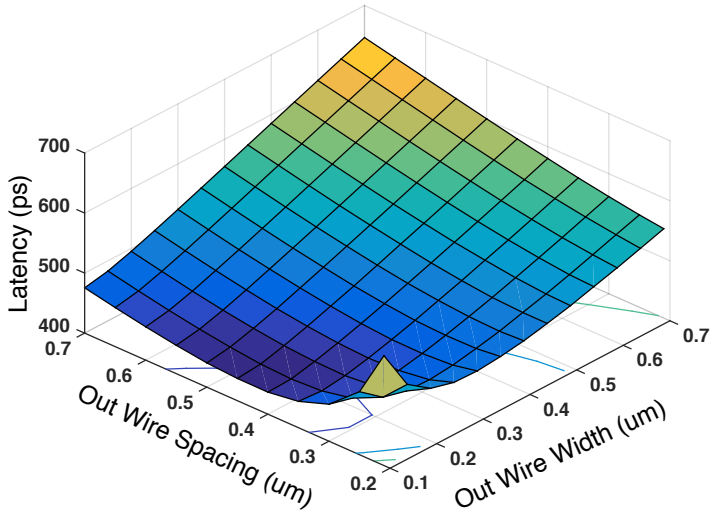


Figure 2.18: Delay for different output wire width and spacing.

uct for different number of metal layers for a radix-64 crossbar with 64bits of data. Using 2 metal layers for each input and output wires (4 layers total) improves energy-delay product ~ 4X. Although using more metal layers improve performance further (at a lower rate), it requires more than 6 layers dedicated just for the high level routing of the crossbar, which is a big portion of wire resources for current technologies.

2.4.5 Data Width

Area and energy grows exponentially with data width (DW) scaling. Delay, on the other hand, grows at different rates for different DW values for crossbars.

Increasing the DW increases switch circuit area (improves drive strength) and IO wire lengths. When DW is small, constant overhead delay (timing margins for flip-flops and dynamic logic) and output parasitic loading of the switch is a bigger portion of the delay. Therefore, delay increases at an even lower rate than DW scaling. As DW gets larger, this rate becomes linear, and quadratic when unrepeated wire delay dominates.

throughput (TP) would stay constant with DW scaling if delay would scale linearly. Opti-

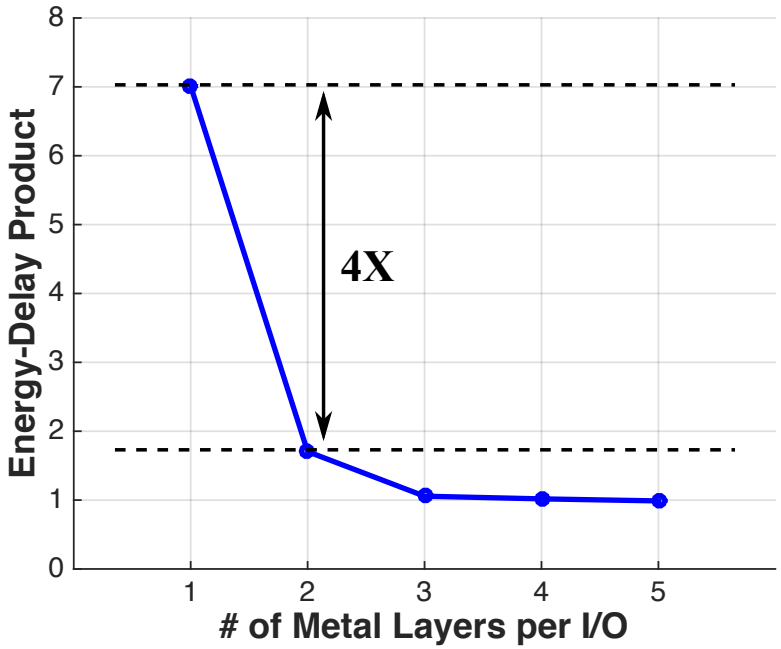


Figure 2.19: Energy-delay product vs number of metal layers.

mum design point is where delay starts scaling worse than linear. As shown in Figure 2.20 TP increases until the optimum point and declines afterwards. Furthermore, optimum DW point changes with radix scaling as well. Optimum DW is larger for smaller radices and vice versa as seen in Figure 2.20 as the main bottleneck of the crossbar design is total area determined by N and DW.

2.5 Summary

In this chapter, we present a modeling tool for crossbar design to have a better understanding of the design space and parameters. Our design space exploration suggests that smaller designs offer better performance due to shorter input, output and internal wires. Both input and output wire widths should be minimized to decrease the design area and wire capacitances. Wire spacing, on the other hand, can be increased to improve the multiplexer drive strength. Smallest switch option that offers the best performance is flat multiplexers

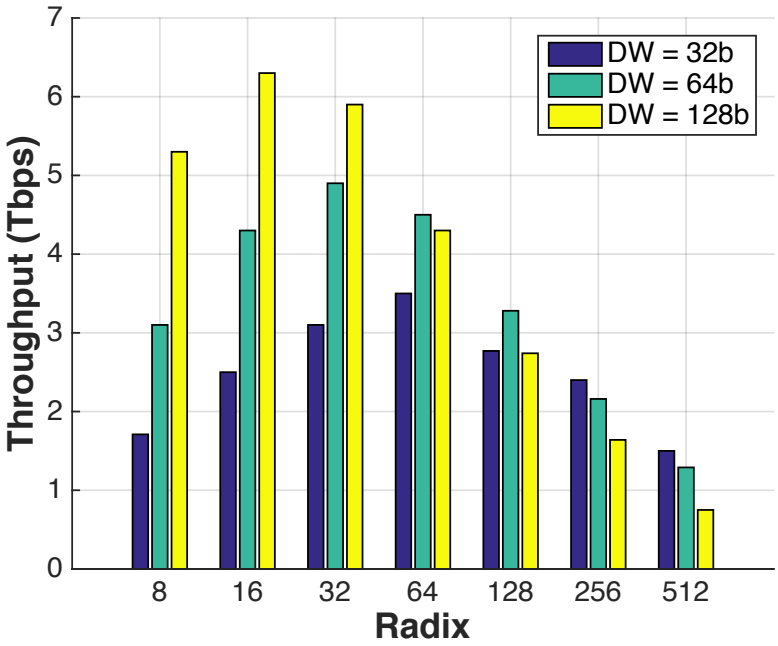


Figure 2.20: Throughput for different data widths and radices.

built with domino-style tri-state buffers. The optimum data width of the design depends on the radix and should be chosen to keep design area reasonable. Next chapter will explore low-swing signaling to improve energy-efficiency of crossbar switches and present a low-swing crossbar switch design using capacitive coupling.

Chapter 3

Low Power Crossbar Circuits

In this chapter, we explore low-swing signaling to improve energy-efficiency of crossbar switches. Switching energy of long I/O wires are one of the major contributors to the high energy consumption of crossbar switches. [58] presents an energy efficient crossbar that uses low-swing signaling by using early detection as in SRAM bit lines, and [29, 45] demonstrates another crossbar with reduced voltage swings using dual voltage supply, differential, reduced-swing drivers. We propose a low-swing crossbar design that uses capacitively driven wires [22, 56] and capacitively coupled multiplexers [8]. The long input wires are driven with series coupling capacitances that pre-emphasize transitions to reduce wire delay and to reduce the load seen by the driver. It also lowers the voltage swing without the need of a second voltage supply, thus, offering low swing signaling, higher bandwidth, and low energy. A capacitively coupled multiplexer uses a similar drive scheme for the inputs to drive a common node. A particular advantage of this scheme is that the un-selected inputs present a lower parasitic loading because the coupling MOS capacitors used to drive the wire are off, and thus have a lowered capacitance.

3.1 Low-Swing Crossbar Switch Design

Our low-swing crossbar switch design adapts matrix-style wiring where input and output wires run perpendicular to each other (inputs run vertically whereas outputs run horizontally) to minimize the design area. Further, as common practice, the I/O data for each module is routed to/from the edges of the crossbar design. Thus, I/O data for each module is also grouped within the design (also called port-slicing) in order to avoid extra I/O scrambling at the edges. However, this means that input wires span the entire height of the design to distribute the data to every multiplexer, and multiplexers span the entire width of the design. Distributed inputs of the multiplexers (from each input module) are connected to the output wires via switches to form the multiplexers.

Even for minimum pitch wires and low-radix crossbar designs, the wire parasitics are significant. The quadratic wire latency increase can be mitigated by repeater insertion, but repeaters come at an energy and area cost. In order to improve both latency and energy consumption, we propose to use capacitively coupled wires [22, 56]. Another major bottleneck for crossbars is the needed multiplexer structures. Designing an N:1 multiplexer, where N is large, is a challenge due to parasitic loading from the unselected inputs. Further, the multiplexers are required to drive the long output wires. Capacitively coupled multiplexers can improve the performance of high-radix multiplexers where the un-selected input capacitances are significantly smaller than the active input capacitance and driving the output wires capacitively. The idea and the implementation as the crossbar switch are also explained in detail in this section.

3.1.1 Capacitively Coupled Wires

Driving a long wire through a series coupling capacitor as seen in Fig. 3.1 reduces the signal swing through a capacitive voltage divider [22, 35, 49, 56]. While a conventional driver drives the coupling capacitance with a full-swing voltage transition (at node A), the

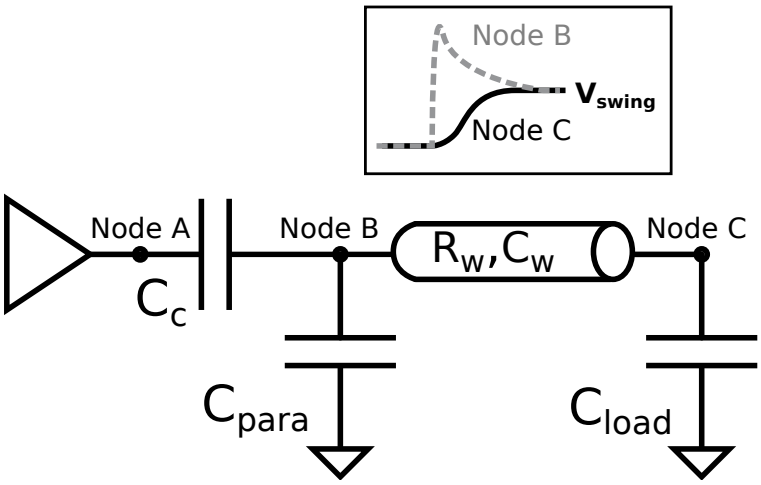


Figure 3.1: A capacitively driven wire scheme.

very end of the wire (node C) only receives a reduced voltage transition due to the capacitive divider formed by the coupling capacitance, the wire capacitance, and the load and parasitic capacitances as can be seen in Equation 3.1. Node B will initially overshoot and then settle to V_{swing}, while node C shows a rapid rise to V_{swing}. The coupling capacitance can be sized depending on the desired voltage swing on the wires.

$$V_{swing} = V_{dd} \frac{C_c}{C_c + C_w + C_{para} + C_{load}} \tag{3.1}$$

One of benefits of this scheme is the latency improvement through pre-emphasis and the reduced energy consumption. This is faster than an inverter driving a wire because the driver sees only C_c not the C_{wire}, and the charge redistribution across the wire cuts the wire delay approximately in half as seen in Equation 3.2. For our implementation, we sized the coupling capacitance for twice the desired reduced voltage swing and sense when the voltage drop reaches the desired voltage swing.

$$\tau \approx R_{driver}(C_c + C_{driver,para}) + R_w C_w / 4w \tag{3.2}$$

Further, the wire switching energy improves linearly with the reduced voltage swing without requiring a second voltage supply as seen in Equation 3.3, and the overall energy consump-

tion also improves significantly with the reduced size of the drivers since it only sees the C_c. The required differential wiring and the sense amplifiers to restore the low-swing signals reduces the total energy savings marginally. For example, a 10mm long capacitively driven wire with 1/9th of Vdd swing offers 3.8X better energy savings compared to long full-swing buses optimized for lowest energy-delay product in 180nm CMOS bulk process [22] .

$$Energy_{wire} = C_{wire} V_{swing} Vdd \tag{3.3}$$

We designed capacitively coupled input data transmitters and input receivers for the low-swing crossbar design.

3.1.2 Capacitively Coupled Input Data Transmitters

Capacitively coupled input data transmitters are designed to drive the long input wires and the detailed circuit is shown in Fig. 3.2. The transmitter consists of differential input generator, NAND gates for data enable and energy reduction, drain-source connected PMOS gates as the series coupling capacitances, and the PMOS gates as the precharge devices for DC biasing.

The reduced voltage swing comes with reduced noise margins. In order to reduce the crosstalk from the neighbors and the complementary data signals, twisted and interleaved differential wires are used to distribute the data [22, 46]. The overhead of using twice the wire resources for the differential wiring is reflected in the total design area.

Drain source connected PMOS transistor acts like a nonlinear capacitance. The gate-to-source and gate-to-drain capacitances of the PMOS are higher when the transistor is enhancement mode due to the channel formed underneath the gate. Therefore, the drain source connected PMOS has a high capacitance when the device is turned-on. The output port of the PMOS capacitance is biased high at Vdd. If the data is not enabled, the output of the NAND gate is also high, hence no transition on the input data wires. But when the

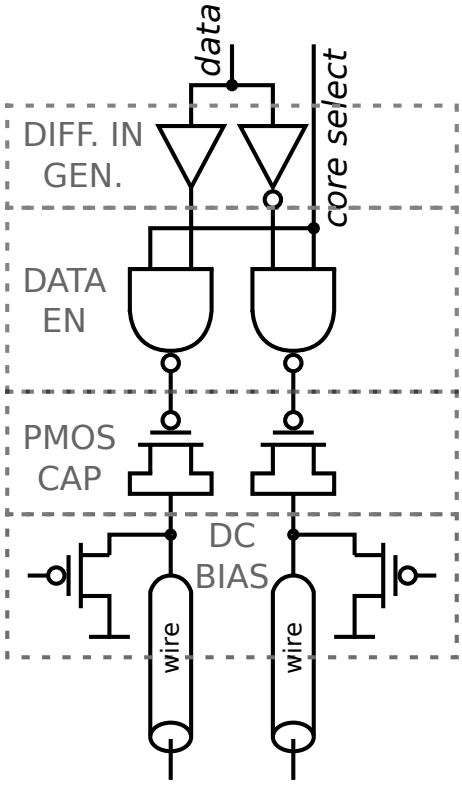


Figure 3.2: Capacitively coupled input data transmitters.

data is enabled, the high-to-low transition on the NAND output propagates through the input wires via the on PMOS capacitance. The size of the capacitance can be adjusted by changing the width and length of the transistor.

The data wires need a path to a DC source to have the proper DC bias. We adopted a precharge scheme where the input and output data wires operate on different phases of the clock and precharged when not in use. The precharge transistors are distributed throughout the wire and sized accordingly to meet the timing requirements.

3.1.3 Input Data Receivers

Data for every input module is received by the output multiplexers. The receivers present a parasitic load capacitance on the input wires (total load scales linearly with N), and this is also included in the latency and energy calculations and the corresponding input transmitter sizing. StrongARM sense amplifiers are used as the receivers to restore the full-swing data from the low-swing data signals. Symmetric and mirrored layout topology is adopted to minimize the systematic offset and the NMOS transistors of the cross-coupled inverters are sized large enough to have the desired random offset (calculated using Monte Carlo simulations). The offset of the sense amplifiers is important as it limits the minimum voltage swing on the input wires. Input receivers with gated sense enable signals are used to build the capacitively coupled multiplexers for our low-swing crossbar design.

3.1.4 Capacitively Coupled Multiplexers

Capacitively coupled multiplexers use nonlinear capacitances as the switches to drive the long output wires capacitively and exploit the fact that the capacitances of unselected inputs are significantly smaller than the active input capacitance. This enables building fast and efficient multiplexers that can drive long wires with a reduced voltage swing.

The drain-source connected PMOS capacitances are used as the nonlinear capacitances. As

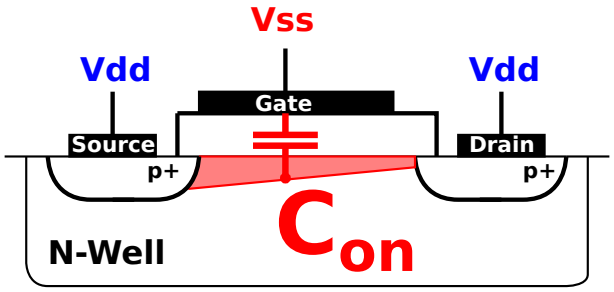


Figure 3.3: The ON capacitance ratio for drain-source connected PMOS.

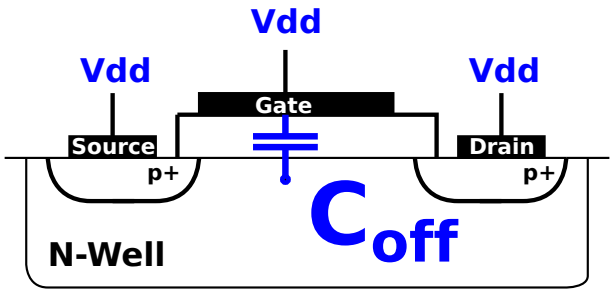


Figure 3.4: The OFF capacitance ratio for drain-source connected PMOS.

mentioned above, when the PMOS transistor is turned-on, the gate capacitance is large due to the formed channel (Figure 3.3). However, when the transistor is turned-off, the gate capacitance is to the bulk instead of the channel, hence significantly smaller (Figure 3.4). Simulation results in 40nm CMOS bulk showed that the ratio of ON capacitance to the OFF capacitance can be larger than 10X when longer lengths (>5μm) are used for the PMOS transistors. Simulations and our modeling also show that the ratio is more sensitive to the length (L) of the transistors than the width (W) of the transistor, since the on capacitance is strongly proportional to both W and L, whereas the off capacitance is mostly proportional to W of the transistor.

This scheme improves the energy savings by decreasing the coupling capacitance required for the desired voltage swing as well as the savings from the smaller driver and low-swing wires. Assuming load and parasitic capacitances are much smaller than the wire and the coupling capacitances in Equation 3.1, for N:1 multiplexer, the reduced voltage swing can be represented by Equation 3.4 using a nonlinear coupling capacitance (C_{on} and C_{off} depending on the state of the transistor).

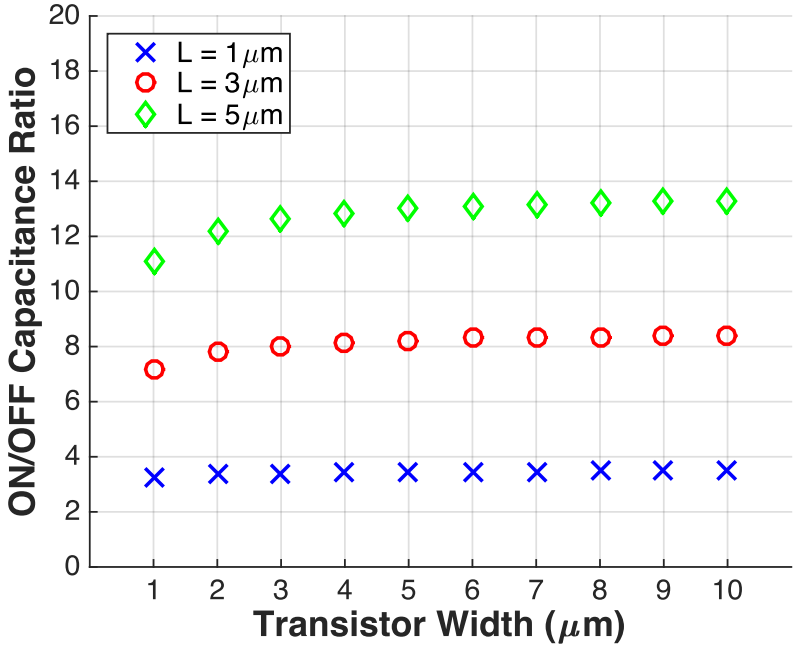


Figure 3.5: The on to off capacitance ratio for drain-source connected PMOS.

$$V_{swing} = V_{dd} \frac{C_{on}}{C_{on} + (N-1) * C_{off} + C_w} \tag{3.4}$$

Similar to the scheme used in input data transmitters, NAND gates drive the PMOS capacitances that are connected to the output wire to form the capacitively coupled multiplexer as seen in Fig 3.6. The drain-source of the PMOS capacitance is biased at Vdd. The selected NAND gate's output transitions from high-to-low that turns on the PMOS capacitance, hence achieves a large capacitance to drive the output wire. On the other hand, the unselected NAND gates' outputs stay at Vdd that keeps the PMOS transistors turned-off, hence smaller parasitic load capacitances on the output wire. The output wire is driven via a series capacitance that is the PMOS capacitance of the selected input.

The low-swing output wires also required to be differential and requires twice wire and circuit resources. Only one of the differential data is driven via the series capacitance while the complementary data stays high, hence not increasing the switching energy consumption. This scheme offers as much energy savings as the capacitively coupled wires.

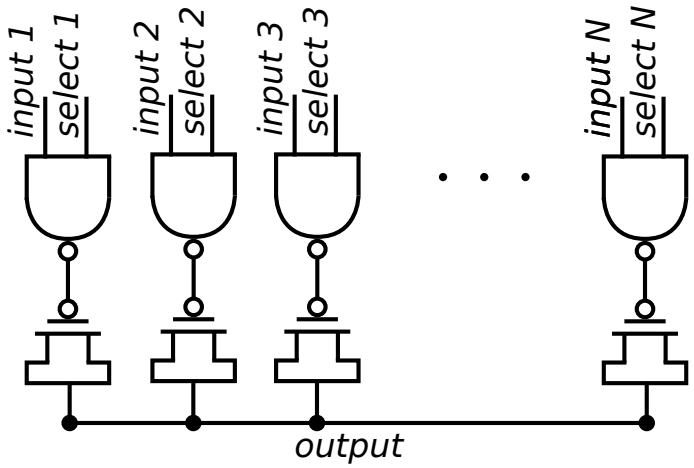


Figure 3.6: Capacitively coupled multiplexer.

Further, this scheme improves the delay performance due to decreased parasitic loading from the off paths. We compared the latency performance of the capacitively multiplexer to a conventional multiplexer for a crossbar design using 40nm CMOS bulk. The capacitively multiplexer is built to achieve a voltage swing of 200mV (sensed at 100mV) with the corresponding PMOS sizing. Clocked StrongARM sense amplifiers are used to restore the full-swing output data, therefore an extra 30ps (for sense amplifier) and 10% of clock period (for clock skew) is added to the latency of the capacitively coupled multiplexer. On the other hand, the conventional multiplexer is built using tri-state inverters. The inputs for both multiplexers are assumed to be distributed horizontally and the multiplexor switches drive the long output wire as in a crossbar design.

We ran experiments with different driver sizes for radices of 8, 16, and 32 for an output wire of 1mm (0.15 ohm/sq, and 100fF/mm) as can be seen in Figure 3.7, 3.8, and 3.9. For the tri-state multiplexer, increasing the driver size decreases the driver resistance, but also increases the parasitic loading. For smaller driver sizes, the latency decreases as the driver size increase since the parasitic loading is much smaller than the wire capacitance. But when the parasitic loading becomes comparable to the wire capacitance, the latency improvement saturates. Further the latency starts increasing for larger radices like 16 and 32 when parasitic loading starts dominating. We repeated the same experiment for radix-16

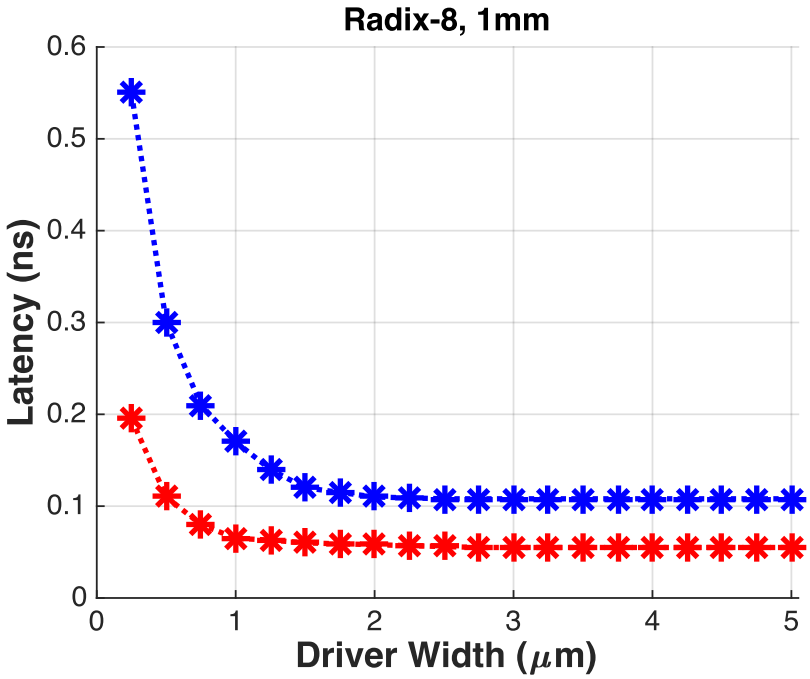


Figure 3.7: Latency comparison for radix-8 tri-state multiplexer and capacitively coupled multiplexer with a 1mm long output wire.

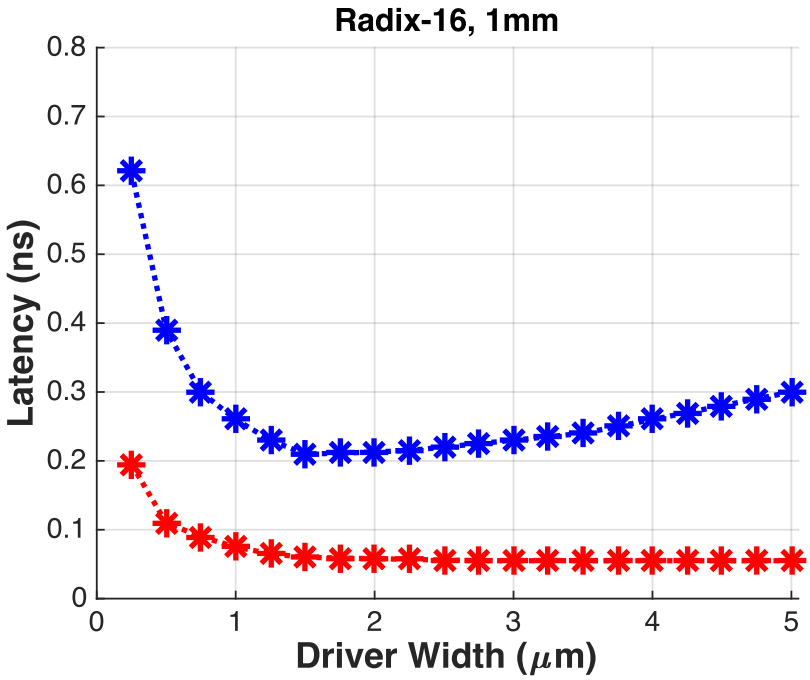


Figure 3.8: Latency comparison for radix-16 tri-state multiplexer and capacitively coupled multiplexer with a 1mm long output wire.

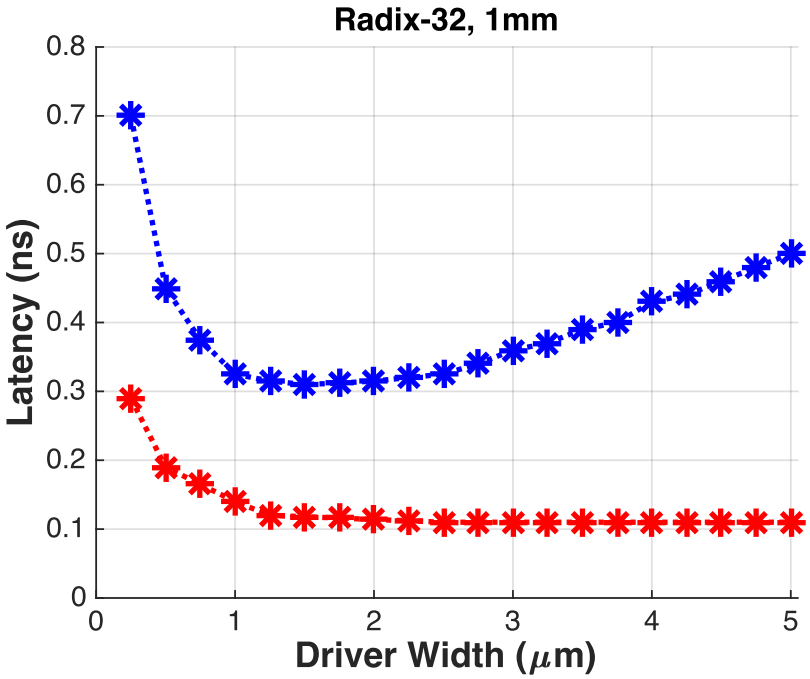


Figure 3.9: Latency comparison for radix-32 tri-state multiplexer and capacitively coupled multiplexer with a 1mm long output wire.

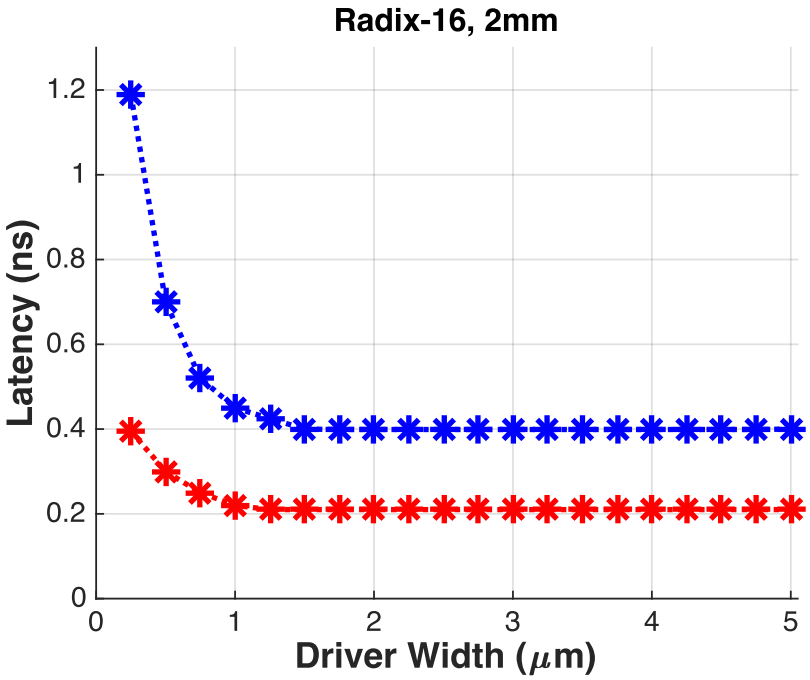


Figure 3.10: Latency comparison for radix-16 tri-state multiplexer and capacitively coupled multiplexer with a 2mm long output wire.

with a 2mm output wire in Figure 3.10. In this case, the wire capacitance is bigger than to the 1mm wire, hence the latency increase at the larger driver sizes is not as significant. On the other hand, the capacitively coupled multiplexer offers ~1.5X better latency than the tri-state multiplexer. In this case, the driver is decoupled from the output wire (increasing driver size does not effect the parasitic loading), and using nonlinear capacitances allow us to use smaller series capacitances to drive the output wires for the desired voltage swing since it eliminates the significant parasitic loading from the non-active inputs.

For our low-swing crossbar scheme, we used the input receivers as the drivers of the nonlinear PMOS capacitances instead of the nand gates as explained below.

3.1.5 Capacitively Coupled Output Multiplexing and Wires

The inputs of the crossbar multiplexers are distributed throughout the entire design, hence the multiplexer is built as a long output wire spanning the design width and the inputs connecting to the output wire via switches. In addition, as mentioned in our low-swing crossbar scheme, the inputs are low-swing, hence receivers are required at each switch point of the multiplexer. In light of these, we used the input receivers (sense amplifiers) as the switch drivers for the PMOS capacitances to build the capacitively coupled multiplexers for our low-swing crossbar design.

A single switch point of the capacitively coupled multiplexer is shown in Fig 3.11. The sense amplifier drives the PMOS capacitances. The differential outputs of the sense amplifier are both precharged to Vdd when the sense enable signal is low. If the corresponding input module is granted priority and wants to talk to the corresponding output module, the sense enable signal transitions high. When the sense amplifier is enabled, the differential low-swing input signal is amplified quickly via positive feedback and one of the differential outputs transitions from high-to-low. The drain-source connection of the PMOS capacitances is also biased at Vdd, therefore if the gate of the PMOS (the output of the sense amplifier) transitions low, the PMOS turns on and drives the output wire capacitively. On the other

hand, if the sense amplifier for that switch point is not enabled, the differential outputs both stay high, hence the driven PMOS stays turned-off and presents a low parasitic capacitance on the output wire.

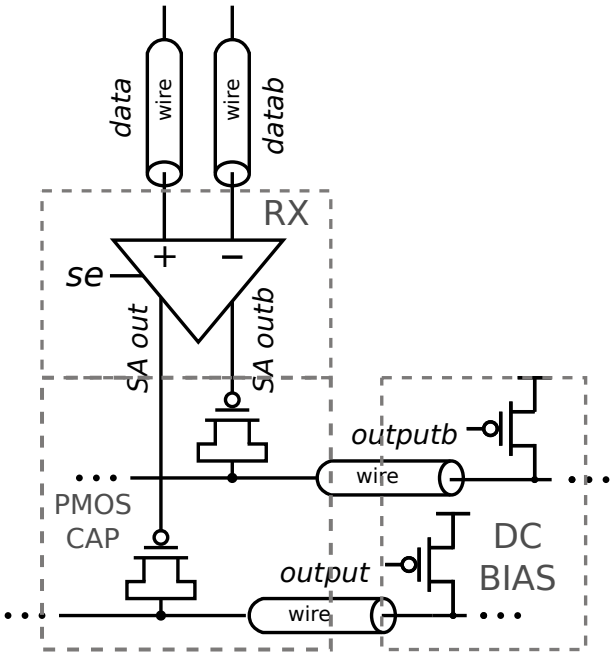


Figure 3.11: A single switch connection of the capacitively coupled multiplexer.

N:1 capacitively coupled multiplexer for our low-swing crossbar is built using N of these switch points in Fig 3.11. Only one of the input modules can be granted priority, hence only one the sense amplifiers will be enabled driving the long output wire via the ON PMOS capacitance, whereas the rest of the parasitic loading will be the OFF PMOS capacitances. The outputs are also routed using twisted and interleaved differential wires and the same precharge scheme is used for the DC bias of the output wires.

Since the outputs are also low-swing wires, receivers to restore the full-swing signals are needed. We used sense-amplifier based flip-flops (SAFF) [38] to mask the full-swing restoration latency and to store the output data. The SAFFs are also designed to have low systematic and random offset to enable lower voltage swings at the output wires.

3.2 16x16 Low-Swing Crossbar Implementation

We designed a 16x16 72b low-swing crossbar in 40nm CMOS bulk process using capacitively coupled input data transmitters and capacitively coupled output multiplexers (Fig. 3.12).

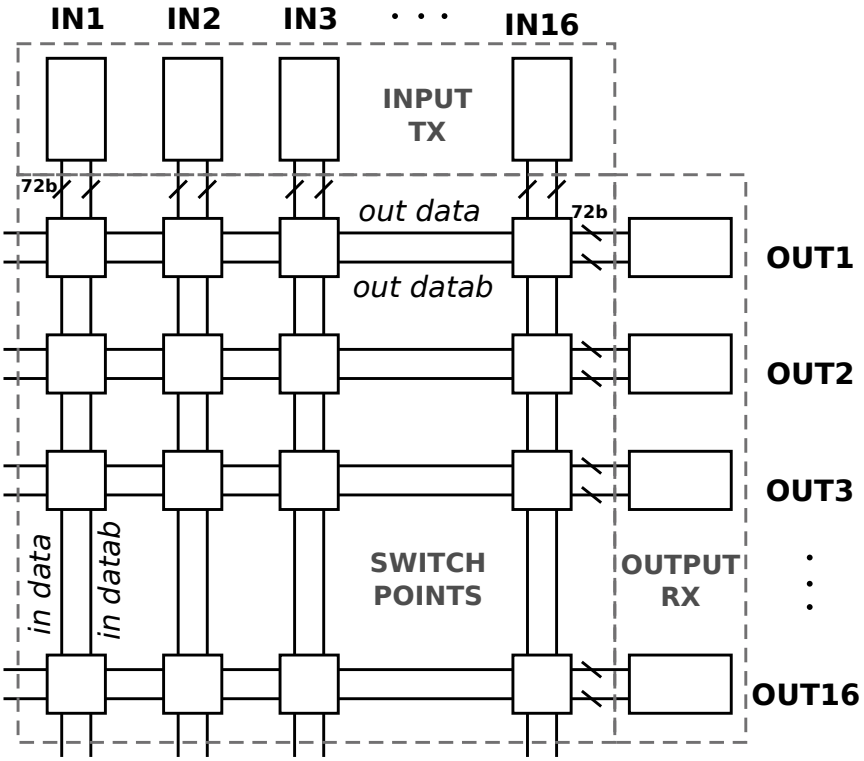


Figure 3.12: Top-level view of the low-swing crossbar design.

Differential input data wires are routed vertically using two metal layers (M3 and M5), hence the input data transceivers are located on top of the design and stacked according to input wire pitches in order to align with the input data wires. Every input module has 72b of input data (64 data bits + 8 ECC bits) that are routed as a group and distributed to every switch point for each output module. The control signals for each input module are also routed vertically with the input data. The control signals are the priority (if the input module is selected to transfer data) and the address (4bits) of the output module that the input module wants to communicate with. At each switch point, the address is decoded

and logically ANDed with the priority bit to generate the sense enable signal. Among N of the vertical switch points, only the one with the correct address will be selected if the input module is granted priority.

The output wires are routed horizontally, also using two metal layers (M3 and M5) and span the entire width of the design. The switch points are placed underneath the I/O wires, and the wire pitches and the number of data bits determine the total area. Every switch point shown in Fig 3.12 has 64 (for each data bit) of a single switch connection of the16:1 capacitively coupled multiplexers and can only use lowest two metal layers (M1 and M2) for routing. The sense enable signal is shared between and routed to these switch connections.

Finally, the output data receivers (sense amplifier based flip flips) are located on the right side of the design to receive the low-swing output data wires. Similar to the input data transmitters, the output receivers are also stacked according to the wire pitches and aligned with the wires.

3.2.1 Design Optimization

The main challenge of the crossbar design is the large design area and hence the long wires. The wire pitches determine the area, and the input wire pitch determines the output wire length and vice versa due to the matrix-style wiring. Therefore, the wire performance cannot be improved by just increasing width to decrease resistance or increasing spacing to decrease the capacitance since increasing the wire pitch will increase the wire length (and hence capacitance and resistance) as well.

Further, the switch points are placed underneath the wires and their area is limited by the wire pitches and the number of data bits. Smaller switch areas are desirable for shorter wires, but this prevents designing low-offset sense amplifiers (that limits the input swings) and limits the size of the PMOS capacitance (that limits the output swings).

Table 3.1: Frequency, Power, Area, and Tbps/W Comparison

Design	Radix	Data Width	Freq (GHz)	Power (W)	Area (mm^2)	Tbps/W
Proposed (40nm)	16x16	72	2.2	0.244	0.94	10.49
Conventional (40nm)	16x16	72	2.5	0.950	0.32	3.03
[50] (45nm)	4x6	128	2	0.467	0.39	2.74
[53] (45nm)	64x64	128	0.559	1.3	4.06	3.4

We modified the crossbar modeling tool to incorporate capacitively coupled wires and multiplexers. The wire pitches (width and spacing) are limited by the layout rules and our modeling results suggested using 0.7μm for input and 0.9μm for the output wire pitch. This is larger than the minimum pitch in order to be able to increase the area for the switch points. The output wire pitch determines the input wire length, and since the input data transmitters can be stacked on top of the design and can be larger, the output wire pitch is larger than the input wire pitch. The sense amplifier and the SAFF are sized to have a 3σ offset of 45mV. Both the input and output PMOS capacitances are sized according to the wire capacitances and the parasitic capacitances (the receivers for the input wires and the off PMOS capacitances for the output wires) in order to generate a 200mV (sensed at 100mV, 1/9th of the nominal Vdd of 0.9V) swing on the wires. The input PMOS transistor (W=1.5μm, L=5μm) has a 53fF of C_{on} and 4 fF of C_{off}, whereas the output PMOS transistor (W=0.7μm, L=10μm) has a 60fF of C_{on} and 3fF of C_{off}. Crosstalk from the neighboring wires and especially complementary signals are minimized using twisted and interleaved I/O wires.

3.2.2 Timing Scheme

The timing diagram of the low-swing crossbar is shown in Fig. 3.13. The input and the output circuitries are enabled by the rising and the falling edge of the clock respectively. Hence, they are both dedicated a half cycle period. The input generation consists of differential input generation and data enable(40ps) (stacking of these blocks resulted in longer interval wires, hence increased the propagation delay), and low-swing input drive (120ps).The output generation consists of the sense enable signal generation and distribution to the

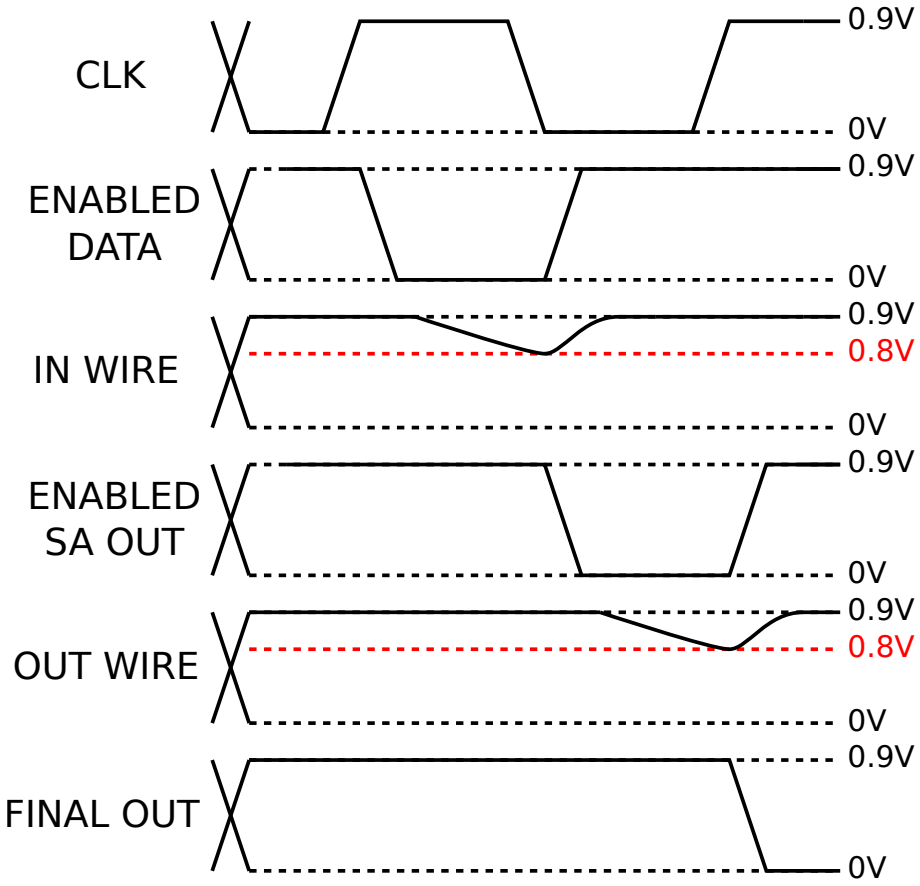


Figure 3.13: Timing diagram for the low-swing crossbar design.

sense amplifiers (40ps), the low-swing input sensing (30ps), the low-swing output drive (120ps) and the 10% clock skew for the SAFF. Since the output portion is the bottleneck, it determines the clock frequency of 450ps.

3.2.3 Evaluation Results and Comparison

Post-layout simulation shows the design operating at a maximum frequency of 2.2GHz, achieving a bandwidth of 2.56Tb/s at 0.9V (nominal Vdd) with an area of 0.94mm^2. Total energy consumption for full, half, and minimum bandwidths are 110pJ, 84pJ, and 64pJ respectively and the power consumption for full bandwidth is 0.244 mW. We also implemented a 16x16 72b conventional crossbar in 40nm CMOS bulk for comparison. The conventional crossbar is designed with full swing I/O wires and 16:1 multiplexers. The conventional crossbar runs at 2.5GHz, consumes 0.95W with an 0.32mm^2 (0.31μm input wire pitch and 0.55μm output wire pitch). As can be seen in Table 3.1, the low-swing crossbar offers 4X improvement in power consumption at a comparable operating frequency (1.13X worse frequency). The main overhead of our scheme is the additional area. The low-swing crossbar area is 3X larger than the conventional structure due to differential wiring and large switch point circuitries, hence prevents reaching the maximum energy savings.

Further, we compared our low-swing crossbar to previously published results as also can be seen in Table 3.1. Although we compare against a very similar technology point (45nm), the radix and the data width of the designs are different. Therefore, we compared throughput per Watt of the designs. Our proposed scheme achieves 10.49 Tbps/W and offers at least 3X efficiency improvement over previously published results and the conventional crossbar.

3.2.4 Radix Scaling

We also evaluate the radix scaling of the low-swing crossbar switches. Figures 3.14 and 3.15 show latency and energy scaling with radix. Although the proposed design offers iso-performance and significant energy savings at medium radices like 16, the design performance degrades significantly at high-radices. The main bottleneck is the area overhead due to differential wiring and sense amplifiers (at the switch points). Differential wiring doubles the number of wires and the sense amplifiers occupy more area than tri-state buffers as multiplexer switches. Further, sense amplifiers increase the minimum wire pitch due to extra wire tracks required for internal connections. At low-to-medium radices, these overheads are less emphasized, since capacitively coupled wires perform better than full-swing wires, and energy savings are significant. However, at high-radices (64 and more) low-swing crossbars suffer significantly from area overhead compared to conventional implementations, and cannot compete in terms of both performance and latency. As discussed in the previous chapter, design choices that lead to smaller areas usually offer better performance for high-radix crossbar switches due to shorter wires, hence better wire latency and switching energy.

3.3 Summary

In this chapter, we present a low-swing crossbar scheme that uses capacitively couples wires and multiplexers to improve the performance and efficiency. We showed a 16x16 low-swing crossbar in an industrial 40nm CMOS bulk process. Post-layout simulation showed it operating at significantly improved efficiency over a comparable conventional design and previous published designs. Although scaling of the low-swing crossbar is not feasible, we contend that capacitively coupled drivers and multiplexers are a promising scheme for low-to-medium crossbar switches. Next chapter will focus on improving the performance of crossbar switches and present a modular crossbar switch with linear performance scaling

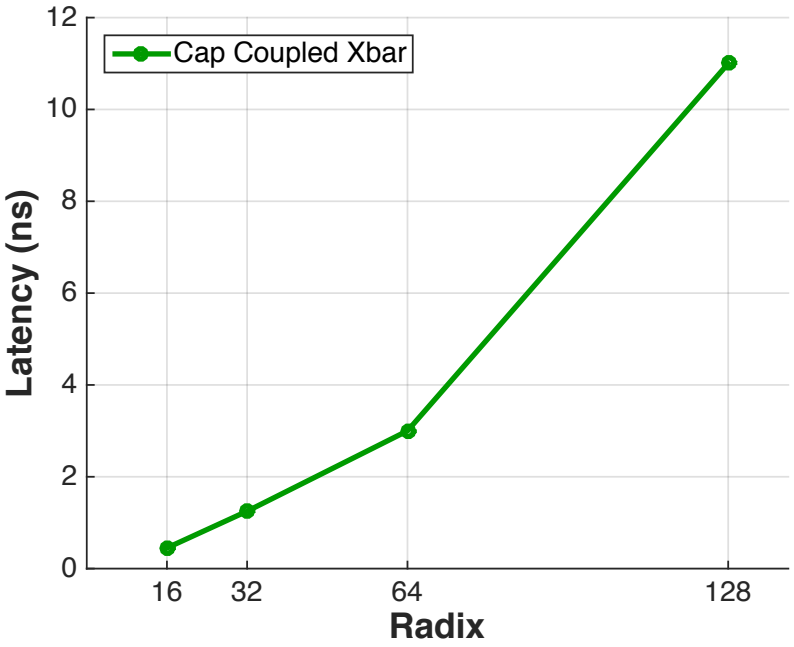


Figure 3.14: Low-Swing Crossbar Latency Scaling with Radix

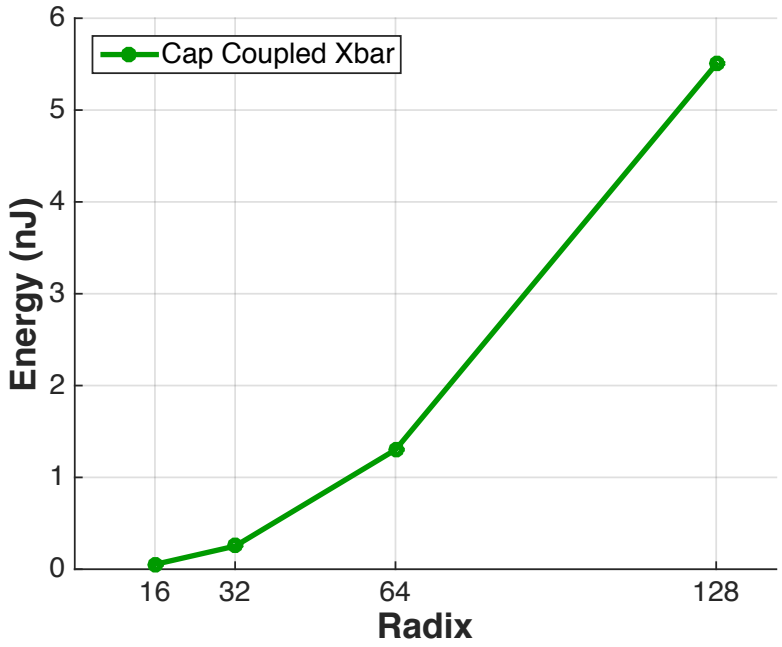


Figure 3.15: Low-Swing Crossbar Energy Scaling with Radix

with radix.

Chapter 4

Modular Crossbar Switches

Using our insights from previous chapters, we focus on how to improve performance and energy-efficiency of high-radix crossbars by optimizating the design parameters and adopting a modular architecture. In this chapter, we present scalable modular crossbar switches that perform better at high radices than the monolithic designs. The blocks are designed to achieve high throughput by limiting the radix, hence area, wire lengths, and parasitic wire parameters. They are arranged in a flow-through scheme to eliminate global connections and maintain linear performance scaling and pipelined to achieve high throughput. Small block sizing and modularity enable deactivating unused I/O wires to improve energy efficiency. The high-performance modular switches offer internal speedup and improve the crossbar network performance as well. Complete modular crossbar architecture will be discussed in Chapter 6.

4.1 Design Optimization and Radix Scaling

Modeling and evaluation results in Chapter 2 provides guidelines to optimize crossbar switch designs for energy-delay product. Optimized designs use flat multiplexers implemented with distributed dynamic tri-state inverters as can be seen in Figure 4.1. Input wire width and

spacing, and output wire width are minimum size, whereas the output spacing is 6X the minimum spacing, and 2 metal layers are dedicated for both the input and the output wires.

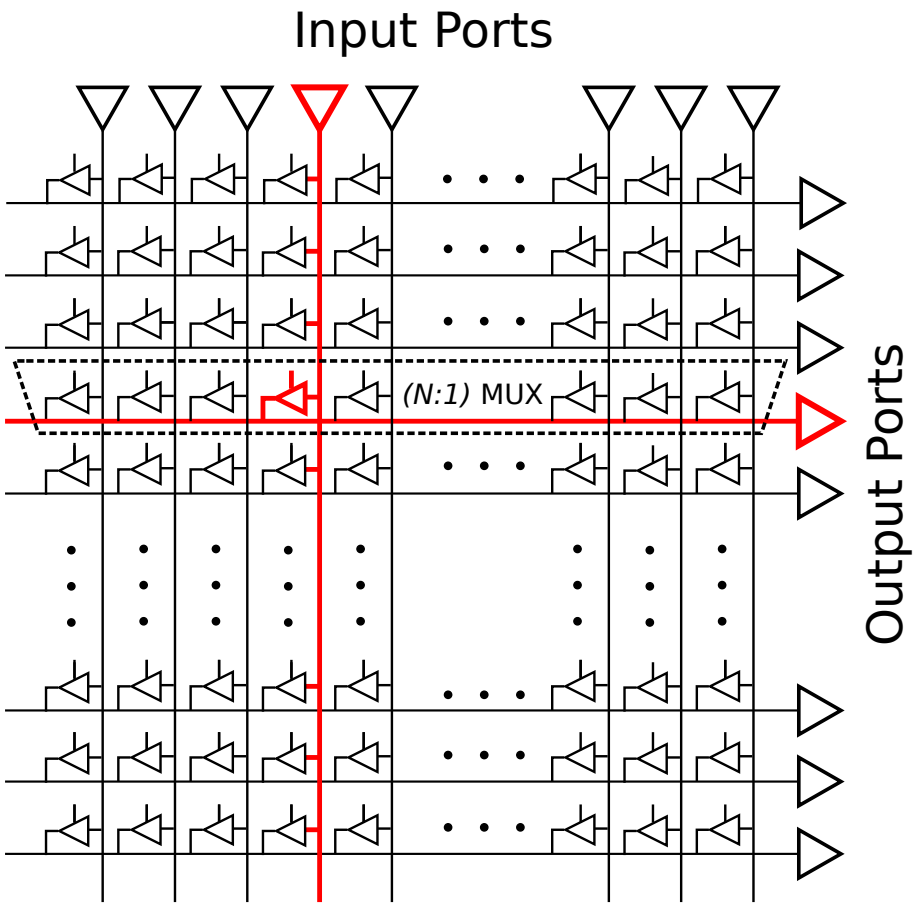


Figure 4.1: Crossbar architecture with dynamic tri-state buffers.

Multiplexers with distributed implementation style have shown to offer minimum energy-delay product, however unidirectional repeaters cannot be used in these designs. The input connections of the multiplexers are distributed along the output wires, therefore the wires can be driven from any of these input connections, hence preventing unidirectional repeater insertion. This presents a concern since output wires can ve very long at high-radices and quadratic latency scaling of wires can dominate the crossbar switch latency.

We compared the optimized design with a crossbar design (using tree-style multiplexers [41]) with repeaters to observe the discrepancies between the latency scaling trends. Figures 4.2

and 4.3 show latency (left) and energy (right) trends with radix scaling using the analytical crossbar modeling tool. Optimized crossbar offers lower latency and energy consumption using circuit and layout level parameter optimizations, however both latency and energy consumption scale quadratically with radix. Conventional crossbar with repeaters achieves linear latency scaling, however it has higher latency and energy consumption than the optimized crossbar.

Although the optimized crossbar offers better performance and energy at high-radices, it can be further improved with linear scaling. We propose modular crossbar switches that use smaller blocks of high-performance switches and arrange them in a flow-through scheme to achieve linear performance scaling. We speculate using smaller blocks to build the high-radix crossbars can overcome quadratic latency scaling of crossbar switches and achieve higher performance and energy-efficiency. We present the implementation details of the modular crossbar switches and performance evaluation using the modeling tool.

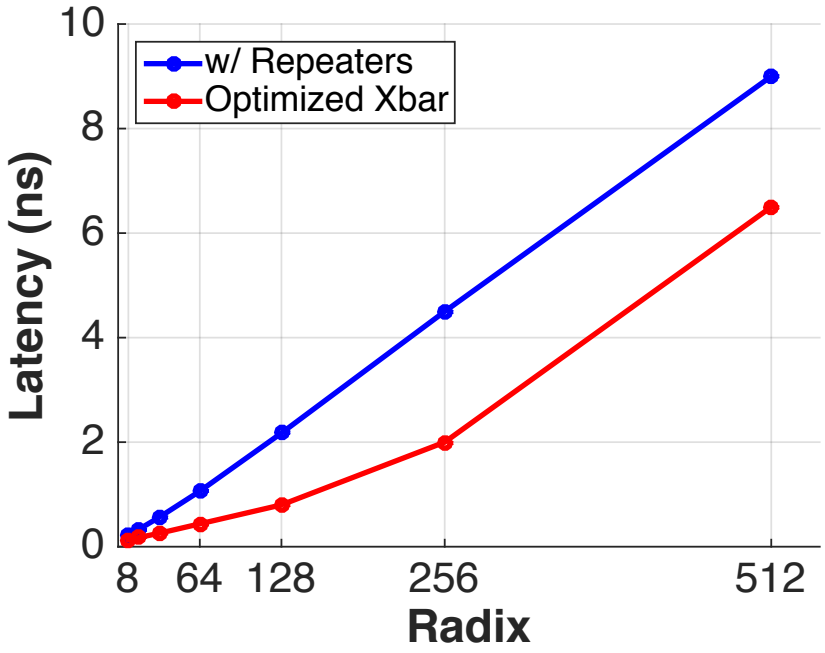


Figure 4.2: Latency scaling trends with radix.

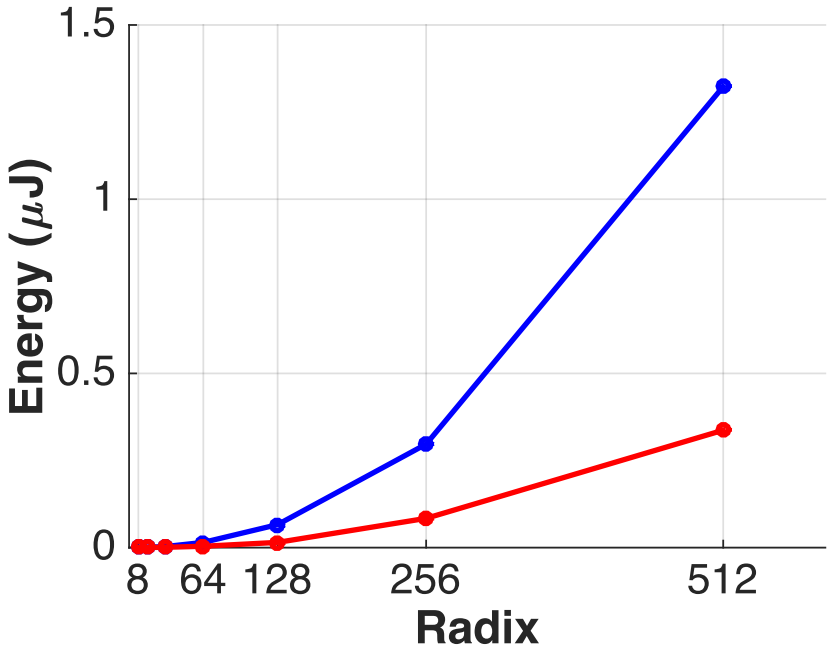


Figure 4.3: Energy scaling trends with radix.

4.2 Modular Switches

Recall that an N x N conventional crossbar switch core is a non-blocking switch that connects N input nodes to N output nodes, where each node has a specific data width (DW). The crossbar switch is shown in Figure 4.4. It consists of N x DW input data bits, N x DW output data bits, and N x N multiplexers, each N:1. As N increases, performance and energy-efficiency degrade significantly due to quadratic area scaling, long I/O wires, and high fan-in and fan-out multiplexers.

The key features of the proposed modular switch are modularity, controlled flow-through operation, and pipelining to overcome scaling challenges. It offers linear performance scaling with radix and energy savings, leading to a better energy-delay product (as much as 5.3X for radix-512) compared to a conventional crossbar switch, with 30% area overhead.

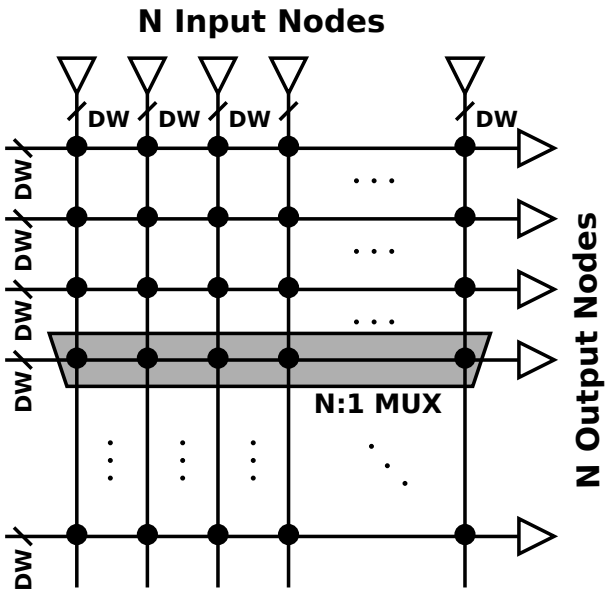


Figure 4.4: N x N **crossbar switch.**

4.2.1 Modularity

An N x N modular switch is built using k^2 blocks, each N/k x N/k. N/k is the radix of the blocks and can be referred as N_{Block} as well. Figure 4.5 shows N x N modular crossbar switches built using 4 blocks of $N/2$ x $N/2$ blocks (left) and 16 blocks of $N/4$ x $N/4$ crossbar blocks (right).

Modular switch blocks consist of crossbar switches and I/O interfaces. Switches handle data traversal from inputs to outputs within every block, whereas I/O interfaces are responsible for the data flow between the blocks. The blocks are connected in a controlled flow-through fashion as explained below to maximize performance and energy-efficiency.

Block-to-Block Interface

In order to avoid global top-level wiring, modular switch blocks are connected in a flow-through fashion. External input ports are located on the top of the switch design and external output ports are located on the right side of the switch design to minimize the area (using matrix-style wiring). Therefore, input data flows vertically (top to bottom) and

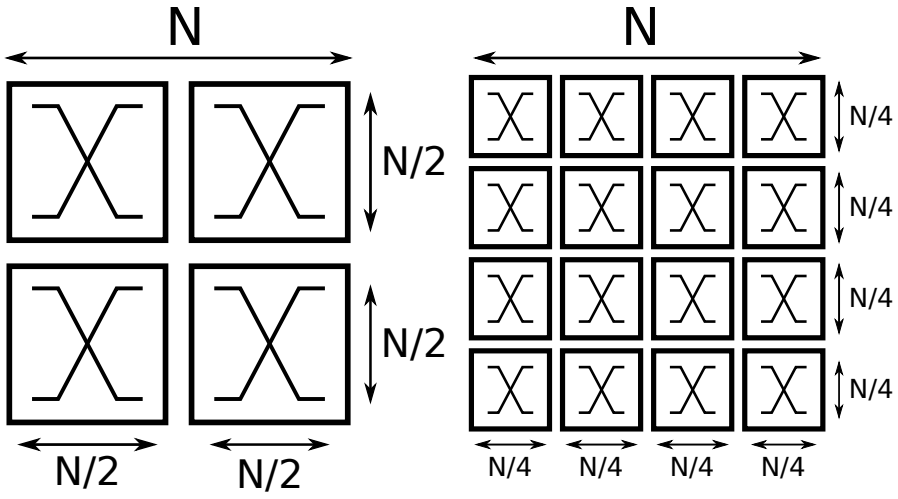


Figure 4.5: N **x** N **modular crossbar switches built with** $N/2$ **x** $N/2$ **(left) and** $N/4$ **x** $N/4$ **(right) blocks.**

output data flows horizontally (left to right) within the modular crossbar switch.

Input interfaces receive the inputs from the top block and distribute the inputs to output multiplexers in that block. In order to achieve all-to-all connectivity, input ports of every block in a column is connected by input interfaces. Further, input interfaces control the data flow using switches to save energy. The data distribution for a given input port is terminated once the corresponding output port is reached.

Output data are generated in a block, and propagate through rest of the blocks in that row to be delivered to the corresponding external output port. In order to enable the output propagation, output interfaces connect their output data as extra inputs to the adjacent blocks on the right (as seen in Figure 4.6). Therefore, for modular crossbar blocks, output data can either be generated or propagated through, and blocks will behave as $(N_{Block}+1)$ x N_{Block} with the addition of external inputs from the adjacent blocks on the left.

Data flow through the modular switches are pipelined to maximize the throughput via latches or flip-flops in the I/O interfaces. Inputs are distributed and outputs are generated or propogated every clock phase or cycle. Since outputs of a block in a column behave as extra inputs to the adjacent block, output generation of that block and input distribution of the adjacent block should be synchronized. In order to achieve that, external input data

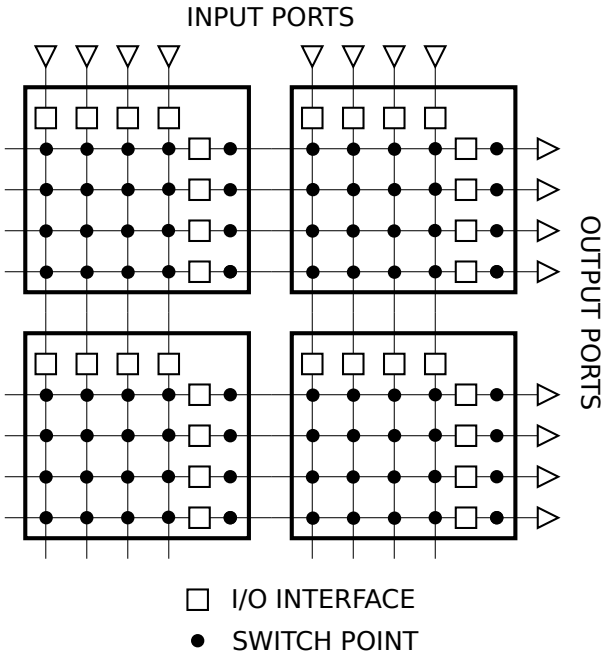


Figure 4.6: An 8 x 8 modular switch built using 4 blocks of 4 x 4 blocks.

is distributed with a phase or cycle shift to the top blocks (from left to right).

Figure 4.6 shows an 8 x 8 modular switch built using 4 blocks of 4 x 4 blocks. Blocks consist of switches and I/O interfaces. Input interfaces pipeline and control data flow from top to bottom. Output interfaces also pipeline the data and connect outputs of a given block via switches to the adjacent (on the right) block's outputs. Switch points of output interfaces are shown separately for easier display of connections.

Example Clocking

We implemented the modular switches using switch blocks with optimized circuit and layout parameters. Previous design space exploration shows that crossbar switches with minimally sized I/O wires and flat multiplexers using dynamic circuits can achieve the best performance and energy-efficiency [9]. Dynamic outputs are precharged while inputs are distributed and evaluated at the next clock phase. Input distribution and output generation have similar latencies, therefore both are dedicated a half clock cycle. Latches are used in

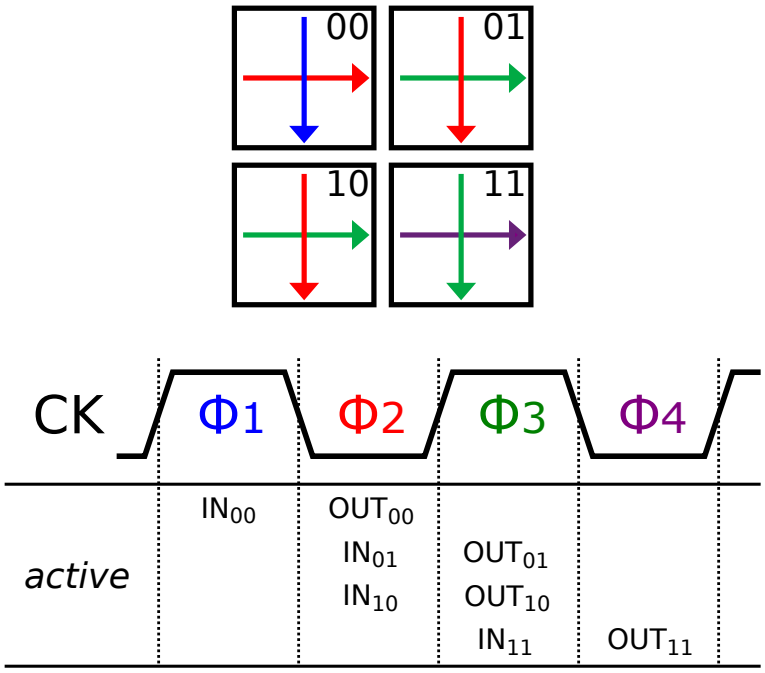


Figure 4.7: Clocking scheme for a modular crossbar with 4 blocks.

the I/O interfaces to connect blocks and pipeline the data flow.

As seen in Figure 4.7, Block 00 inputs are distributed in Φ1, and the outputs are generated in Φ2. Inputs flow from top to bottom and they are distributed to adjacent blocks every clock phase. Since Block 00 inputs are active in Φ1, Block 10 inputs are active in the next phase, Φ2. This minimizes the input distribution latency of the modular crossbars. On the other hand, outputs are generated or propogated every clock phase as well. Since Block 00 outputs are active in Φ2, Block 01 outputs are active in the next phase, Φ3. Block 01 generates outputs from its inputs or propogates Block 00 outputs, therefore Block 01 input distribution is synchronized with Block 00 output generation. Since inputs are distributed every phase, top-to-bottom (worst case) input flow takes $k/2$ cycles. Outputs are generated or propagated every phase as well, therefore left-to-right (worst case) output flow also takes $k/2$ cycles.

4.2.2 Performance Scaling

External input ports are located on top, and the external output ports are located on the right hand side of the design. Therefore, the critical path of an N x N modular crossbar is the data traversal from the leftmost input port to the bottom output port. Input data is distributed from top to bottom blocks and output data propagates from left to right blocks, which takes k clock cycles ($k/2$ for inputs + $k/2$ for outputs). Clock cycle time is set by the crossbar block latency, and total latency for the modular crossbar is calculated as shown below:

$$latency_{total} = k \times latency_{Block} \tag{4.1}$$

As data traversal through the modular crossbar takes k clock cycles, total latency of modular crossbar is linearly proportional to latency of the blocks, where the proportionality constant is $k = \frac{N}{N_{Block}}$. Therefore, latency grows linearly with radix scaling (by increasing k and adding more N_{Block} x N_{Block} blocks) using flow-through modular block scheme.

Moreover, packet sizes are larger than physical switch sizes (flit), hence multiple flits are transferred through the modular crossbar for the same I/O connection configuration. It is pipelined to maximize the throughput (shown in Equation 4.2). However, it takes k cycles to transfer the first flit, thus there is a stall in the pipeline. Nonetheless, the speedup from modular crossbars decreases the average network latency and masks the extra cycles introduced by modularity as will be discussed in Section 6.2.3.

$$TP_{total} = N \times DW \times frequency_{Block} \tag{4.2}$$

4.2.3 Energy Savings

The major contributor to switch energy consumption is switching capacitance of input and output wires. Modular scheme deactivates I/O wires for the unused blocks using switches in I/O interfaces where every input and output port is controlled individually. Input data distribution stops when the block with the destination output port is reached, hence input wires of the lower blocks (than the output) are deactive. On the other hand, output data is generated in a block and propagates through the blocks on the right towards the external output ports. Therefore, output wires of the blocks on the left are deactive.

The worst case energy consumption is when all the ports are active. The energy consumption of a monolithic design is approximately k^2 times the energy consumption of block I/O wires due to input and output wires that span the whole design height and width. However, for the modular design only $\frac{k(k+1)}{2}$ I/O wires are active due to controlled flow of the modular crossbar. This means 25% less energy for a modular crossbar built using 4 (k=2) blocks, 37.5% less energy for a modular crossbar built using 16 (k=4) blocks, and 43.5% less energy for a modular crossbar built using 64 (k=8) blocks. Increased granularity allows deactivating larger portions of the I/O wires, however it also increases the interface area overhead and the number of clock cycles through the switch.

Figure 4.8 shows 8 x 8 modular crossbars built using 4 x 4 (left) and 2 x 2 (right) switch blocks. 0-7 input ports are connected to 7-0 output ports. Active I/O wires are shown with black lines, whereas inactive I/O wires are shown with blue lines. For modular crossbars with 4 x 4 and 2 x 2 blocks, 25% and 37.5% of the switch block I/O wires are inactive.

4.3 Switch Core Evaluation

We evaluated latency, energy consumption, energy-delay product, and area overhead of modular crossbar switches and compared to conventional switches.

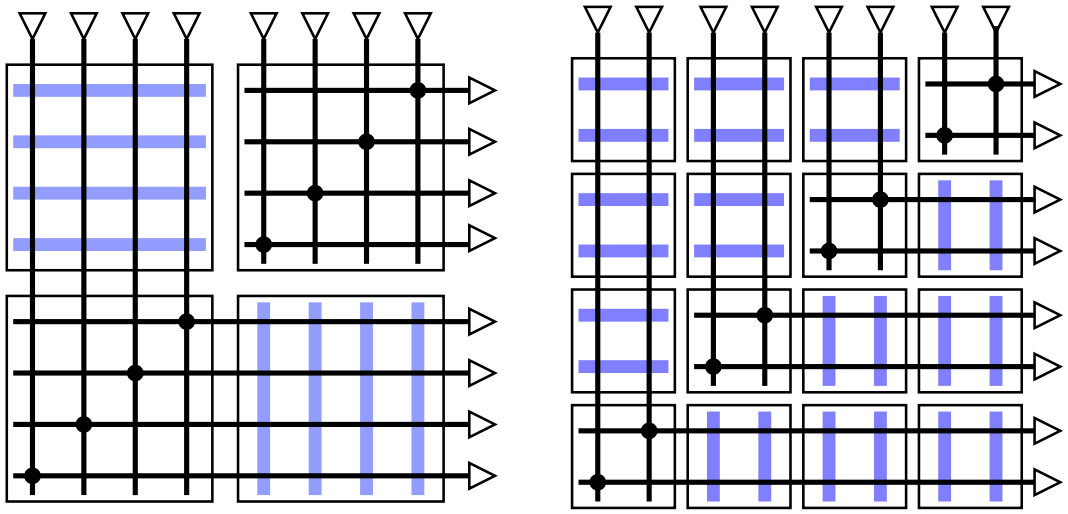


Figure 4.8: Power savings of 8 x 8 modular crossbars built using 4 x 4 (left) and 2 x 2 (right) switch blocks.

4.3.1 Experimental Setup

We used the crossbar modeling tool [9] developed for fast design space exploration of crossbar switches using analytical models calibrated with post-layout extracted simulations. The tool explores a variety of full-custom design options and crossbar design parameters to achieve minimum energy-delay product.

Using the tool, we calculated latency, energy, and area for both conventional and modular crossbar switches with radix scaling in 40nm CMOS bulk process. For every design point, optimized (for energy-delay product) design parameters such as minimum I/O wire pitches, flat multiplexers with dynamic circuits, and a common data width of 64bits are used. We initially chose an optimum block radix and datawidth for the modular design, and used that switch block to build the larger radix modular crossbars. Evaluation and comparison are discussed below.

4.3.2 Optimum Block Radix and Datawidth

Switch block optimization is crucial for maximizing the performance and energy-efficiency of the modular crossbars. The layout and circuit parameter optimization is already discussed in the previous chapters, however system level parameters of block radix and datawidth are yet to be optimized.

Block Radix

Block radix determines the modular crossbar area, frequency, number of cycles for a single-flit transfer, and energy consumption. Area depends on the block radix mainly because of the interface overhead. Choosing a smaller block radix will result in a modular crossbar with more blocks, hence larger area overhead. Energy, on the other hand, also increases by using smaller radix blocks due to extra energy consumption of the interfaces. However, using smaller blocks increases the granularity, and hence allows deactivating larger portions of I/O wires and decreases energy consumption. Finally, using smaller blocks would increase the frequency, however in return it would take more cycles to complete the single-flit data transfer.

Datawidth

Datawidth determines the modular crossbar area, frequency, number of cycles for a packet transfer, and energy consumption. Please note that radix determines the number of cycles for a single-flit transfer while data width determines the number of cycles for a packet transfer. Increasing the data width increases the design area, and energy consumption. Due to larger area, increasing the datawidth decreases the frequency, however it would decrease the number of cycles for the total packet transfer.

Evaluation

In order to incorporate all these trade-offs, we explored evaluation metrics of throughput, single-flit latency, 32Byte (common cache line) packet latency, energy-delay and energy-delay-area product of 32Byte packet transfer. We compared radices of 8, 16, 32, and 64 and datawidths of 64 and 128 bits using the crossbar modeling tool.

Throughput: Figure 4.9 shows throughput for different radices and datawidths. Had switch performance scaled linearly, throughput would have stayed constant with radix. However, as can be observed, throughput initially increases with radix when I/O wires are small and multiplexers dominate the latency. After the peak throughputs are reached at radix-16 and radix-32 for datawidths 128 and 64 bits respectively, they start decreasing with radix scaling as I/O wires become significantly long and wire performance degrades. As can be seen, as datawidth increases, the radix that offers peak throughput decreases since the area and wire lengths are already significant for large datawidths. As can be seen, highest throughputs can be achieved using radix-16, 128 bits and radix-32, 64 bits blocks, therefore these are the promising candidates to build the high-radix modular crossbars. For further evaluation, we explore radix-64 modular crossbar performance for a single-flit and 32 Byte packets.

Performance of a Modular Radix-64 Crossbar: We explore rest of the evaluation metrics for a radix-64 modular crossbar built with different block radices and datawidths. Single-flit latency evaluation as seen in Figure 4.10 is complementary with throughput results. Minimum latencies are achieved by the blocks with the peak throughputs. Packet latency results in Figure 4.11 also follow the same trends, where minimum packet latency can be achieved using radix-16, 128 bits and radix-32, 64 bits blocks. In this case, it is also seen that radix-16, 128 bits blocks have the smallest packet latency. However, larger datawidth implies more area and energy consumption. Therefore, we explored both energy-delay (Figure 4.12) and energy-delay-area (Figure 4.13) products of a packet transfer to find the optimum design point. As can be seen, the minimum energy-delay-area product for a packet transfer can be achieved with radix-32, 64 bits switches. Therefore, we used radix-32, 64

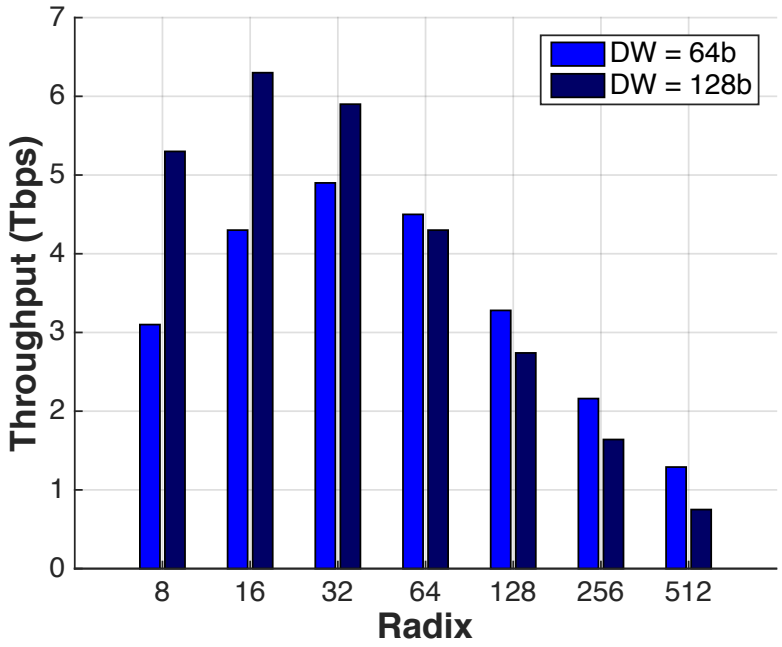


Figure 4.9: Throughput scaling with radix for datawidths of 64 and 128 bits.

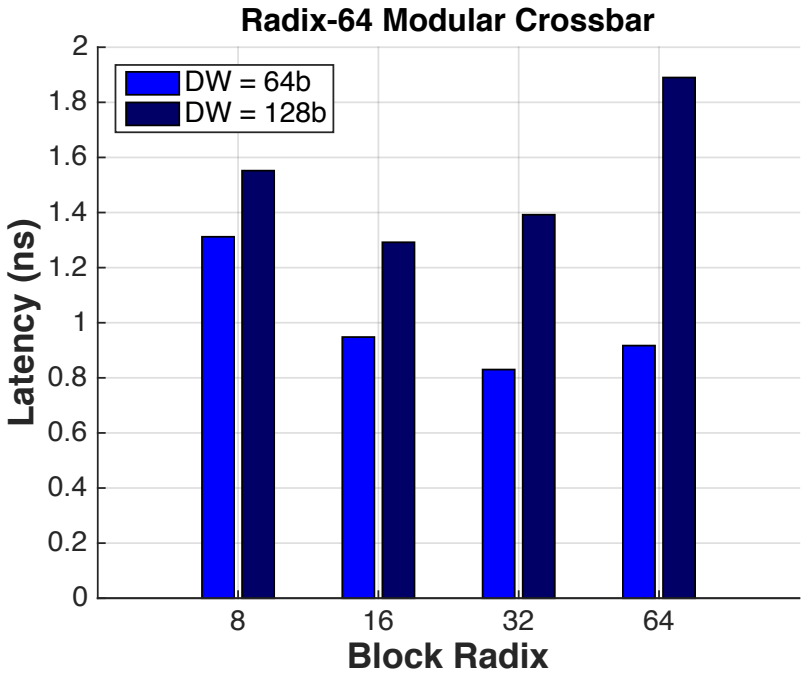


Figure 4.10: Single-flit latency for a radix-64 modular crossbar using different block radices and datawidths.

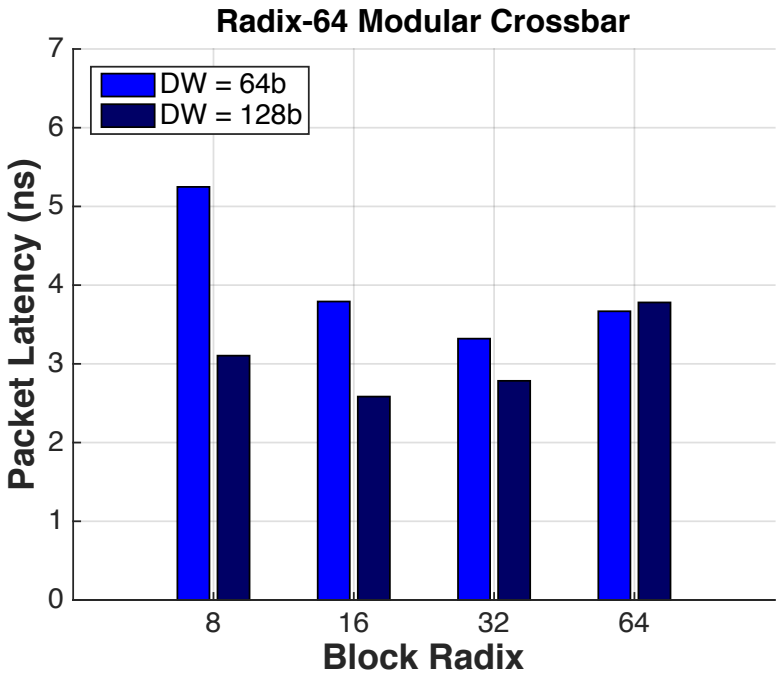


Figure 4.11: 32 Byte packet latency for a radix-64 modular crossbar using different block radices and datawidths.

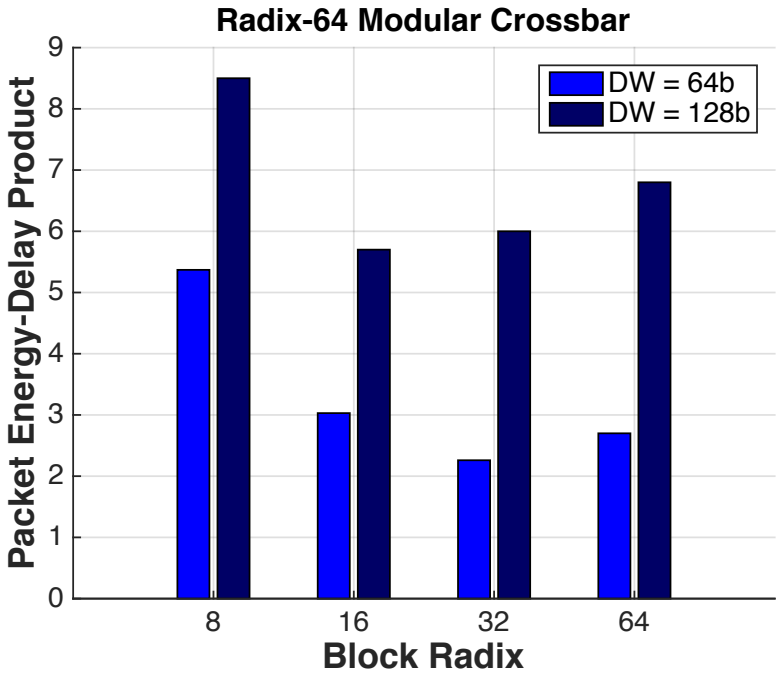


Figure 4.12: 32 Byte packet energy-delay product for a radix-64 modular crossbar using different block radices and datawidths.

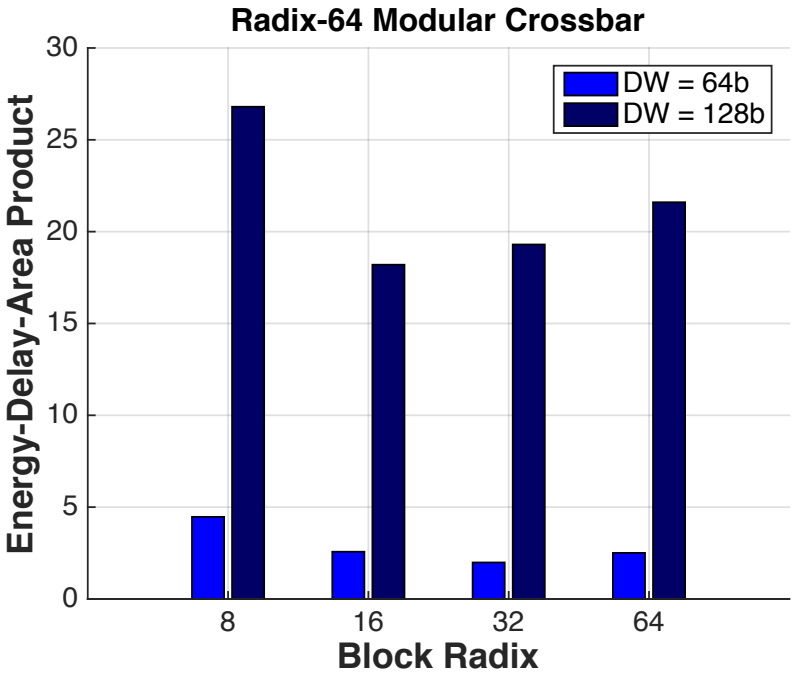


Figure 4.13: 32 Byte packet energy-delay-area product for a radix-64 modular crossbar using different block radices and datawidths.

bits switches as the main building blocks of the modular crossbars.

4.3.3 Energy-Delay Product

Latency of modular switches is evaluated using block latency (including the interfaces) and number of blocks. As can be seen in Figure 4.14, latency of modular switches scales linearly with radix while latency of the conventional switches scales quadratically. This shows that quadratic latency scaling of the long wires within the switch can be eliminated using modular flow-through architecture.

Energy consumption for both conventional and modular switches with radix scaling are shown in Figure 4.15. Modular switches offer less energy consumption than conventional switches due to the controlled data flow of the proposed scheme. Energy savings further increase with radix scaling due to increased modularity and number of blocks.

In order to incorporate both latency improvements and energy savings, we compare energy-

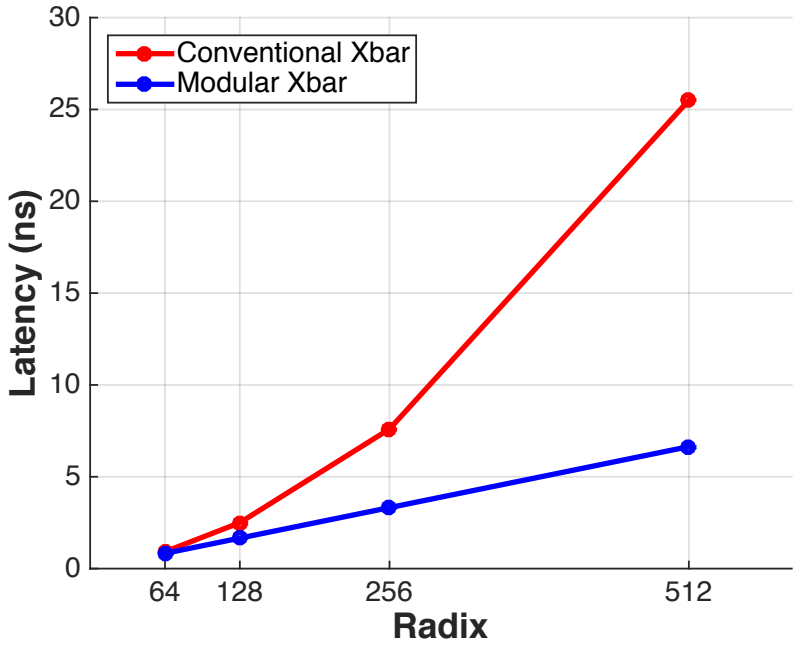


Figure 4.14: Latency comparison for modular and conventional crossbars with radix scaling.

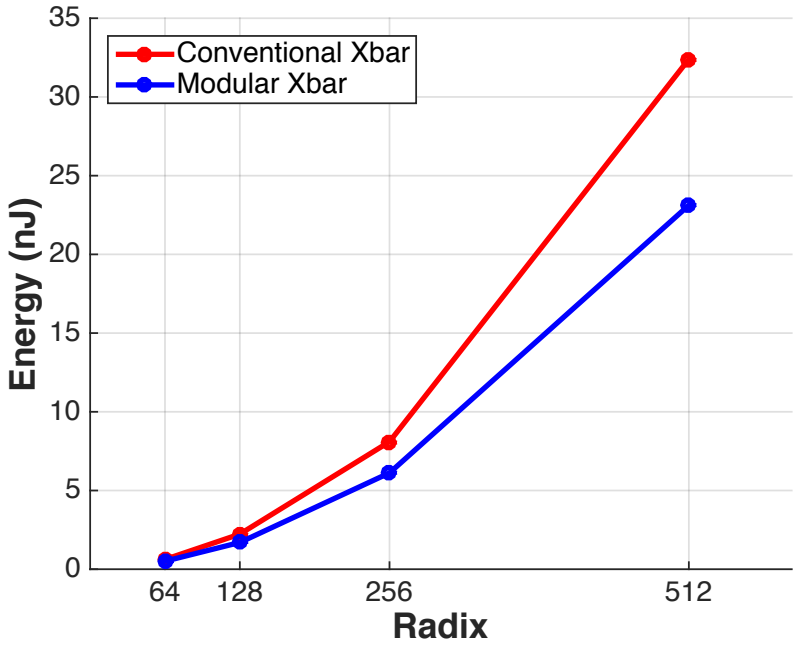


Figure 4.15: Energy comparison for modular and conventional crossbars with radix scaling.

delay product. As can be seen in Figure 4.14, modular switches offer 1.3X, 2X, 3X, and 5.3X better energy-delay product for radices 64, 128, 256, and 512 respectively. As can be seen, benefits of the modular scheme is even more significant at very high radices.

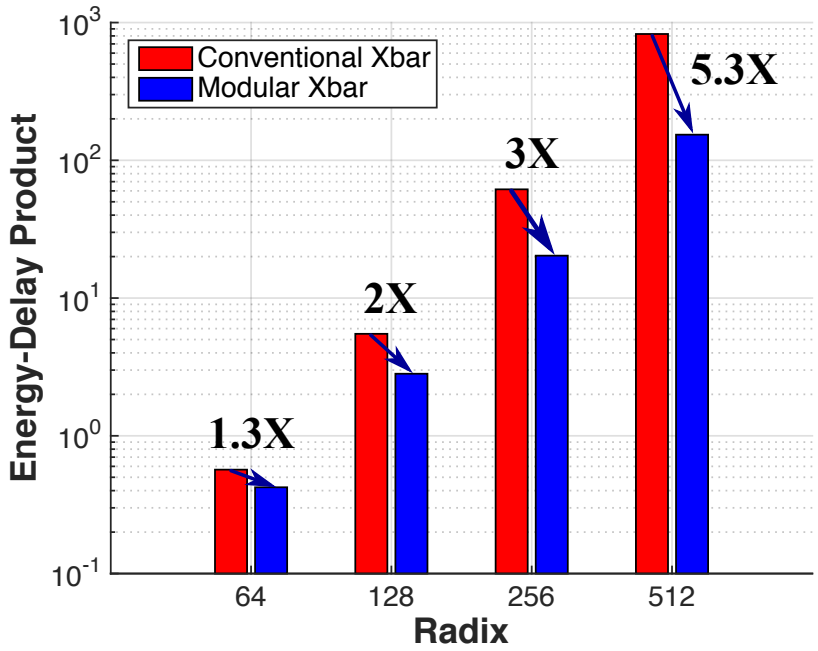


Figure 4.16: Energy-delay product comparisons for modular and conventional crossbars with radix scaling.

4.3.4 Area Overhead

We model and estimate the area overhead of the modular designs due to extra input and output interfaces. Minimum sized latches are used for synchronization and switch circuits are used for controlling the data flow. Layout implementation in 40nm occupies an area of 0.08 mm^2 for I/O interfaces that increases the radix-32 conventional crossbar area by 30%. Area scaling for both conventional and modular switches can be seen in Figure 4.17.

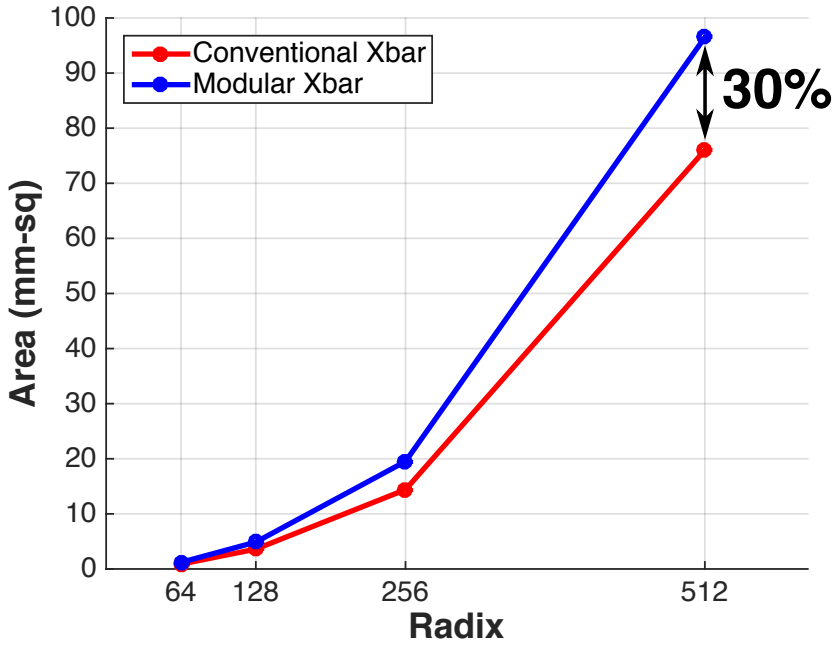


Figure 4.17: Area comparison for modular and conventional crossbars for different radices.

4.4 Summary

High-radix crossbar switches suffer from quadratic area scaling and hence have long I/O wires, and quadratic latency scaling. Modularity divide the long I/O wires into smaller segments, and thus avoids the wire performance degradation. It is similar to using repeaters for long wires, however in this case using smaller switches also help with decreasing the fan-in and fan-out of the multiplexers in the switches. In this chapter, we present modular crossbar switches that maintain linear performance scaling, offer high throughput, energy efficiency, and negligible area overhead at high-radices. Next chapter will detail the prototype modular testchip that verifies the modeling results and feasability of the proposed design.

Chapter 5

Modular Crossbar Testchip

In the previous chapters, we have presented various techniques to optimize crossbar switch design and a modular switch that offers linear performance scaling and energy savings. Our design goal was to achieve high-performance and energy-efficiency at high-radices. We implemented a prototype crossbar switch testchip as a proof-of-concept for our design. The testchip verifies our modeling results and demonstrates that we can build scalable, high-radix, fast, and energy-efficient crossbar switches.

Table 5.1: Technology and Testchip Features

Crossbar Radix	64
Crossbar Datawidth	64
Crossbar Area	1.6 mm^2
Supply Voltage	1.0 V
Frequency	2.38 GHz
Power Dissipation	1.2 W
Technology	40nm CMOS, 10-metal Cu
FO4	9ps (TTLH)

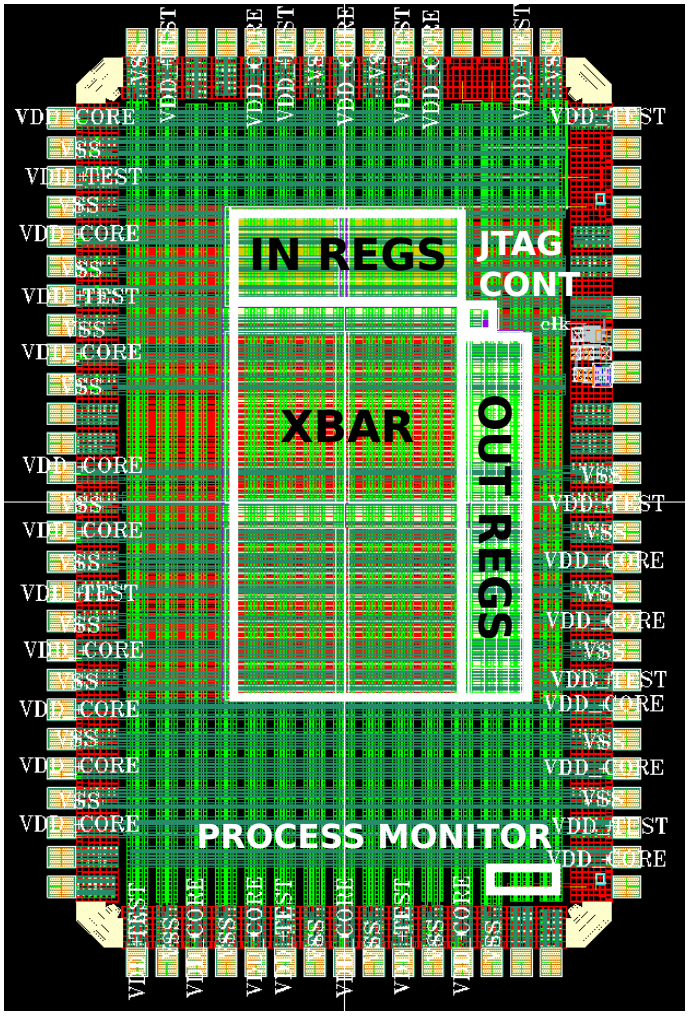


Figure 5.1: Top level layout of test chip.

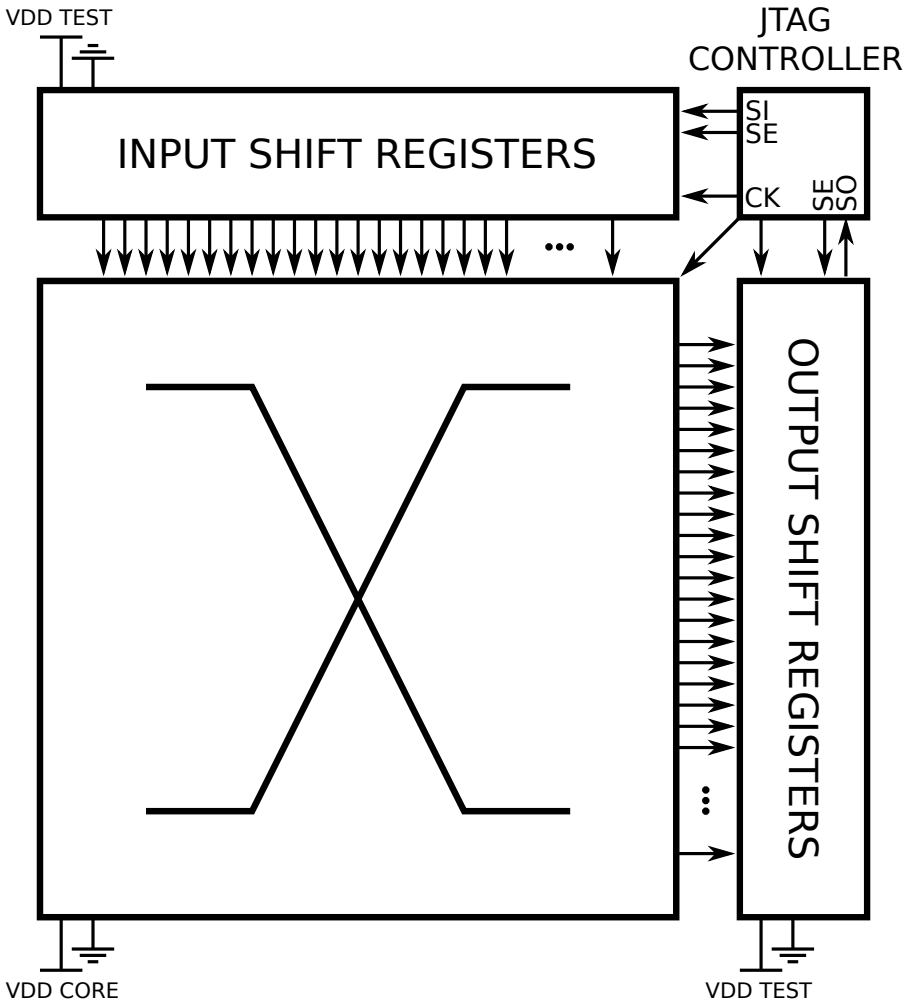


Figure 5.2: Top level block diagram of the test chip.

5.1 Test Chip Overview

We designed the modular crossbar switch testchip in a 40nm CMOS TSMC process with 10-metal layers. Top-level layout of the testchip can be seen in Figure 5.1. The die area is 2 mm by 1.2mm and it has 83 pads along the periphery. Table 5.1 summarizes the technology and testchip features. The testchip consists of the modular crossbar switch (1.6 mm by 1mm), input shift registers, output shift registers, jtag controller, and process monitor as seen in Figure 5.2. The supply voltage of the modular crossbar is seperate from the test infrastructures to enable measurements at different supply voltage levels.

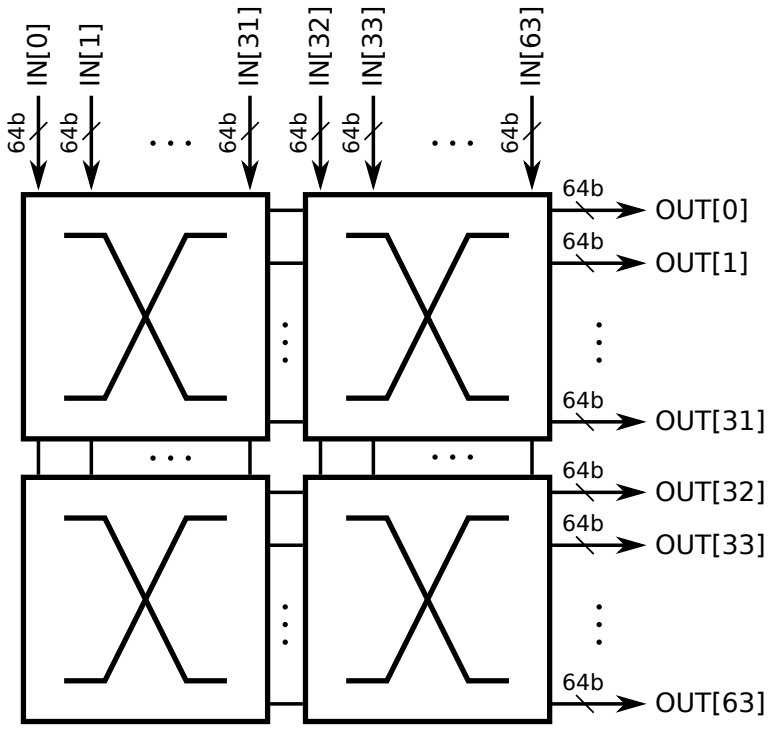


Figure 5.3: Radix-64 64bits modular crossbar switch built using radix-32 blocks.

In order to demonstrate modular crossbar functionality and benefits, we designed a radix-64, 64bits modular crossbar built using radix-32 blocks. As explained in the prevoius chapter, we used the crossbar modeling tool to calculate the frequency, energy consumption, and the area overhead of modular crossbar designs. Modeling results suggest minimum energy-delay-area product for a 32Byte (a common cache line size) packet can be achieved using radix-32, 64bits blocks as seen in Figure 5.3.

As discussed, one of the key aspects of improving crossbar switch performance is to minimize the area. It is determined by the I/O wires and the multiplexers, therefore wire resources are managed carefully considering multiplexer routing, I/O data wires, and top level routing. Bottom 2 metal layers (M1-M2) are dedicated for circuit routing, upper 4 metal layers (M3-M6) are dedicated for I/O data routing, and top 4 metal layers (M7-M10) are dedicated for clock, power, and pad routing.

The modular crossbar runs at 2.38 GHZ at nominal supply voltage (1V) and consumes 1.2

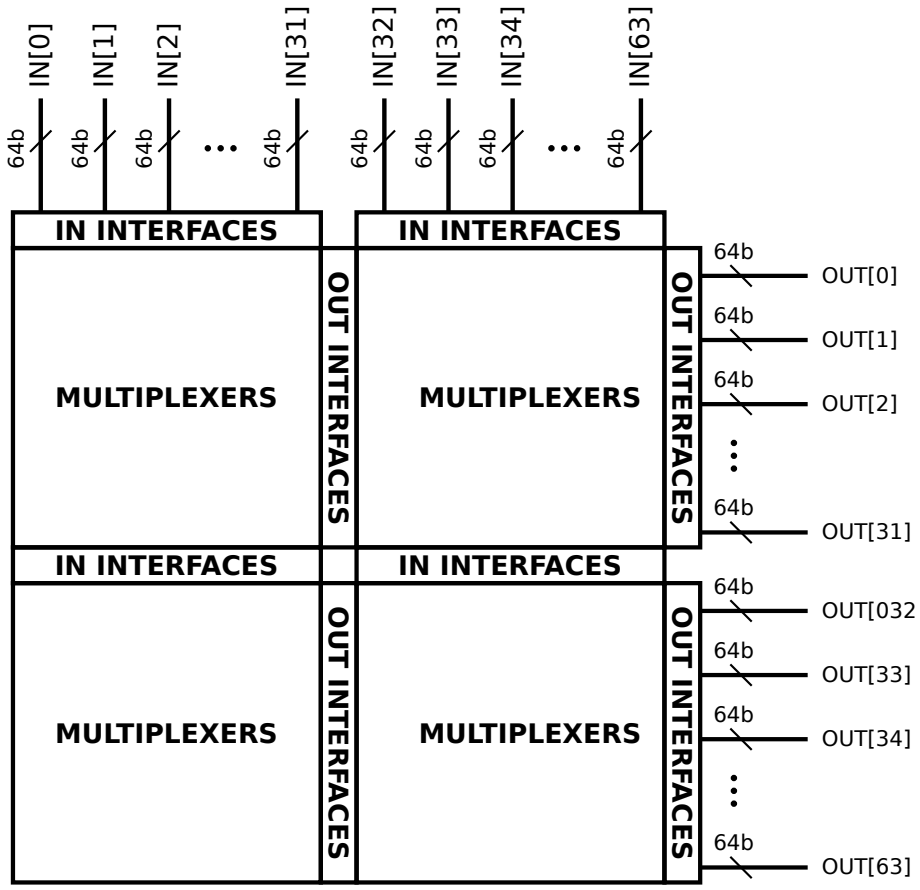


Figure 5.4: Floorplan of the radix-64 modular crossbar switch.

W power. Data transfer from an external input port to an external output port takes 2 cycles through the modular switch. However, the design is pipelined, thus for data packets (with multiple flits), outputs are received every cycle after the initial 2 cycles for the first flit.

Rest of the chapter is dedicated to the floorplan, datapath, control logic, timing, and the evaluation results of the modular crossbar testchip.

5.2 Floorplan

Floorplan of the radix-64 modular crossbar is seen in Figure 5.4. External input ports are located on top of the design and the external output ports are located on the right side

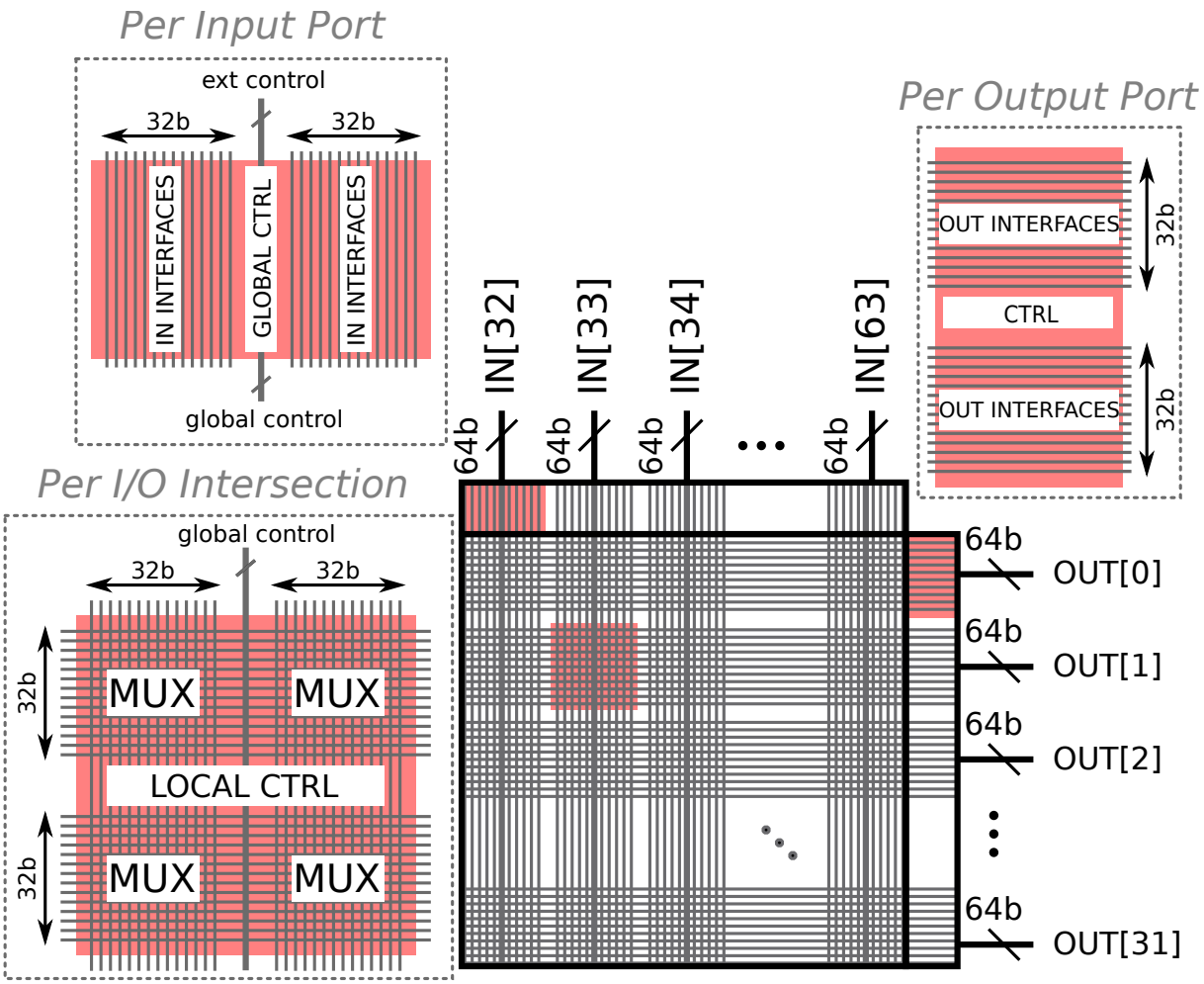


Figure 5.5: Floorplan of the radix-32 switch block.

of the design (from the I/O shift registers). Radix-32 blocks are placed adjacent to each other and connect the I/O data through interfaces. Within every block, input interfaces are located on top of the multiplexers and the output interfaces are located on the right side of the multiplexers.

The detailed floorplan for the radix-32 is shown in Figure 5.5. The I/O data wires are routed on top of the circuits as represented in the figure. Input data and control bits for every input port are grouped and routed together. For every input port, input interfaces for every data bit and global control block is located on top of the block. Global control block receives external control bits such as grant, address bits, and external clock to generate global control bits and local clock.

Every I/O port intersection consists of tri-state buffers of output multiplexers and local control circuits. Every data bit connects to its corresponding output multiplexer in this switch intersection. Local control receives the global control bits and generates local control signals for dynamic tri-state buffers.

Output data bits are also grouped and routed together. Output interfaces for every output port are grouped together for every output data bits. Control circuits to generate local clock are also placed in between output interfaces for every output port as can be seen in Figure 5.5.

5.3 Datapath

The switch blocks are designed and optimized following the guidelines in Chapter 2. Width of input and output wires are 1.4X minimum width to have smaller wire capacitances and a smaller design area. Input wire spacing is minimum distance allowing placement of a contact (3X minimum spacing). However, output wire spacing is larger to allow a minimum size transistor placed with contacts since multiplexers are pitchmatched with corresponding output wires (6X minimum spacing).

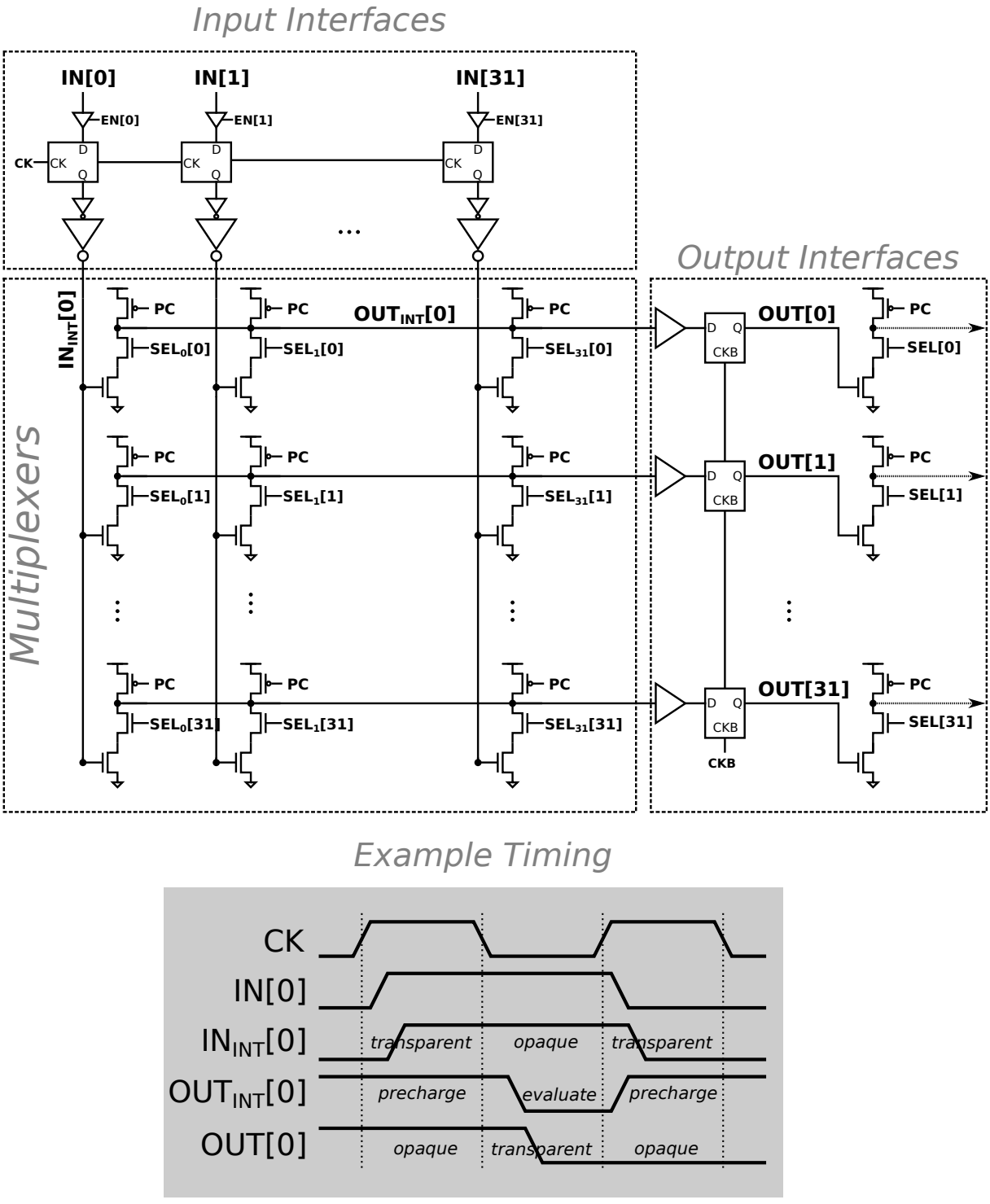


Figure 5.6: Radix-32 1bit crossbar switch block with input interfaces, multiplexers, output interfaces, and an example timing diagram

Top-level circuit diagram for the datapath including the input interfaces, multiplexers, and the output interfaces and an example timing diagram are shown in Figure 5.6. Details for every block is explained below.

Input interfaces

Input interfaces (on top of the design) receive the inputs from the external inputs or the previous block. They control, synchronize and distribute input data as well as the control signals such as grant and output address.

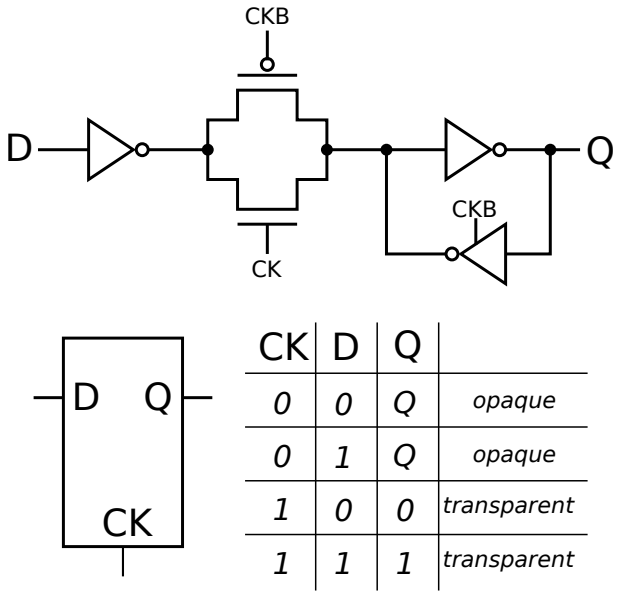
Switches control the input data using an enable signal. The data propogates through the block if the input port is granted permission, and if the chosen output port can be accessed through that block.

Following switches are the static D-latches [68]. Active-high or active-low D-latches (Figure 5.7) are chosen to synhronize the data due to their smaller size (compared to flip-flops). They are transparent on the active clock phases, and opaque on the remaining phases. The data is distributed to output multiplexers on the active phase, and locked on the following phase. The input data flows from external inputs on top to the bottom blocks and adjacent blocks' input interfaces operate on different clock phases to minimize the propogation latency.

Last part of the input interfaces are the drivers. The input wires are long as they span the whole design height, thus inverter chains are used to drive the long wires with reasonable fanout.

Multiplexers

Multiplexers are implemented using dynamic tri-state buffers and every tri-state buffer is pitch-matched with the corresponding input and output wire. For a given output data, the multiplexer fits underneath the output wire pitch and every tri-state buffer is distributed



CK	D	Q	
0	0	Q	opaque
0	1	Q	opaque
1	0	0	transparent
1	1	1	transparent

Figure 5.7: Active-High D-latch.

and centered around its input data. Figure 5.8 shows layout of a single tri-state buffer at the intersection of an input and an output.

In the first clock phase dynamic outputs are precharged (while the input data is distributed) and in the following clock phase, they are evaluated. PMOS transistors precharge the outputs and can be sized small since every tri-state buffer precharges the output whether it is selected or not. They (32 PMOS transistors for radix-32 block) are sized to fully precharge the outputs in half clock cycle.

One of the main advantages of using dynamic tri-state buffers is the small area of these PMOS transistors compared to static tri-state buffers with stacked PMOS transistors. The pull down path consists of two transistors controlled by the select and data signals. Select signal controls the top transistor to prevent charge sharing on the output wires when the input port is not selected.

Further, to decrease the parasitic loading on the output wires, this transistor is sized smaller than the bottom one. The select and precharge signals are generated carefully not to overlap, therefore there is no extra footer device to disconnect the pull-down path from ground during

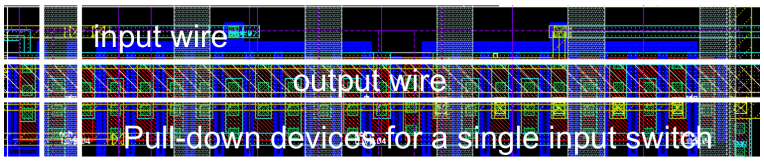


Figure 5.8: Layout of a tri-state buffer connecting a single input and output data.

precharge.

The multiplexer pull-down path is crucial for crossbar performance and sized using the modeling tool. The size of these transistors determine the design area and output wire lengths, drive strength of the multiplexers, and the parasitic loading from the unused inputs, hence there are a lot of trade offs to be considered. Modeling tool enabled us to explore different sizes and to find the optimum size that gave the minimum energy-delay product.

Output Interfaces

Output interfaces receive and synchronize the multiplexer outputs and connect them to the adjacent block (Figure 5.6). The outputs are received by buffers and synchronized using D-latches. The active phase of the latches are opposite of the input interfaces. If the inputs are active on the high-phase, the outputs are active on the low-phase of the clock since half clock cycles are dedicated for input distribution and output generation. The output latches capture the dynamic outputs on the transparent phase when the outputs are evaluated and lock the output data on the opaque phase before the dynamic outputs are precharged. Therefore, outputs are available for a full clock cycle and can be processed by the adjacent blocks.

Following the latches are the extra tri-state buffers to connect the outputs to the next blocks. These tri-state buffers introduce the outputs as extra inputs to the adjacent block. Outputs propogate through blocks towards the external output ports on the right using this scheme.

It should be noted that the output interfaces of the rightmost blocks do not have extra

tri-state buffers since they connect directly to external output ports.

5.4 Control

External control signals of the crossbar switch are grant and output addresses per input port, and the external clock. External clock controls the latches in the input and output interfaces and the dynamic multiplexer operation. Local true and inverted clocks are generated with same skew for active-high and active-low latches, and for control of dynamic multiplexers that operate on different clock phases (for different blocks).

Global control block generates enable and predecoded address signals for every input port, and local control blocks at every I/O port intersection generate the final output address and control the dynamic multiplexer operation. Detailed circuits are explained below.

Global Control

Global control block for every input port receives the external input grant and output address signals and generates input enable and predecoded address signals for the block as seen in Figure 5.9. These global signals are distributed with the input data signals and shared among every output multiplexer switch points. As can be seen in Figure 5.5, the data bits for an input port is group as the top and bottom bits and the global control bits are located in between these groups. The global control circuitry is placed in between the input interfaces.

The enable signal is generated by ANDing the input grant signal and the most significant bit (MSB) of the address bits. MSB of the address bits determines whether the top or the bottom block is selected (top block with outputs 0-31 and bottom block with outputs 32-63). Hence, MSB and the grant signal are used to generate the enable signal specific to blocks. The enable signal spans the whole block height as it is distributed to every

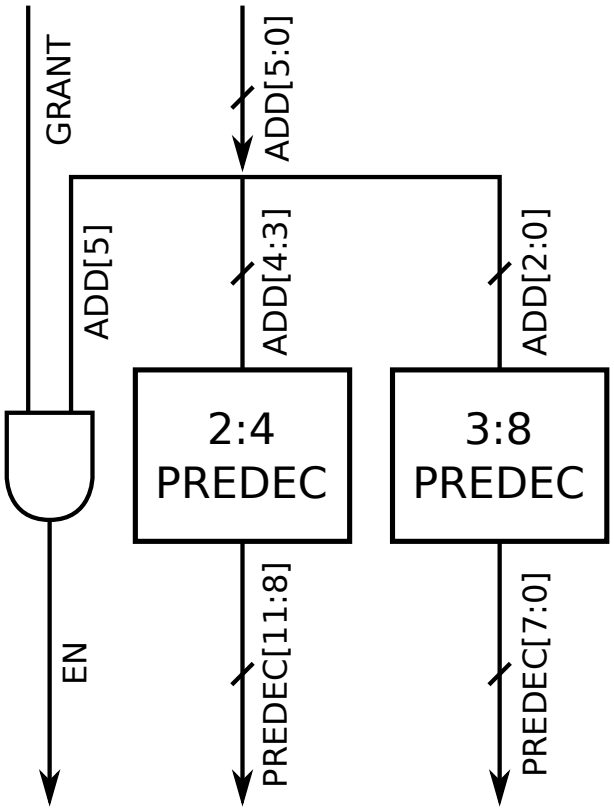


Figure 5.9: Global control circuits per input port.

output multiplexer switch point. Enable generation by ANDing of grant and MSB signals is implemented using a NAND gate and inverters to drive the long wire effectively.

Output address bits for every input port is predecoded in the global control block. Predecoding the address bits prevents redundancy of using local decoders at every output switch point. Instead, only an AND gate is sufficient for every output switch point to generate the final output address (similar to row decoders in SRAMs). For 5 adress bits (radix-32 blocks), 2:4 and 3:8 predecoders are used to generate 12 predecoded signals in total. Predecoded signals span the whole block height as well and the predecoding logic is implemented considering the long output wires.

Local Control

Every I/O port intersection has a local control block to control the multiplexers as seen in Figure 5.10. 64 input data bits are grouped together, therefore at every I/O intersection there are 64 tri-state buffers for every input bit connected to its corresponding output multiplexer for that output port. If the input port is granted access, and wants to communicate with the corresponding output port (for that intersection), tri-state buffers are selected and drives the outputs of the multiplexers.

The select signal is shared among 64 tri-state buffers, therefore is is generated locally and distributed to every tri-state buffer. Initally, final address is generated using the predecoded signals to determine whether the corresponding output address is matching with the requested output address. Select signal depends on the final address as well as the enable signal for that block and the clock phase. As mentioned, dynamic output wires are controlled using clock and is synchronized with I/O interfaces. Therefore, final select signal is also synchronized by ANDing clock (or inverted clock) with the decoded address and enable. It is shared among 64 tri-state buffers, therefore there is reasonable parasitic gate and wire loading. It is driven considering these and timing of the dynamic multiplexers are controlled according to the skew in the select signal.

The precharge signal for the dynamic tri-state buffers are controlled by the true or the inverted clock. External clock is distributed to every local control block, where true and inverted clocks are generated. The precharge signal do not overlap with the select signal, therefore it is active on the opposite clock phase. For example, in Figure 5.10, the select signal is controlled by inverted clock, hence active on the low-phase of the clock. The precharge is generated by the inverted clock as well, hence active on the high-phase of the clock (since it controls PMOS transistors). The precharge signal is also shared among 64 tri-state buffers. The precharge and select signals are both buffered and distributed carefully not to overlap for correct operation.

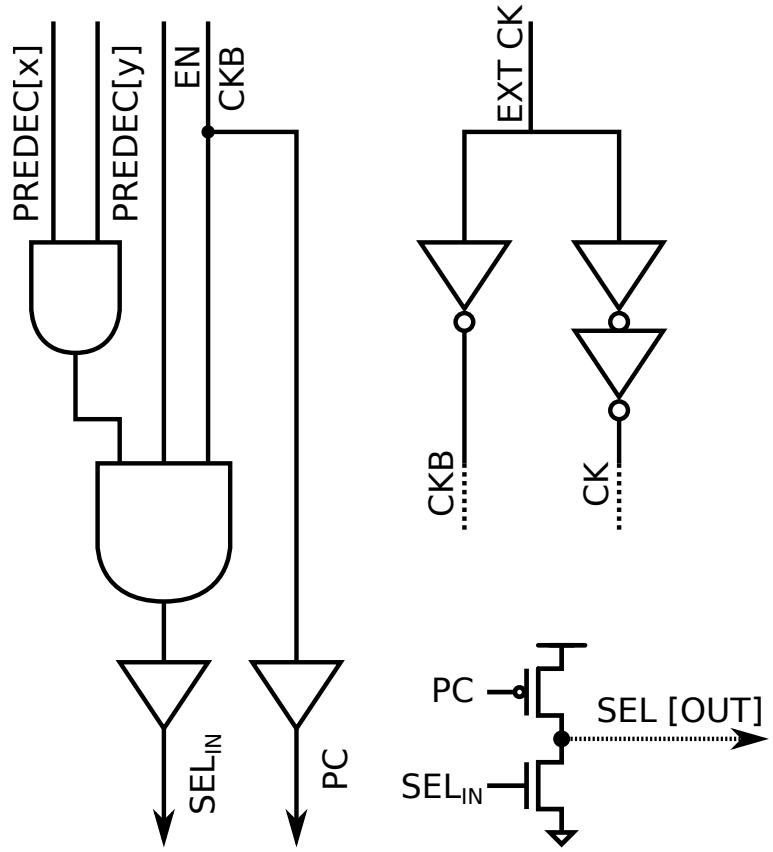


Figure 5.10: Local control circuits per I/O port.

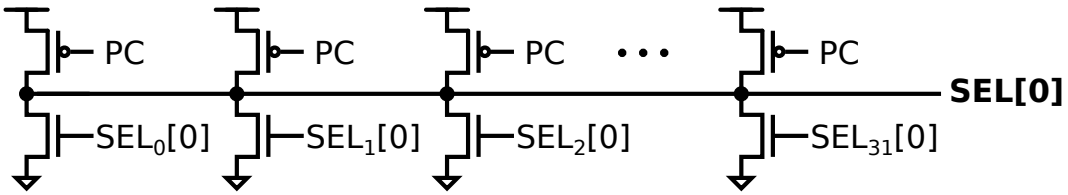


Figure 5.11: Select signal generation for data propagation.

Finally, if the output port is driven by a selected input port within block, it should be selected for the adjacent block as well to be able to propogate the selected data towards external outputs ports. The select signal for the extra tri-state buffer in the output interfaces is active if any of the input ports is selected. This is achieved by implementing a dynamic NOR gate with select inputs from every input port. As can be seen in Figure 5.10, select signal is fed to a dynamic inverter in the local control. The output of this dynamic inverter is shared among 31 (for radix-32) other dynamic inverters from rest of the input port local control blocks. Hence it is implemented as a distributed dynamic NOR gate as can be seen in Figure 5.11. The final select is initally precharged high, and only pulled down if any of the inputs are selected. The distributed implementation prevents dedicating extra wire tracks to route local select signal to the output interfaces.

5.4.1 Example Clocking

An example clocking scheme can be seen Figure 5.12. Input ports 0-63 send all 1s data to output ports 63-0. Input interfaces and output interfaces of a given block operates on different clock phases. Input interfaces of vertically adjacent blocks also operate on the opposite clock phases to minimize input distribution latency.

As seen, Block 00 inputs are distributed in $\Phi 1$, and the outputs are generated in $\Phi 2$. Since Block 00 inputs are active in $\Phi 1$, Block 10 inputs are active in the next phase, $\Phi 2$. On the other hand, outputs are generated or propogated every clock phase as well. Since Block 00 outputs are active in $\Phi 2$, Block 01 outputs are active in the next phase, $\Phi 3$. Block 01 generates outputs from its inputs or propogates Block 00 outputs, therefore Block 01 input

distribution is synchronized with Block 00 output generation.

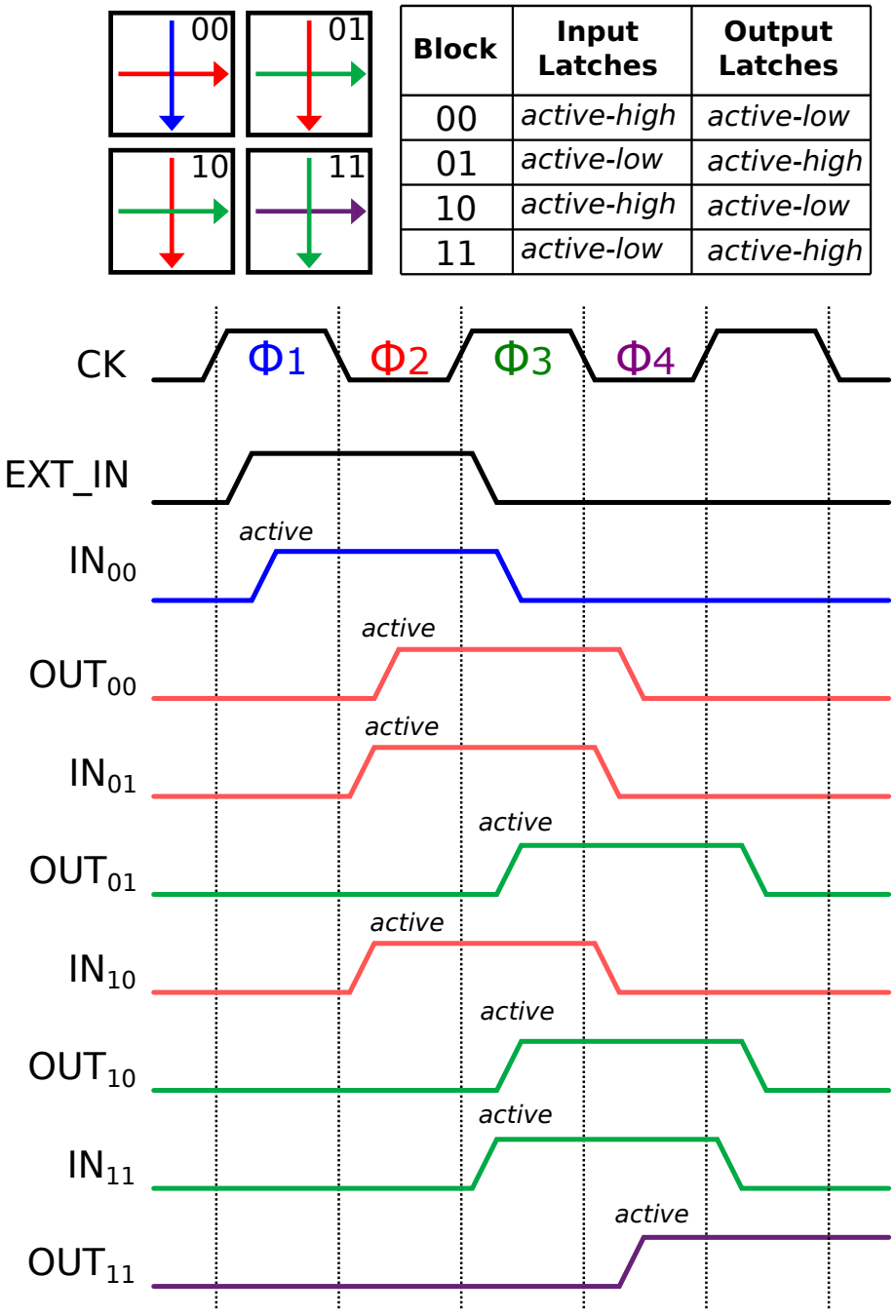
5.5 Test Infrastructure

On-chip testing infrastructures are 4 banks of input and output shift registers and a jtag controller. Every bank consists of 64 (number of ports) times 70 (64 data bits + 6 control bits) registers. Data packets and their corresponding control signals are generated externally and scanned into the input shift register banks for testing. 2 banks of the input shift registers are shown in Figure 5.13. Registers are implemented as scan flip-flops. Scan input and output of the flops are connected in series whereas data input and output of the blocks are connected in parallel (to the next bank). In scan mode, external data is scanned in one bit at a time. In real operation mode scanned in data of a bank is transferred to the next bank. For continuous operation and power measurements, the data outputs of the last input register bank is connected as data inputs of the first input bank.

Output shift registers are implemented similarly to input shift registers, however last outputs are not connected back to the first inputs. They capture the input data packets traversing from the crossbar, and once the operation is completed, the data is scanned out and processed externally. Testing mode, core operation and clocking is controlled by the jtag controller. Off chip testing equipment is set up and extensive functional tests are done using the top level verilog model of the chip.

The testchip contains a process monitor block to measure the fan-out-of-four inverter delay of each die. The block consists of a ring oscillator with 13 inverters. The output of the ring oscilator goes to a by-64 toggle flop frequency divider to achieve a signal with kHz frequency range.

The external clock is received from the jtag. It is distributed to every switch block and I/O shift register blocks in a H-tree structure and distributed in a grid within every block. The clock buffers are used to equalize the skew for local clocks.



Block	Input Latches	Output Latches
00	*active-high*	*active-low*
01	*active-low*	*active-high*
10	*active-high*	*active-low*
11	*active-low*	*active-high*

Figure 5.12: Clocking scheme for a modular crossbar with 4 blocks. Block 00 inputs are distributed in $\Phi 1$, and Block 00 outputs are generated in $\Phi 2$. Block 01 and Block 10 inputs are distributed in $\Phi 2$, and Block 01 and Block 10 outputs are generated or propogated in $\Phi 3$. Block 11 inputs are distributed in $\Phi 3$, and Block 11 outputs are generated or propogated in $\Phi 4$.

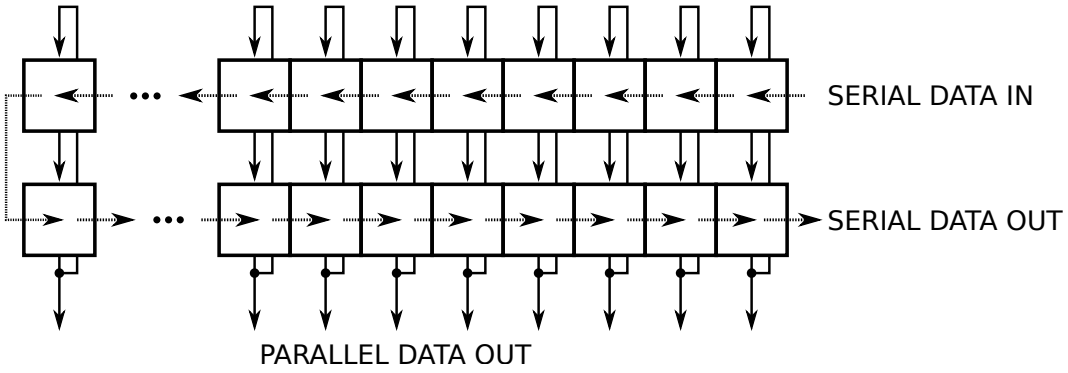


Figure 5.13: 2 banks of shift registers built using scan flip-flops. Test I/O of the flip flops are connected in series for scan mode, and D and Q of the flip-flops are connected in parallel with the next bank for core operation mode.

5.6 Extracted Simulation Results

Due to delays in testchip fabrication, extracted simulation results are reported instead of measurement results. Total chip RC extraction is not feasible for this large of a die size, therefore we extracted critical pieces of the design for the frequency and power measurements. The critical path (connection of the first input port to the last output port) is RC extracted and simulated for frequency measurements. Single I/O port connection for the longest (first input to last output) and shortest (last input to first output) paths are extracted and simulation results are used to interpolate the total switching and idle power consumption.

The modular crossbar runs at 2.38 GHz at 1V nominal supply voltage at room temperature. Frequency results for the modular crossbar for supply ranging from 600mV to 1V are shown in Figure 5.14. Peak throughput is 9.75 Tb/s at 1V. Power consumption at 1V is 1.2 W with 30 % switching activity. This translates to 8.2 Tb/s/W energy-efficiency. Figure 5.15 shows power consumption and energy-efficiency scaling with frequency. As can be seen, energy efficiency can be as high as 38 Tb/s/W with 0.6V operating at 580MHz.

In order to find optimum operating conditions, we evaluated energy-delay product. Figure 5.16 illustrates energy-delay product scaling with voltage. As can be seen, optimal range for energy-delay product is 0.8V-1.0V and below 0.8V energy-delay product increases.

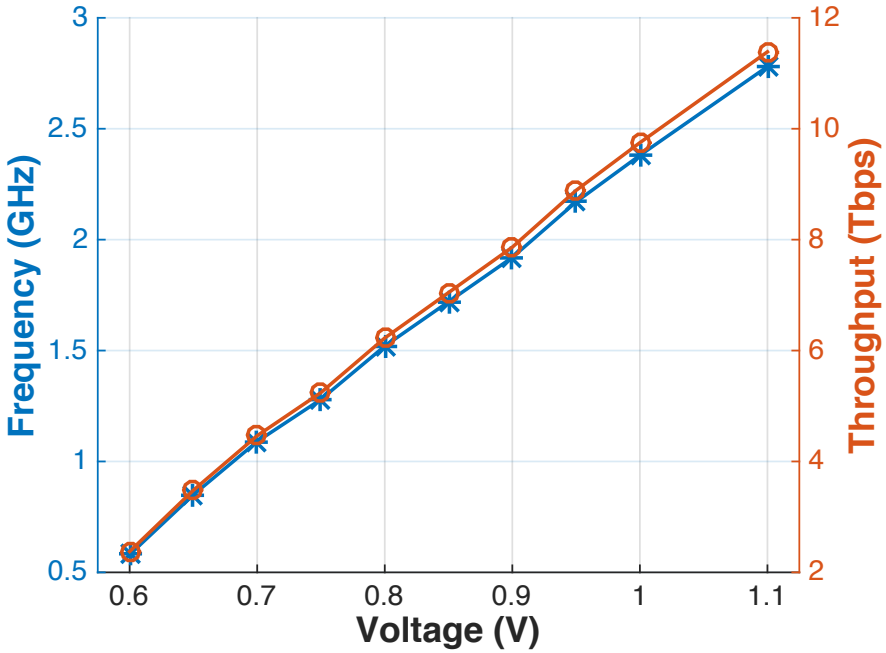


Figure 5.14: Frequency and throughput scaling with supply voltage.

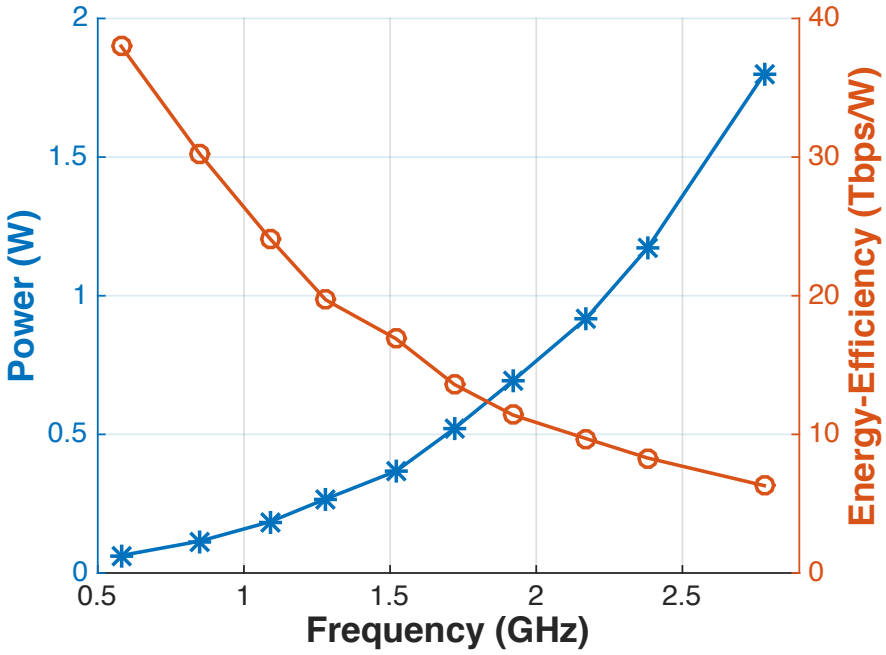


Figure 5.15: Power and energy-efficiency scaling with frequency.

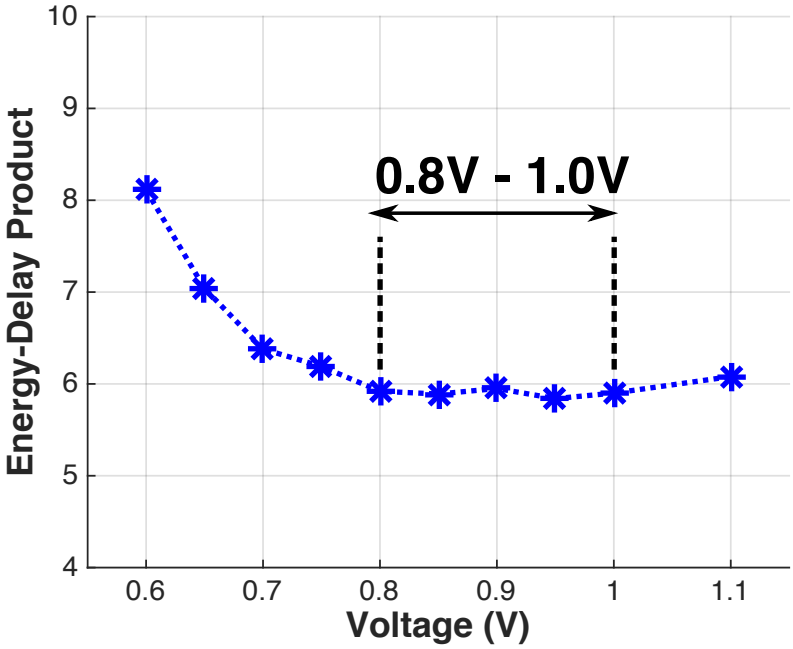


Figure 5.16: Energy-delay product scaling with voltage.

The worst case crosstalk is measured as 100mV (10% of supply voltage) when the victim wire is precharged high and floating, and adjacent wires are pulled-down to ground. The leakege was also another concern for the floating dynamic output wires when the selected input data is 0 (disconnected pull-down path). Leakage sources are OFF PMOS precharge transistors, and the OFF pull-down path of series NMOS transistors from unselected tri-state buffers. Simulation results at 0C, 27C, and 80C can be seen in Figure 5.17. 100mV droop is captured at 450ns, 410ns, and 390ns for 0C, 27C, and 80C respectively. As can be seen for normal operating frequencies on the order of GHz, leakage is not a concern. Increase in temperature helps keeping the precharged dynamic outputs high by increasing the leakage of the OFF PMOS transistors.

Table 5.2 compares frequency, throughput, power, and energy efficiency of the testchip with previously published results [41, 53] in 45nm CMOS bulk process. Since crossbar radix and datawidth are different, frequency and power comparison would not be fair. However, throughput and enegy-efficiency can be compared fairly. As can be seen, modular crossbar

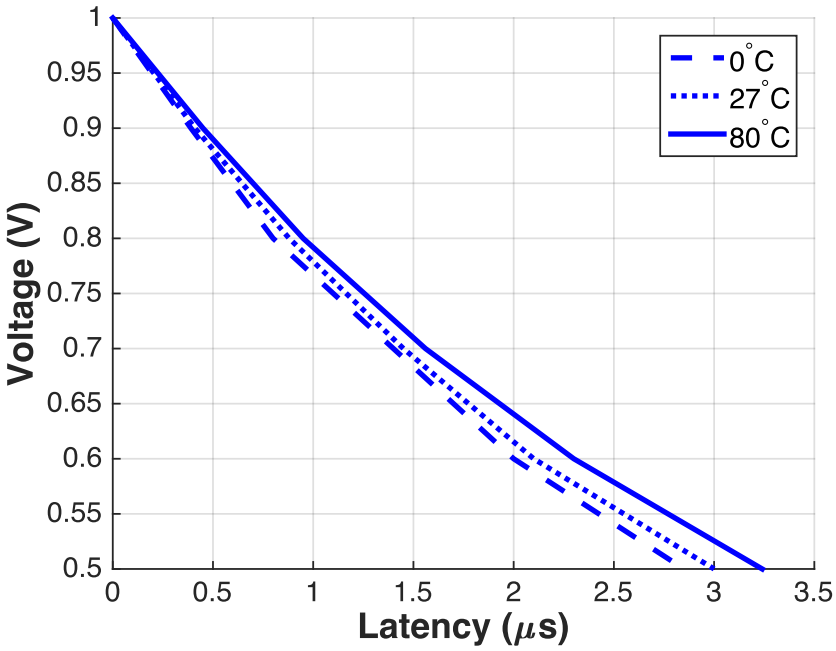


Figure 5.17: Voltage droop of floating output nodes.

Table 5.2: Frequency, Throughput (TP), Power, and Energy-Delay Product Comparison to Previously Published Results

Design	N	DW	Freq (GHz)	TP (Tbps)	Power (W)	Energy-Efficiency (Tbps/W)
Proposed (40nm)	64	64	2.38	9.75	1	9.7
[53] (45nm)	64	128	0.559	4.47	1.3	3.4
[41] (45nm)	128	32	1	4.09	2.3	1.77

offers at least 2.2X better throughput and 2.4X better energy-efficiency than the reported results.

5.7 Summary

This chapter described the prototype modular crossbar testchip and the post-layout extracted simulation results. Radix-64, 64bits modular crossbar is implemented with radix-32 blocks in 40nm CMOS bulk process with 10 metal layers. It operates at 2.38GHz at 1V nominal supply voltage and comsumes 1.2W power. Throughput is 9.75 Tb/ps, 2.2X better than previously published results and the energy-efficiency is 8.2 Tb/ps/W, 2.4X better

than previously published results. Next chapter will present the modular crossbar architecture built using the high-performance modular crossbar switch and network evaluation results.

Chapter 6

Modular Crossbar Architecture

Crossbar switch performance plays a crucial role in the high-radix network performance. Internal switch speedup simpifies the allocation and enables high network saturation capacity with low memory cost. Therefore, high-performance switches can enable building efficient high-radix crossbar networks.

In this chapter, we present a modular crossbar architecture that uses the internal speedup from the modular switches to improve network performance. Complete modular crossbar architecture with the modular switch, centralized allocator, and input and output buffers is shown in Figure 6.1. Internal speedup offered by the modular switch increases the saturation capacity and decreases average network latency with linearly scaling memory cost. Further, high-radix allocation is feasible [44], and the scheduling latency can be amortized over total data packet transfer time through the switch. We evaluated modular crossbar networks with the proposed switch cores using BookSim2, cycle-accurate detailed network on chip tool. The proposed design achieves more than 90% saturation capacity with an internal speed up of 1.5, supports data line rates as high as 102.4Gbps (in 40nm CMOS bulk), and offers lower average network latency compared to conventional crossbars.

6.1 Related Work

Kim *et al.* [26, 27] proposed hierarchical crossbars (HC) built using smaller sub-blocks with combined input and output queuing that lower the allocation complexity and improve switch performance. Mora *et al.* [37] proposed partitioned crossbar input queued (PCIQ) architectures using partitioned crossbar organizations that allow use of simple allocators and crossbars. However, both of the proposed architectures require global connections to distribute and connect I/O data signals to crossbar sub-blocks. For high-radices and large datawidths, global wiring lead to inefficient area and power scaling. The modular switch exploits modularity to improve performance without adding buffers at the intermediate stages like HQ or PCIQ, and flow-through connection of the blocks avoids extra global wiring.

Another approach is to replace crossbar networks with multi-hop networks such as clos networks with high-radix routers. Ahn *et al.* [1] show better scalability of folded-clos switches over hierarchical crossbars while offering equivalent saturation capacity. Similarly, Chrysos *et al.* [12] proposes high-performance multi-stage bufferless scalable clos on-chip (SCOC) architectures that are indistinguishable from hierarchical crossbars in terms of efficiency and fairness. However, compared to conventional crossbars, multi-hop clos networks have significantly larger areas [1], and multi-hop programming complexities. The modular switch area is slightly larger than the conventional crossbar area unlike HQ, and it achieves high-performance while still keeping single-hop network simplicity.

6.2 Crossbar Architecture

Crossbar switch cores are primary building blocks of crossbar networks. High-performance switches offer internal speedup (runs faster than the external data line rates) that increase network saturation capacity and lower average network latency. We use modular crossbar switches to build modular crossbar networks and demonstrate performance improvements

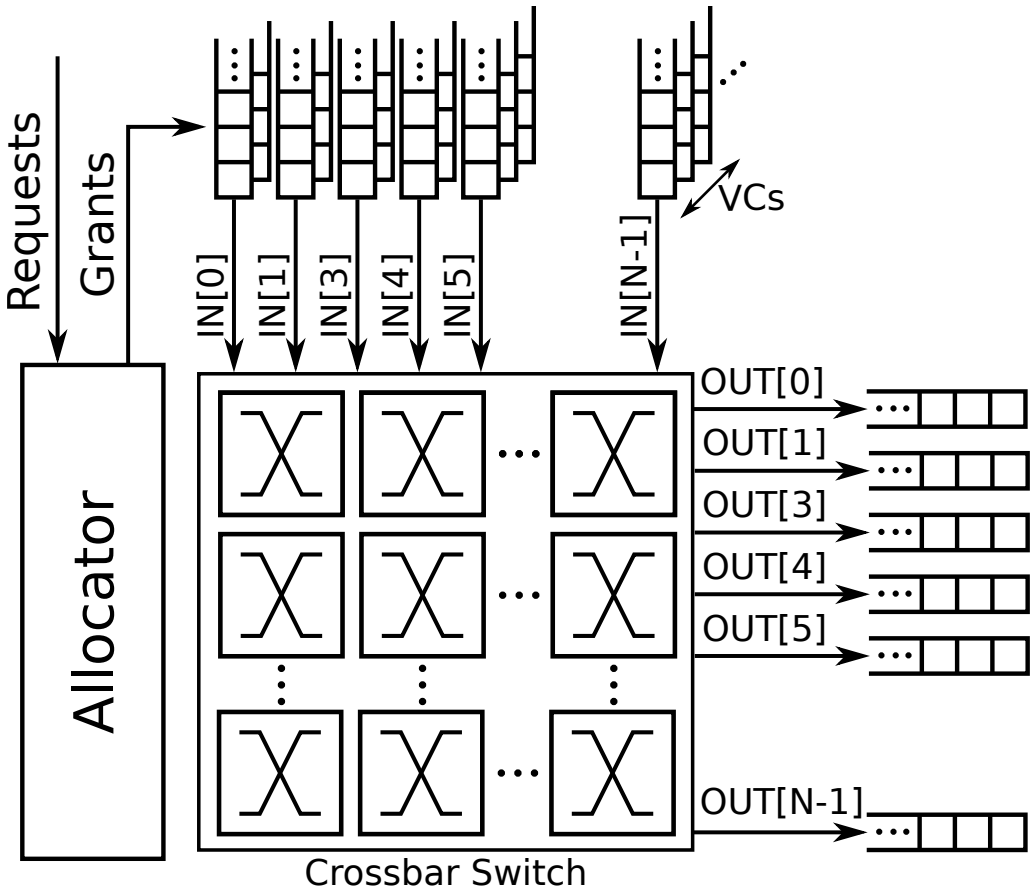


Figure 6.1: Modular crossbar architecture with modular flow-through switch, centralized allocator, and combined input and output queueing with virtual channels(VC).

at the network level.

Modular crossbar architectures consist of modular switches, centralized allocators, and input and output buffers as seen in Figure 6.1. It takes multiple cycles to transfer a single flit through the modular switch, however modular crossbars still offer one-hop connectivity and simplicity. Centralized allocators schedule data flow before the multi-cycle data traversal, and every input port has a dedicated route to every output port. Therefore, modular crossbar programming is as simple as conventional crossbars.

Combined input and output queueing with virtual channels offers high-saturation capacity with internal speedup from the switch and adopted as the input and output buffering scheme for the modular crossbars. Further, cross-*i*SLIP architectures ([44]) demonstrate that building fast and small schedulers are possible at high-radices, and the scheduling latency can be amortized over total data packet transfer through the modular switch.

6.2.1 Input and Output Buffers

Input and output buffers ensure that packets are not lost when conflicts occur (when multiple inputs select the same output). Output queued (OQ) architectures place buffers at the output ports, but require high internal speedup when multiple inputs want to transmit data to the same output. Speedup can be achieved by using a higher clock frequency or implementing multiple write ports.

Input queued (IQ) architectures, on the other hand, locate the buffers at the input ports. There architectures do not require internal speedup, however they suffer from head-of-line (HoL) blocking (packet destined for one port is blocked behind a packet destined for another port). It has been shown that HoL limits the throughput of an input-queued switch to approximately 58.6% under certain conditions [20, 25].

Wide variety of solutions has been proposed to eliminate HoL and achieve 100% saturation throughput. Some of the commonly used solutions are placing multiple queues at the input

ports, placing buffers at crosspoints, and accepting internal speedup. In IQ architectures with virtual output queueing (VOQ) [2, 34, 62], every input port has multiple queues dedicated for every output port, hence every input data packet is located in the queue destined for the requested output port. However, in this scheme, required number of queues scales quadratically with radix, and the memory cost can be significant at high-radices. On the other hand, buffered crossbar switch organization places buffers at every crosspoint [31, 48]. Every input is connected to multiple output buffers, and every output buffer is destined to the requested output port, therefore this organization also eliminated HoL. However, similar to VOQ, the memory cost scales quadratically wth radix.

Another viable approach to eliminate HoL is to accept internal speedup. Combined input and output buffering (CIOQ) offers a solution by assuming some internal speedup and using both input and output buffers [13, 42, 59, 71]. Input buffers can be implemented with virtual channels (VC) [15] to help eliminate HoL and output buffers are required to absorb the internal speedup. We adopt CIOQ for modular crossbars as it offers linear memory resource scaling with radix, and high network saturation capacity with reasonable internal speedup. Modular crossbar architectures support very high data line rates as switches run at high frequencies and the required speedup for maximum saturation capacity is low.

6.2.2 Allocation

Allocation handles access to multiple output resources of multiple input resources. For every shared output resource, arbiters are used to decide which input gets access to the resource. Every output resource can be granted only to requested inputs, inputs are granted access to at most one output resources, and output resources can be assigned to at most one inputs. Good allocator traits are maximizing the matching between the requests and the grants and having low implementation complexity, and hence area, power, and latency [5, 19, 65].

Separable allocators are commonly used in crossbars for scheduling where input and output ports have separate arbiters that run in parallel and exchange matching decisions to con-

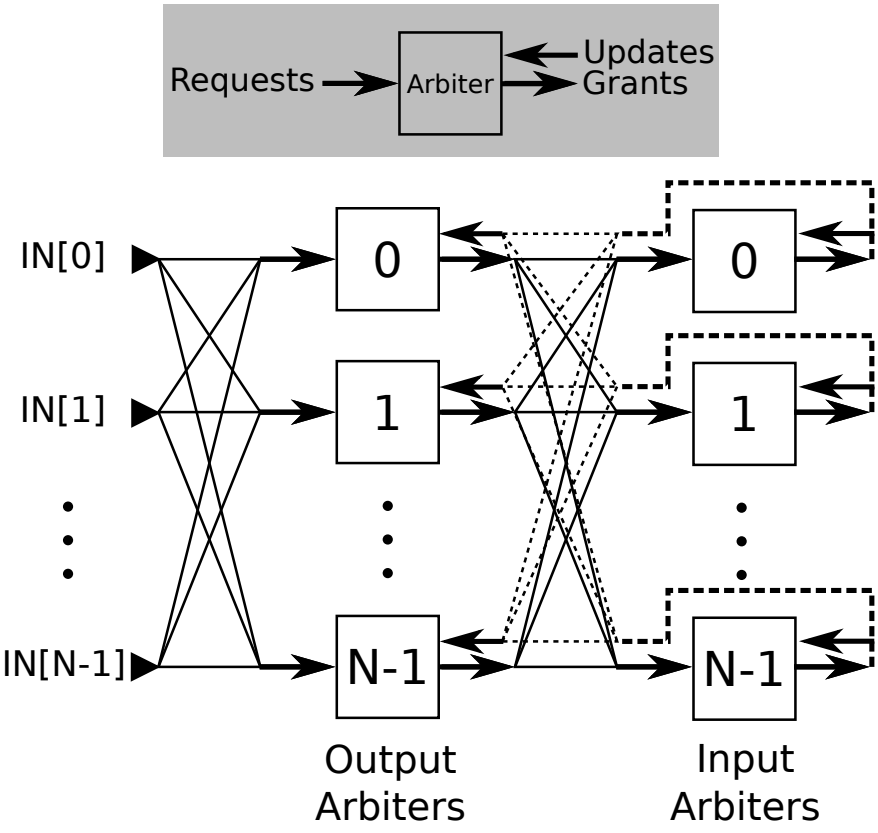


Figure 6.2: Output-first eperable allocation.

sent on a match. They usually employ fast heuristics to find a good matching that is not guaranteed to be maximal. However, they are very popular since they can be parallelized and pipelined to improve the latency, and matching can be improved with more iterations. Allocation is performed in two sets of arbitration, once for inputs and once for outputs. Input-first seperable allocator performs input arbitration first, whereas an output-first allocator performs output arbitration first. An example output-first seperable allocator can be seen in Figure 6.2.

Different seperable allocation methods are parallel iterative matching (PIM) [2, 28], iSLIP, and lonely output allocators (LOA). PIM performs random seperable allocation iteration by using randomized priorities for every iteration. *i*SLIP, on the other hand, uses round-robin arbiters [57] and updates the priority only if the arbiter has got a grant [33, 70]. Finally, LOA adds an extra stage before the input abiters that gives the priorities to the outputs with the least requests to increase output utilization.Previous work [16] has shown that these allocators with low complexity can achieve reasonable performance that can be further increased with multiple iterations.

Passas *et al.* [44] explore scaling of separable allocators using the *i*SLIP arbitration algorithm. Two prominent allocation scaling challenges are quadratic scaling of number of gates in the arbiters and potentially more than quadratic scaling of global point-to-point data links connecting input and output arbiters (as can be seen in Figure 6.2). The proposed microarchitecture called cross-*i*SLIP inverts the locality of wires by orthogonally interleaving the input with output arbiters to decrease the cost of global data links and shows that crossbar allocators (for VOQ architecture) are small and fast for even very high radices. It is extended to virtual channel (VC) & switch allocators used in BookSim2 router architecture [24, 26] that consists of a seperate VC allocator which allocates VCs to packets and a switch allocator that allocates crossbar to the flits of packets.

Since allocation is performed only once for every packet, scheduling latency can be masked over switch traversal latency of packets (assuming constant packet size). The critical path for

radix-128 cross-*i*SLIP implementation is 4ns for seperable input output allocators and below 5ns for VC & switch allocators (scaled to 40nm CMOS bulk [44]). Therefore, for a radix-128 64 bits modular crossbar switch (runs at 2.4GHz using radix-32 blocks), the allocation latency can be amortized over switch traversal latency for packets larger than 80Byte and 96Byte for separable input output allocators and VC & switch allocators respectively.

6.2.3 Architectural Evaluation

To understand the network performance, we evaluate modular crossbar by analyzing how much internal speedup is required to increase network capacity and by obtaining load-latency curves for saturation capacity and zero-load latency measurements using BookSim2 network on chip evaluation tool. Internal speedup is the ratio of switch bandwidth to the data line bandwidth. Speedup increases switch throughput and simplifies allocation.

Evaluation metrics are:

Supported Data Line Rate: Switch data line rate divided by speedup. Given modular switch performance, supported data line rates decrease as accepted internal speed up increases.

Load-Latency Curves: Drawn by measuring average network under different loads for a specified network traffic. Load is the average amount of injected flits by each input port per cycle and latency is the average latency of a packet from the time first flit is injected to the input port when the last flit is delivered to the output port.

Saturation Capacity: Load where the first channels saturates and network achieves saturation throughput.

Zero-Load Latency: Assumes that a packet never contends for resources with other packets and gives a lower bound on the average packet latency.

Experimental Setup

We used BookSim2 [6, 24], a detailed, cycle-accurate network on chip simulator to evaulate the network performance of modular crossbars. It offers a wide variety of configurable network parameters in terms of topology, routing algorithm, flow control, and router microarchictecture. The network components in the tool accurately model the actual hardware and matches the RTL implementation results. Top level networks comprise a collection of routers and channels, however the crossbar topology is modeled as a single router network. The router model is the input-queued virtual channel router implemented with canonical 4-stage pipeline for routing/queueing, VC allocation, switch allocation, and crossbar traversal. The chosen routing and scheduling algorithms are destionation tag and *i*SLIP due to their simplicity and efficiency. The input queues use 4, 8, and 16 virtual channels with 8 buffers each for radix-64, radix-128, and radix-256 crossbars respectively.

In order to model the modular crossbar network, we modified the crossbar traversal stage of the 4-stage router pipeline. Crossbar traversal latency (switch latency) of modular crossbar is k times the block latency and the operation is pipelined. Therefore, the single cycle crossbar traversal in the router pipeline is modified as k cycles. As mentioned, radix-32 64 bits switch block (in 40nm CMOS bulk) achieves maximum data throughput, runs at 2.38 GHz, and supports data line rates as high as 156.6 Gbps. This high switch performance allows us to trade off some speed as internal speedup to overcome performance degradation due to high-radix centralized allocation and I/O buffering. By assuming that switches run at higher frequencies than the I/O line rates, higher saturation capacity and lower zero-load latency can be achieved. The simulator adds output queues to the input-queued virtual channel router implementation to absorb the internal speedup, thus modular crossbar routers have combined input and output queued architectures.

We compared network performance of modular crossbars to conventional crossbars for evaluation metrics of supported data line rates, saturation capacity, and zero-load latency. BookSim2 is a cycle-accurate simulator, therefore it assumes both conventional and mod-

Table 6.1: Radix-64 Modular Crossbar

Speedup (SU)	1	1.5	2	4
Saturation TP (flits/cycle)	0.62	0.93	0.98	0.98
Data Line Rate (Gbps)	153.6	102.4	76.8	38.4

ular crossbar switches have the same switch traversal latency. However, as discussed in the switch evaluation section, modular switches offer significantly better latency than conventional switches. To be fair, we assumed they run at the same frequency (2.38GHz) except the switch traversal stage. We back annotated switch traversal latencies from the testchip results and using the crossbar modeling tool for the optimized designs for both the conventional and modular crossbars at different radices.

Data Line Rates

High-radix modular crossbars can be built using radix-32 switch blocks and run at the same frequency as the blocks. Switch frequency is an important metric as it determines the supported data line rates with given internal speedup. Table 6.1 shows how internal speedup changes the saturation throughput and the resulting supported line rates for a radix-64 modular crossbar. The saturation capacity is only 62% without any speedup and can be increased above 90% with only 1.5 speedup. Although increasing the speedup further increases the saturation capacity and lowers the zero-load latency, the supported line rates decrease significantly. Available off chip I/O bandwidth increases every year [27], and modular crossbar offers an option to support very high line rates that achieve at least 90% saturation capacity. However, if the available line rates are smaller, assuming a larger speedup will further improve the saturation capacity and zero-load latency.

Load vs Latency

We simulated crossbar performance under uniform traffic. Figure 6.3, 6.4, 6.5 show load-latency curves for conventional crossbars and modular crossbars without speedup and with

speedups of 1.5, 2, and 4 for radices 64, 128, and 256. BookSim2 is a detailed simulator that is not optimized for high-radix networks and slows down significantly at high loads. Therefore load-latency curves are generated for single-flit packets. As can be seen, both conventional and modular crossbar networks without speedup saturate at 60% capacity (0.6 load) while modular crossbars with speedup saturate at more than 90% capacity for all radices. We accepted speedup for the modular crossbars since the modular switches can run at higher frequencies (with pipelining) compared to conventional switches as discussed.

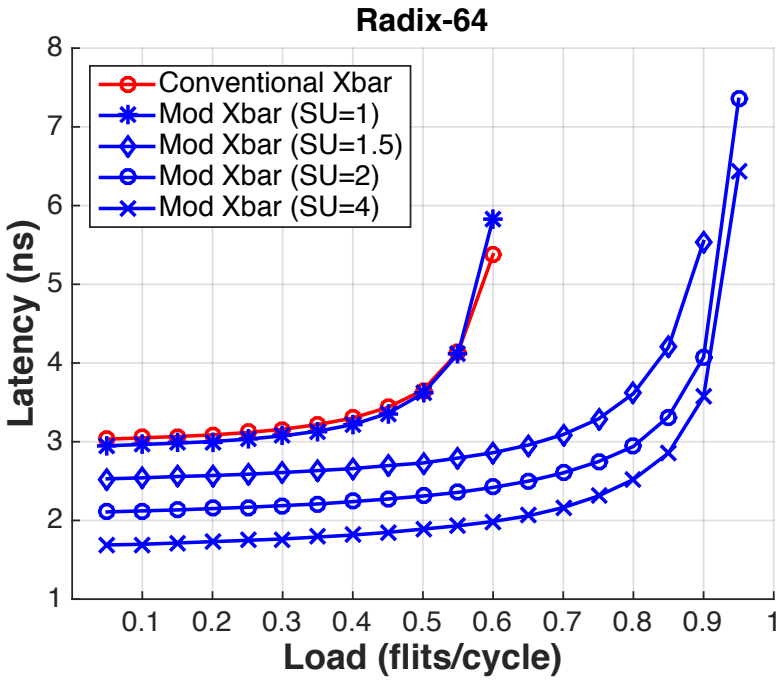


Figure 6.3: Load-latency curves for radix-64 conventional and modular crossbars (with speedup 1, 1.5, 2, and 4).

Comparison of zero-load latencies can be seen from the load-latency curves and are also shown implicitly in Table 6.2 for different radices. We plugged in the latency results for the crossbar traversal stage for both canonical and modular crossbars, and report average network latencies from cycle results of the simulator and the modeled radix-32 frequency. For 1.5 speedup (minimum SU that can achieve 90% throughput), modular crossbars offer 1.2X, 1.3X, 1.6X, 2.7X lower network latency than conventional crossbars for radices 64,

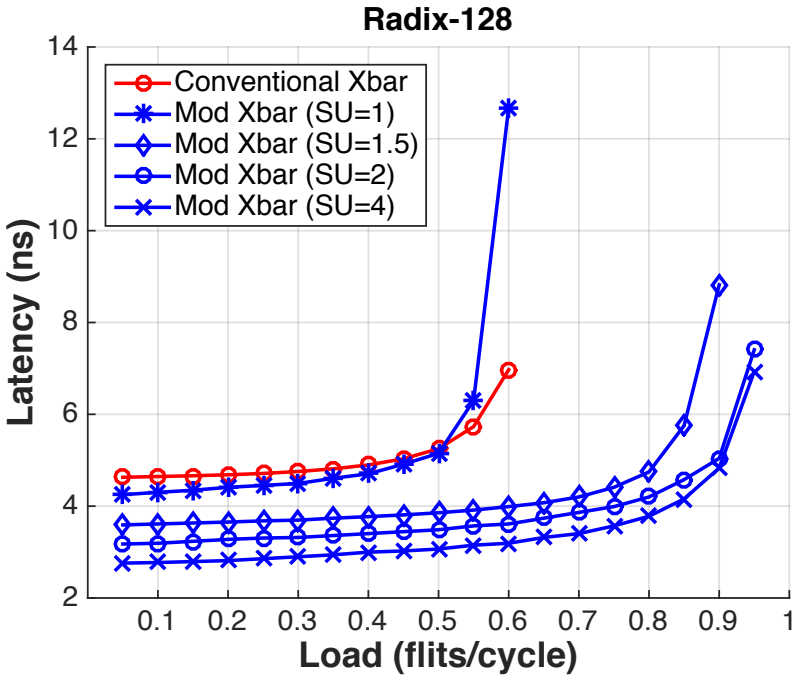


Figure 6.4: Load-latency curves for radix-128 conventional and modular crossbars (with speedup 1, 1.5, 2, and 4).

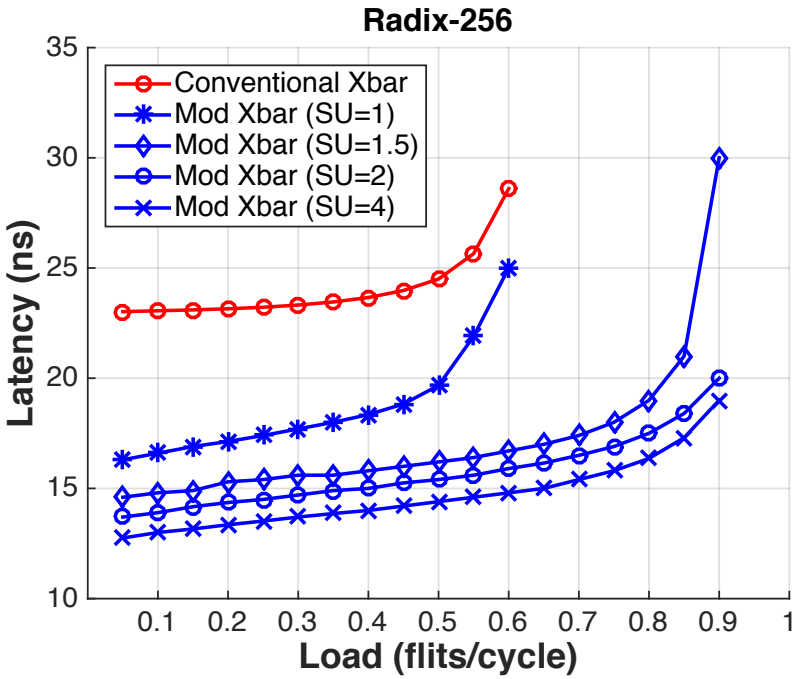


Figure 6.5: Load-latency curves for radix-256 conventional and modular crossbars (with speedup 1, 1.5, 2, and 4).

Table 6.2: Zero-Load Latency for a Single-Flit Packet

N	64	128	256	512
Conventional Xbar	3ns	4.5ns	9.5ns	27ns
Modular Xbar (SU=1.0)	2.9ns	4.2ns	6.8ns	12ns
Modular Xbar (SU=1.5)	2.5ns	3.5ns	5.8ns	11.2ns
Modular Xbar (SU=2.0)	2ns	3.1ns	5.6ns	11ns
Modular Xbar (SU=4.0)	1.7ns	2.7ns	5.3ns	10.6ns

Table 6.3: Zero-Load Latency for a 32Byte Packet

N	64	128	256	512
Conventional Xbar	5.8ns	12ns	32ns	103ns
Modular Xbar (SU=1.0)	4.15ns	5.4ns	8ns	13.28ns
Modular Xbar (SU=1.5)	3.7ns	4.8ns	7ns	12.4ns
Modular Xbar (SU=2.0)	3.3ns	4.3ns	6.9ns	12.2ns
Modular Xbar (SU=4.0)	2.9ns	3.9ns	6.5ns	11.8ns

128, 256, and 512. Moreover, assuming a larger speedup such as 2 or 4 decreases zero-load latency further, since there is less congestion in the network for all radices as can be seen in Figure 6.6.

Single-flit packet simulations are actually the worst case simulations for modular crossbars for zero-load latency comparison, since it takes multiple cycles to transfer the initial flit and the rest of the flits in the packet can be pipelined. Therefore, we also report zero-load results for a 32Byte packet (a common cache line size, 4 flits for a 64 bits crossbar switch) in Table 6.3. With the larger packet size, modular crossbars with 1.5 SU offer 1.6X, 3.2X, 6.6X, and 8.3X lower zero-load latency than conventional crossbars for radices 64, 128, 256, and 512 respectively as can be seen in Figures 6.7.

6.3 Summary

We present modular crossbars using the modular switch, centralized allocators, and input (with virtual channels) and output buffers. The internal speedup of the switch increases the network saturation capacity and lowers the average network latency compared to conventional crossbar switches. With process scaling, future processor and SoC chips are expected

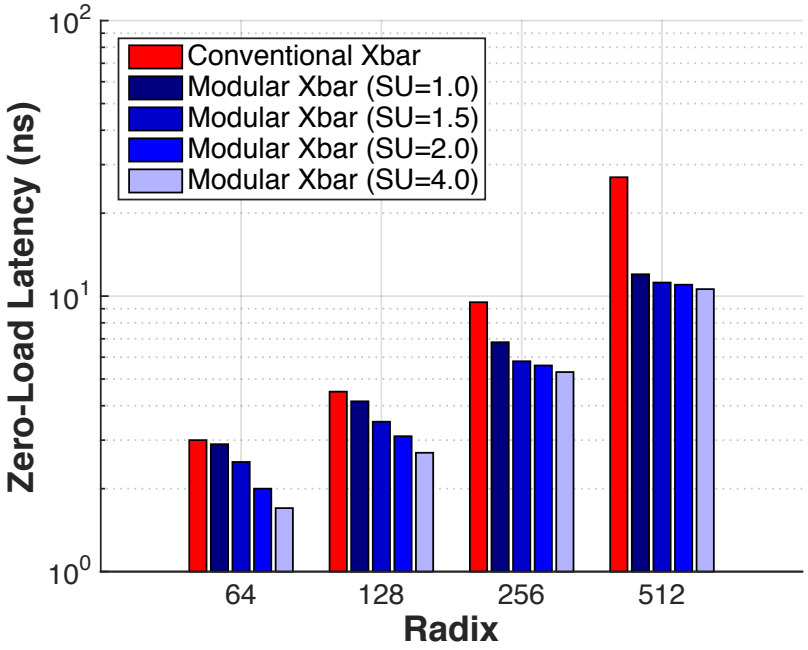


Figure 6.6: Zero-load latency for a single-flit packet transfer.

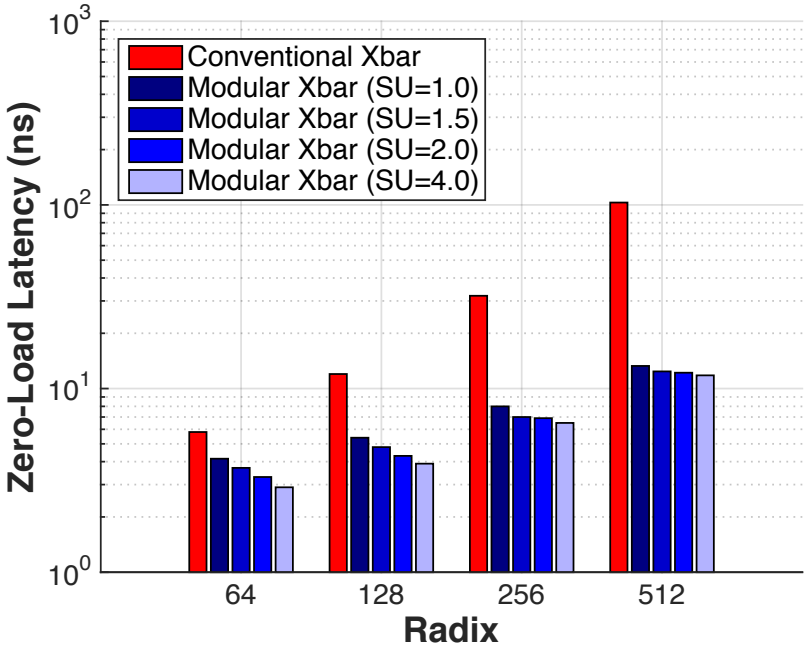


Figure 6.7: Zero-load latency for a 32 Byte packet transfer.

to have increased core counts, cache sizes, and off-chip I/O bandwidths, which will lead to increasing on-chip communication demands. High-radix modular crossbars can meet these performance requirements while offering simplicity and one-hop latency, and thus are promising alternatives to multi-hop networks at high radices.

Chapter 7

Conclusions

Processor cores and memories are getting smaller, faster, and inexpensive with every new process technology. This, combined with increasing concerns for power density, led to a growing interest in system-on-chip designs with increasing number of components. However, performance and density of the wires that connect these system components scale at a lower rate than the components, since speed of light remains unchanged. These factors have made communication one of the major bottlenecks and driven the need for more complex on-chip interconnection networks.

Crossbar switches are primary building blocks in such networks-on-chip, as they can be used as fast single-stage networks or as the core of the router switch in multi-stage networks. While crossbars offer non-blocking, single-hop, all-to-all communication, they tend to scale poorly with the number of nodes due to quadratic area growth, high-radix multiplexer structures needed, and significant latency and energy consumption. Therefore, multiple recent many-core processor designs have opted for multi-stage networks instead of single-stage crossbar networks. In this thesis, we show scalability of crossbars to high-radices by presenting a modular crossbar architecture with concentrated focus on high-performance and energy-efficient switch design using various circuit techniques and layout details.

We present a detailed analysis and modeling of crossbar switch architecture to understand scaling limitations and to improve performance and energy-efficiency. In Chapter 2, we introduce an analytical crossbar switch modeling tool calibrated using post-layout extracted simulations for fast design space exploration. Proposed tool covers both system level (number of modules and data width) and circuit level (multiplexer circuits, wire pitches, etc.) design parameters to achieve design optimization as close to full custom design as possible. Our design space exploration suggests that smaller and simpler designs offer better performance due to shorter input, output and internal wires.

To further improve energy consumption, we explore low swing signaling in Chapter 3. We present a low-swing crossbar scheme that uses capacitively coupled wires and multiplexers to improve the performance and efficiency. We discuss the implementation details of a radix-16 low-swing crossbar and the evaluation results show it operating at a significantly improved energy-efficiency. However, due to area overhead of differential wires and sense amplifiers to restore the low swing signals, low swing crossbar latency increases significantly with radix.

From our design space exploration, we conclude that, for low-to-medium radices, I/O wires are shorter and multiplexers have a significant impact on the latency, therefore optimizing multiplexer performance can improve the total switch performance even at the cost of larger area. Even further, adopting low-swing signaling techniques will improve the energy-efficiency since the benefits are more significant than the overheads. However, for high-radices, circuit techniques that will overhead in area result in substantially longer I/O wires, hence increased latency and energy consumption. Such techniques include tree multiplexers that require extra wires tracks for intermediate connections, centralized multiplexer implementations with extra input wire routings, and low-swing signaling with differential wiring.

Area minimalization is especially a concern for high-radices. Even with minimum wire pitches, the minimum design area is quite large due to large datawidths (usually 64-128

bits) and high-radices. Thus, building a high-performance and efficient high-radix switch depends highly on wire engineering. The multiplexers can be viewed as wire drivers, and the main object of the design becomes building efficient wire drivers and keeping the wire lengths optimal. Dynamic tri-state buffers has shown to be efficient drivers for crossbar multiplexers due to their small area and high logical effort. The parasitic gate loading from unselected inputs of the multiplexers are also negligible compared to the large wire capacitances for high-radix switches.

Using these insights, we present modular crossbar switches in Chapter 4. As we concluded that building high-radix crossbar switches are similar to driving long wires, we were inspired by the repeater insertion to eliminate quadratic wire latency scaling. Using smaller blocks to build the high-radix crossbar switches keeps the I/O wire lengths short and eliminates the quadratic wire latency scaling. The blocks are arranged in a controlled flow-through, pipelined scheme to eliminate global connections and maintain linear performance scaling and high throughput. Small sub-block sizing and modularity enable deactivating unused I/O wires to improve energy efficiency. To evaluate our design, we implemented and designed a prototype radix-64, 64 bits modular crossbar switch testchip in 40nm CMOS bulk process as explained in Chapter 5. It operates at 2.38GHz at 1V nominal supply voltage and comsumes 1.2W power. Throughput is 9.75 Tb/ps, 2.2X better than previously published results and the energy-efficiency is 8.2 Tb/ps/W, 2.4X better than previously published results. We further evaluate modular crossbar architectures using the modular crossbar switches in Chapter 6. The internal speedup of the switch increases the network saturation capacity and lowers the average network latency compared to conventional crossbar switches.

Our evaluation results demonstrate that modular crossbar switches offer a viable solution to crossbar scaling problem. The area still scales quadratically, however latency scales linearly and modularity allows significant energy savings. Our preliminary network evaluations, on the other hand, suggest that crossbar networks are also scalable to high-radices. With process scaling, future processor and SoC chips are expected to have increased core

counts, cache sizes, and off-chip I/O bandwidths, which will lead to increasing on-chip communication demands. High-radix modular crossbars have potential meet these performance requirements while offering simplicity and one-hop latency, and thus are promising alternatives to multi-hop networks at high radices.

7.1 Future Work

The ideas presented in this thesis are open to further exploration, improvement, and new directions. Although modular crossbar switches offer linear scalability, better evaluation of the crossbar network would be beneficial at extremely high-radices like 512. The die floorplan, available wire resources, global communication with the network, and network workloads are some of the concerns for extremely high-radices that can affect the single-hop crossbar performance. These crossbars can be a very efficient way to exploit increasing off chip memory bandwidth by offering uniform latency for all the system components. However, they might not be the best option for application that requires extensive local communication.

Some of the areas for further improvement are modular switch energy savings, modular crossbar network implementation, modular allocators, and comparison to multi-stage networks.

Modular Crossbar Switch Power-Gating

Modular crossbar switch design improves energy efficiency by deactivating unused I/O wires. This scheme decreases energy consumption significantly since the switching energy of the wires have a major impact on the total energy consumption. However, benefits of this scheme can be further enhanced by power-gating I/O ports or the switch blocks. For the radix-64, 64 bits modular crossbar implementation, idle energy consumption of an I/O port is 25% of the total energy consumption, thus deactivating power supply rather than wires

can further decrease the total energy consumption. Naturally, controlling power supply of every I/O port overheads in area, latency, and energy consumption. However, this also means smaller power grids to control, and latency of power-gating can be amortized over multi-cycle flit and packet transfer. Power-enable signal for I/O ports can be generated ahead of data transfer to allow enough time for power grid to turn on. This high-granularity of power control have potential to offer compelling energy savings. Power-gating can be done at the block level instead of I/O port level as well, however energy savings might not be substantial. Block power can be turned off if only none of the ports are active, which is unlikely for blocks with large number of ports.

Modular Crossbar Network Implementation

Modular crossbar architecture presented in this thesis is evaluated using a cycle-accurate network evaluation tool. The implementation results for the modular switch are back annotated to the tool to achieve more realistic evaluation results. However, the allocators and buffers are not implemented with the switch. Rather, results from previously published papers and the assumptions from the tool are used. Therefore, a full crossbar network implementation will offer a better evaluation in terms of area, latency, and energy overhead of the allocators and the I/O buffers.

Modular Allocators

Modularity and flow-through operation can be extended to allocators. Hierarchical and distributed allocators has been shown to be scalable to high-radices [1, 32]. However, due to arbitration among smaller groups, these allocators need to implement extra features to achieve fairness. Further, extra top level wiring is required to exchange matching decision across different groups. We can build modular flow-through allocators using smaller blocks that operates in a similar fashion to our modular switches. The decision of every group can flow through the blocks to avoid extra wiring and can be pipelined to achieve high

throughput. To improve fairness, priority of the decisions from previous groups can be weighted.

Comparison to Multi-Stage Networks

Crossbars have shown to achieve better network latency and saturation than multi-stage networks like meshes [26, 43] for radices as high as 64, 128. However, modular crossbar networks can scale to very high radices like 512. Although the crossbar switch can scale linearly to these radices, a better evaluation of the crossbar network would be beneficial. Crossbars require global data links to connect the components to the network, and for large number of components and increased die area, global data link communication should be included in the evaluation. On the other hand, mesh networks have distributed routers close to the network components throughput the design and short data links to connect these routers. Therefore, comparison to mesh networks at very high-radices would be very beneficial. Further, different floorplans and applications can be evaluated for a more through comparison.

Bibliography

[1] Jung Ho Ahn, Sungwoo Choo, and J. Kim. Network within a network approach to create a scalable high-radix router microarchitecture. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12, Feb 2012. doi: 10.1109/HPCA.2012.6169048. 6.1, 7.1

[2] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High-speed switch scheduling for local-area networks. *ACM Trans. Comput. Syst.*, 11(4): 319–352, November 1993. ISSN 0734-2071. doi: 10.1145/161541.161736. URL http://doi.acm.org/10.1145/161541.161736. 6.2.1, 6.2.2

[3] R. Ausavarungnirun, C. Fallin, X. Yu, K. K. W. Chang, G. Nazario, R. Das, G. H. Loh, and O. Mutlu. Design and evaluation of hierarchical rings with deflection routing. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on*, pages 230–237, Oct 2014. doi: 10.1109/SBAC-PAD.2014.31. 1.1

[4] James Balfour and William J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06, pages 187–198, New York, NY, USA, 2006. ACM. ISBN 1-59593-282-8. doi: 10.1145/1183401.1183430. URL http://doi.acm.org/10.1145/1183401.1183430. 1.1, 2.1

[5] D. U. Becker and W. J. Dally. Allocator implementations for network-on-chip routers.

In *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, pages 1–12, Nov 2009. doi: 10.1145/1654059.1654112. 6.2.2

[6] Daniel U Becker. *Efficient microarchitecture for network-on-chip routers*. PhD thesis, Stanford University, 2012. 1.1, 6.2.3

[7] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C. C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 88–598, Feb 2008. doi: 10.1109/ISSCC.2008.4523070. 1.1

[8] Cagla Cakir, Ron Ho, Jon Lexau, and Ken Mai. High-efficiency crossbar switches using capacitively coupled signaling. In *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*, pages 98–103. IEEE, 2015. 3

[9] Cagla Cakir, Ron Ho, Jon Lexau, and Ken Mai. Modeling and design of high-radix on-chip crossbar switches. In *Proceedings of the 9th International Symposium on Networks-on-Chip*, NOCS '15, pages 20:1–20:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3396-2. doi: 10.1145/2786572.2786579. URL http://doi.acm.org/10.1145/2786572.2786579. 4.2.1, 4.3.1

[10] L. P. Carloni, P. Pande, and Y. Xie. Networks-on-chip in emerging interconnect paradigms: Advantages and challenges. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, pages 93–102, May 2009. doi: 10.1109/NOCS.2009.5071456. 1.1

[11] Kun-Yung Ken Chang, Shang-Tse Chuang, N. McKeown, and M. Horowitz. A 50 gb/s 32/spl times/32 cmos crossbar chip using asymmetric serial links. In *VLSI Circuits, 1999. Digest of Technical Papers. 1999 Symposium on*, pages 19–22, June 1999. doi: 10.1109/VLSIC.1999.797221. 2.1

[12] N. Chrysos, C. Minkenberg, M. Rudquist, C. Basso, and B. Vanderpool. Scoc: High-radix switches made of bufferless clos networks. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pages 402–414, Feb 2015. doi: 10.1109/HPCA.2015.7056050. 1.2, 6.1

[13] Shang-Tse Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued switch. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1169–1178 vol.3, Mar 1999. doi: 10.1109/INFCOM.1999.751673. 6.2.1

[14] W Dally, Wayne Dettloff, John Eyles, Trey Greer, John Poulton, Teva Stone, and Steve Tell. A single-chip terabit switch. 1.1

[15] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, Mar 1992. ISSN 1045-9219. doi: 10.1109/71.127260. 6.2.1

[16] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689, 2001. doi: 10.1109/DAC.2001.156225. 1.1, 6.2.2

[17] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, Oct 1974. ISSN 0018-9200. doi: 10.1109/JSSC.1974.1050511. 1.1

[18] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 365–376, June 2011. 1.1

[19] P. Gupta and N. McKeown. Designing and implementing a fast crossbar scheduler. *IEEE Micro*, 19(1):20–28, Jan 1999. ISSN 0272-1732. doi: 10.1109/40.748793. 6.2.2

[20] M. G. Hluchyj and M. J. Karol. Queueing in high-performance packet switching. *IEEE Journal on Selected Areas in Communications*, 6(9):1587–1597, Dec 1988. ISSN 0733-8716. doi: 10.1109/49.12886. 6.2.1

[21] R. Ho, K.W. Mai, and M.A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr 2001. ISSN 0018-9219. doi: 10.1109/5.920580. 2.3.2

[22] R. Ho, T. Ono, R.D. Hopkins, A. Chow, J. Schauer, F.Y. Liu, and R. Drost. High speed and low energy capacitively driven on-chip wires. *Solid-State Circuits, IEEE Journal of*, 43(1):52–60, Jan 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2007.910807. 3, 3.1, 3.1.1, 3.1.1, 3.1.2

[23] M. Horowitz, Chih-Kong Ken Yang, and S. Sidiropoulos. High-speed electrical signaling: overview and limitations. *IEEE Micro*, 18(1):12–24, Jan 1998. ISSN 0272-1732. doi: 10.1109/40.653013. 1.1

[24] Nan Jiang, D.U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D.E. Shaw, J. Kim, and W.J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96, April 2013. doi: 10.1109/ISPASS.2013.6557149. 6.2.2, 6.2.3

[25] Mark J Karol, Michael G Hluchyj, and Samuel P Morgan. Input versus output queueing on a space-division packet switch. *Communications, IEEE Transactions on*, 35(12): 1347–1356, 1987. 6.2.1

[26] John Kim. *High-Radix Interconnection Networks*. PhD thesis, Stanford University, 2008. 1.1, 6.1, 6.2.2, 7.1

[27] John Kim, William J Dally, Brian Towles, and Amit K Gupta. Microarchitecture of a high-radix router. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 420–431. IEEE Computer Society, 2005. 1.1, 1.2, 6.1, 6.2.3

[28] N. Kitsuwan, E. Oki, R. Rojas-Cessa, and N. Kishi. Performance analysis on dynamics of parallel iterative matching in an input-buffered switch. In *Communications, 2009. APCC 2009. 15th Asia-Pacific Conference on*, pages 821–825, Oct 2009. doi: 10.1109/APCC.2009.5375476. 6.2.2

[29] 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 *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 439–446, Oct 2010. doi: 10.1109/ICCD.2010.5647666. 3

[30] M. J. E. Lee, W. J. Dally, R. Farjad-Rad, H. T. Ng, Ramesh Senthinathan, J. Edmondson, and J. Poulton. Cmos high-speed i/os - present and future. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 454–461, Oct 2003. doi: 10.1109/ICCD.2003.1240940. 1.1

[31] M. Lin and N. McKeown. The throughput of a buffered crossbar switch. *IEEE Communications Letters*, 9(5):465–467, May 2005. ISSN 1089-7798. doi: 10.1109/LCOMM.2005.1431173. 6.2.1

[32] Jing Liu, M. Hamdi, Qingsheng Hu, and C. Y. Tsui. Scalable scheduling architectures for high-performance crossbar-based switches. In *High Performance Switching and Routing, 2004. HPSR. 2004 Workshop on*, pages 104–110, 2004. doi: 10.1109/HPSR.2004.1303441. 7.1

[33] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, Apr 1999. ISSN 1063-6692. doi: 10.1109/90.769767. 6.2.2

[34] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47 (8):1260–1267, Aug 1999. ISSN 0090-6778. doi: 10.1109/26.780463. 6.2.1

[35] E. Mensink, D. Schinkel, E. Klumperink, E. van Tuijl, and B. Nauta. A 0.28pj/b

2gb/s/ch transceiver in 90nm cmos for 10mm on-chip interconnects. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 414–612, Feb 2007. doi: 10.1109/ISSCC.2007.373470. 3.1.1

[36] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, Jan 1998. ISSN 0018-9219. doi: 10.1109/JPROC.1998.658762. 1.1

[37] Gaspar Mora, Jose Flich, Jose Duato, Pedro Lopez, Elvira Baydal, and Olav Lysne. Towards an efficient switch architecture for high-radix switches. In *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, pages 11–20. ACM, 2006. 6.1

[38] B. Nikolic, Vojin G. Oklobdzija, V. Stojanovic, Wenyan Jia, James Kar-Shing Chiu, and M. Ming-Tak Leung. Improved sense-amplifier-based flip-flop: design and measurements. *Solid-State Circuits, IEEE Journal of*, 35(6):876–884, June 2000. ISSN 0018-9200. doi: 10.1109/4.845191. 3.1.5

[39] Michael K Papamichael and James C Hoe. Connect: re-examining conventional wisdom for designing nocs in the context of fpgas. In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, pages 37–46. ACM, 2012. 2.1

[40] Michael K Papamichael, Cagla Cakir, Chen Sun, Chia-Hsin Owen Chen, James C Hoe, Ken Mai, Li-Shiuan Peh, and Vladimir Stojanovic. Delphi: A framework for rtl-based architecture design evaluation using dsent models. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015. 2.1

[41] G. Passas, M. Katevenis, and D. Pnevmatikatos. Crossbar nocs are scalable beyond 100 nodes. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(4):573–585, April 2012. ISSN 0278-0070. doi: 10.1109/TCAD.2011.2176730. 1.1, 1.1, 1.2, 2.1, 2.2.1, 4.1, 5.6, 5.2

[42] G. Passas, M. Katevenis, and D. Pnevmatikatos. The combined input-output queued

crossbar architecture for high-radix on-chip switches. *IEEE Micro*, 35(6):38–47, Nov 2015. ISSN 0272-1732. doi: 10.1109/MM.2014.56. 6.2.1

[43] Giorgos Passas. *VLSI Micro-Architectures for High-Radix Crossbars*. PhD thesis, University of Crete, 2012. 2.1, 7.1

[44] Giorgos Passas, Manolis Katevenis, and Dionisios Pnevmatikatos. Vlsi micro-architectures for high-radix crossbar schedulers. In *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, pages 217–224. ACM, 2011. 6, 6.2, 6.2.2

[45] J. Postman, T. Krishna, C. Edmonds, Li-Shiuan Peh, and P. Chiang. Swift: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(8):1432–1446, Aug 2013. ISSN 1063-8210. doi: 10.1109/TVLSI.2012.2211904. 2.1, 3

[46] R. Proebsting, R. Ho, and R. Drost. Method and apparatus for routing differential signals across a semiconductor chip. *Patent 7,139,993*, 2006. 3.1.2

[47] Y. Ren, H. An, M. Cong, G. Xu, and L. Wang. Scaling the performance of tiled processor architectures with on-chip-network topology. In *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 1, pages 77–81, April 2009. doi: 10.1109/CSO.2009.233. 1.1

[48] R. Rojas-Cessa, E. Oki, and H. J. Chao. Cixob-k: combined input-crosspoint-output buffered packet switch. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 4, pages 2654–2660 vol.4, 2001. doi: 10.1109/GLOCOM.2001.966256. 6.2.1

[49] J. s. Seo, R. Ho, J. Lexau, M. Dayringer, D. Sylvester, and D. Blaauw. High-bandwidth and low-energy on-chip signaling with adaptive pre-emphasis in 90nm cmos. In *Solid-*

State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International, pages 182–183, Feb 2010. doi: 10.1109/ISSCC.2010.5433993. 3.1.1

[50] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, P. Kundu, and N. Borkar. A 2tb/s 6x4 mesh network with dvfs and 2.3tb/s/w router in 45nm cmos. In *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, pages 79–80, June 2010. doi: 10.1109/VLSIC.2010.5560277. 3.1

[51] S. Satpathy, R. Dreslinski, T. C. Ou, D. Sylvester, T. Mudge, and D. Blaauw. Swift: A 2.1tb/s 32x32 self-arbitrating manycore interconnect fabric. In *VLSI Circuits (VLSIC), 2011 Symposium on*, pages 138–139, June 2011. 2.1

[52] S. Satpathy, R. Das, R. Dreslinski, T. Mudge, D. Sylvester, and D. Blaauw. High radix self-arbitrating switch fabric with multiple arbitration schemes and quality of service. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 406–411, June 2012. doi: 10.1145/2228360.2228432. 2.1

[53] S. Satpathy, K. Sewell, T. Manville, Yen-Po Chen, R. Dreslinski, D. Sylvester, T. Mudge, and D. Blaauw. A 4.5tb/s 3.4tb/s/w 64x64 switch fabric with self-updating least-recently-granted priority and quality-of-service arbitration in 45nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 478–480, Feb 2012. doi: 10.1109/ISSCC.2012.6177098. 1.1, 2.1, 3.1, 5.6, 5.2

[54] Sudhir Kumar Satpathy. *High Performance and Low Power On-Die Interconnect Fabrics*. PhD thesis, University of Michigan, 2012. 2.1

[55] R. R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, Jun 1997. ISSN 0018-9235. doi: 10.1109/6.591665. 1.1

[56] D. Schinkel, E. Mensink, E.A.M. Klumperink, E. van Tuijl, and B. Nauta. Low-power, high-speed transceivers for network-on-chip communication. *Very Large Scale*

Integration (VLSI) Systems, IEEE Transactions on, 17(1):12–21, Jan 2009. ISSN 1063-8210. doi: 10.1109/TVLSI.2008.2001949. 3, 3.1, 3.1.1

[57] E. S. Shin, V. J. Mooney, and G. F. Riley. Round-robin arbiter design and generation. In *System Synthesis, 2002. 15th International Symposium on*, pages 243–248, Oct 2002. 6.2.2

[58] Dogyoon Song and Jaeha Kim. A low-power high-radix switch fabric based on low-swing signaling and partially-activated input lines. In *VLSI Design, Automation, and Test (VLSI-DAT), 2013 International Symposium on*, pages 1–4, April 2013. doi: 10.1109/VLDI-DAT.2013.6533825. 2.1, 3

[59] I. Stoica and H. Zhang. Exact emulation of an output queueing switch by a combined input output queueing switch. In *Quality of Service, 1998. (IWQoS 98) 1998 Sixth International Workshop on*, pages 218–224, May 1998. doi: 10.1109/IWQOS.1998.675242. 6.2.1

[60] Chen Sun, Chia-Hsin Owen Chen, George Kurian, Lan Wei, Jason Miller, Anant Agarwal, Li-Shiuan Peh, and Vladimir Stojanovic. Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 201–210. IEEE, 2012. 2.1

[61] Ivan E. Sutherland and Robert F. Sproull. Logical effort: Designing for speed on the back of an envelope. In *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI*, pages 1–16, Cambridge, MA, USA, 1991. MIT Press. ISBN 0-262-19308-6. URL http://dl.acm.org/citation.cfm?id=112073.112074. 2.3.2, 6

[62] Yuval Tamir and Gregory L Frazier. *High-performance multi-queue buffers for VLSI communications switches*, volume 16. IEEE Computer Society Press, 1988. 6.2.1

[63] M. B. Taylor. A landscape of the new dark silicon design regime. *IEEE Micro*, 33(5): 8–19, Sept 2013. ISSN 0272-1732. doi: 10.1109/MM.2013.90. 1.1

[64] D. Tsamis, B. Yolken, N. Bambos, W. Olesinski, H. Eberle, and N. Gura. Backlog aware scheduling for ingress memories in high-radix, single-stage switches. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–5, Nov 2009. doi: 10.1109/GLOCOM.2009.5426046. 1.2

[65] Feng Wang and Mounir Hamdi. Fast fair arbiter design in packet switches. In *High Performance Switching and Routing, 2005. HPSR. 2005 Workshop on*, pages 472–476, May 2005. doi: 10.1109/HPSR.2005.1503277. 2.1, 6.2.2

[66] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305, 2002. doi: 10.1109/MICRO.2002.1176258. 2.1

[67] Hangsheng Wang, Li-Shiuan Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 105–116, Dec 2003. doi: 10.1109/MICRO.2003.1253187. 2.1

[68] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010. ISBN 0321547748, 9780321547743. 5.3

[69] Ting Wu, Chi-Ying Tsui, and M. Hamdi. A 2 gb/s 256*256 cmos crossbar switch fabric core design using pipelined mux. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 2, pages II–568–II–571 vol.2, 2002. doi: 10.1109/ISCAS.2002.1011051. 2.1

[70] Bo Xu. Design of an expandable crossbar scheduler based on islip algorithm. In

Communication Technology Proceedings, 2003. ICCT 2003. International Conference on, volume 1, pages 540–542 vol.1, April 2003. doi: 10.1109/ICCT.2003.1209135. 6.2.2

[71] K. Yoshigoe and K. J. Christensen. An evolution to crossbar switches with virtual output queuing and buffered cross points. *IEEE Network*, 17(5):48–56, Sept 2003. ISSN 0890-8044. doi: 10.1109/MNET.2003.1233917. 6.2.1