Petabit Switch-Fabric Design

Ian Juch
Bhavana Chaurasia
Yale Chen
Surabhi Kumar
Jay Mistry

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-88
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-88.html

May 14, 2015

Acknowledgement

Special thanks to our advisors Elad Alon and Vladimir Stojanovic. Also many thanks to the graduate students at BWRC who helped us tremendously with the tools setup for our project: Brian Zimmer, Steven Bailey, Nathan Narevsky, and Krishna Settaluri.

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2015

Electrical Engineering and Computer Science

Integrated Circuits

Petabit Switch Fabric Design

Ian Juch

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

Approved by:

1. Capstone Project Advisor #1:

Signature: _____ Date _____

Print Name/Department: Elad Alon/EECS

2. Capstone Project Advisor #2:

Signature: _____ Date _____

Print Name/Department: Vladimir Stojanovic/EECS

Acknowledgements

Special thanks to our advisors Elad Alon and Vladimir Stojanovic. Also many thanks to the graduate students at BWRC who helped us tremendously with the tools setup for our project: Brian Zimmer, Steven Bailey, Nathan Narevsky, and Krishna Settaluri.

Table of Contents

Shared Team Paper

Problem Statement

The current trend in the computing industry is to offer more performance by leveraging more processing cores. Because we have run into some physical limits on how fast we can make a single processor run, the industry is now finding ways to utilize more cores running in parallel to increase computing speeds. Looking beyond the four and eight core systems we see in commercially available computers today, the natural progression is to scale this up to hundreds or thousands of processing units (Clark, 2011). All of those processing units working together cohesively at this scale requires a great deal of communication. Furthermore, these processors need to talk not only to each other, but also to any number of other resources like external memories or graphics processors. Being able to move bits around the chip efficiently and quickly therefore becomes one of the limiting factors in the performance of such a system.

To enable this communication, most of today's multi-core systems use interconnection networks. While there are many different ways to design these networks, network latency, the time it takes to communicate between network endpoints, becomes directly dependent on the number of router hops (Daly, 2004). The number of router hops depends upon the total number of endpoint devices as well as the number of ports available on each router—the router's radix. With higher radix routers, we can connect more endpoint devices with fewer total hops. Our project is thus to explore the design space for a high radix router, which will reduce the latency of the interconnect networks and thus enable more efficient communication. Given an initial design based on the work of Stanford graduate student Daniel Becker, we will be exploring how changing different parameters affects the performance of the overall router design in terms of chip area, power consumed, data transmission rates, and transmission delays. We hope to use this

data to draw conclusions about the optimal configurations for a high-radix router, and to justify our conclusions with data. The researchers at Berkeley Wireless Research Center (BWRC) will consider the results of our analysis as they try to construct future high performance systems.

Works Cited

Becker, Daniel. “Efficient microarchitecture for network-on-chip routers”. Doctoral dissertation submitted to Stanford University. August 2012. http://purl.stanford.edu/wr368td5072

Daly, William, and Brian Towles. *Principles and Practices of Interconnection Networks.* San Francisco: Morgan Kaufmann Publishers, 2004.

Clark, Don. “Startup has big plans for tiny chip technology”. *Wall Street Journal*. 3 May 2011. Accessed 5 April 2015

Industry and Market Trends

I. INTRODUCTION

With current trends in cloud computing, big data analytics, and the Internet of Things, the need for distributed computation is growing rapidly. One promising solution that modern computers employ is the use of large routers or switches to move data between multiple cores and memories. The goal of our Petabit Switch Fabric capstone project is to explore the design tradeoffs of such network switch architectures in order to scale this mode of communication to much larger magnitudes. We aim to examine the viability of using these designs for a petabit interconnect between large clusters of separate microprocessors and memories. High bandwidth switches will allow distributed multicore computing to scale in the future. Given a prototype, we will be studying power, area, and bandwidth tradeoffs. By analyzing the performances of these parameters, we will eventually map a Pareto optimal curve of the design space. The results of the project will provide valuable data for future research related to developing network switch designs. As we consider how to commercialize this project, it becomes useful to understand the market that we will be entering. In this paper, we will use Porter's Five Forces as a framework to determine our market strategy (Porter, 1979).

II. TRENDS

First, we will explore some of the trends in the semiconductor and computing industries that motivate our project. One of the most important trends in technology is the shift toward cloud computing in both the consumer and enterprise markets. On the enterprise side, we are observing an increasing number of companies opting to rent computing and storage resources from companies such as Amazon AWS or Google Compute Engine, instead of purchasing and

managing their own servers (Economist, 2009). The benefits of this are multi-fold. Customers gain increased flexibility because they can easily scale the amount of computing resources they require based on varying workloads. These companies also benefit from decreased costs because they can leverage Amazon's or Google's expertise in maintaining a high degree of reliability. We are seeing that these benefits make outsourcing computing needs not only standard practice for startups, but also an attractive option for large, established companies because the benefits often outweigh the switching costs.

As warehouse scale computing consolidates into a few major players, the economic incentive for these companies to build their own specialized servers increases. Rather than purchasing from traditional server manufactures such as IBM or Hewlett-Packard, companies like Google or Facebook are now operating at a scale where it is advantageous for them to design their own servers (Economist, 2013). Custom built hardware and servers allow them to optimize systems for their particular workloads. In conjunction with the outsourcing and consolidation of computing resources, these internet giants could potentially become the primary producers of server hardware, and thus become one of our most important target customers as we bring our switch to market.

On the consumer side, we have seen a rapid rise in internet data traffic in recent years. Smartphones and increasing data speeds allow people to consume more data than ever. Based on market research in the UK, fifty percent of mobile device users access cloud services on a weekly basis (Hulkower, 2012). The number of mobile internet connections is also growing at an annual rate of 36.8% (Kahn, 2014:7). Data usage is growing exponentially as an increasing number of users consumes increasing amounts of data. Moreover, the Internet of Things (IoT) is

expected to produce massive new amounts of traffic as data is collected from sensors embedded in everyday objects. This growth in both data production and consumption will drive a strong demand for more robust networking infrastructure to deliver this data quickly and reliably. This will present a rapidly growing market opportunity in the next decade (Hoover's, 2015). Overall, the general trends in the market suggest a great opportunity for commercializing our product.

As the IoT, mobile internet, and cloud computing trends progress, they will all drive greater demand for more efficient data centers and the networking infrastructure to support further growth. Concurrently, the pace of advances in semiconductor fabrication technology has historically driven rapid performance and cost improvements every year. However, these gains have already slowed down significantly in recent years, and are expected to further stagnate over the next decade. We are rapidly approaching the physical limits of current semiconductor technology. As a result, we observe a large shift from single core computing to parallel systems with many distributed processing units. With no new semiconductor technology on the immediate horizon, these trends should continue for the foreseeable future.

III. INDUSTRY AND COMPETITIVE LANDSCAPE

Next, we will examine our industry and competitive landscape. The semiconductor industry is comprised of companies that manufacture integrated circuits for electronic devices such as computers and mobile phones. This is a very large industry, consisting of technology giants such as Intel and Samsung, with an annual revenue of eighty billion dollars in the United States alone (Ulama, 2014:19). Globally, the industry revenue growth was a relatively modest 4.8% in 2013 (Forbes, 2014). However, as cloud computing becomes more prevalent, we expect

that the need for better hardware for data centers will continue to rise, and the growth of this sector will likely outpace the overall growth of the semiconductor industry.

Although the sector is growing rapidly and the demand for networking infrastructure is high, competition is fierce in both telecommunications and warehouse scale computing. There are many well established networking device companies such as Juniper Networks, Cisco, and Hewlett-Packard. Large semiconductor companies such as Broadcom and Mellanox, along with smaller startups such as Arteris and Sonics, are also designing integrated switches and network on chips (NoC).

Specifically, one of our most direct competitors is Broadcom. In September of 2014, Broadcom announced the StrataXGS Tomahawk™ Series (Broadcom, 2014). This product line is targeted towards Ethernet switches for cloud-scale networks. It promises to deliver 3.2 terabit-per-second bandwidths. This new chip will allow data centers to vastly improve data transfer rates while maintaining the same chip footprint (Broadcom, 2014). It is designed to be a direct replacement for current top-of-rack as well as end-of-row network switches. This means that the switching costs are extremely low, and it will be very easy for customers to upgrade their existing hardware. Another key feature that Broadcom is offering is packaged software that will give operators the ability to control their networks for varying workloads (Broadcom, 2014). The Software Defined Network (SDN) is proprietary software customized for the Tomahawk family of devices. This software might be a key feature that differentiates Broadcom's product from other competitors.

We distinguish ourselves from these companies by targeting a very focused niche market. For example, Sonics has found its niche in developing a network on chip targeted towards the

mobile market. Their product specializes in connecting different components such as cameras, touch screens, and other sensors to the processor. We find our niche in fulfilling a need for a high speed high radix switch in the warehouse scale computing market. Data centers of the future will be more power hungry and will operate at much faster rates (Hulkower, 2012). Therefore, our product aims to build more robust systems by minimizing power consumption while maximizing performance.

The semiconductor industry already competes heavily on the basis of price, and as performance gains level off, we expect this competition to increase (Ulama, 2015, p. 27). As a new entrant, we want to avoid competing on price with a distinguished product. As previously mentioned, our switch product is meant to enable efficient communication between collections of processors in data centers. However, it also has potential applications in networking infrastructure. Given the strong price competition within the industry, we would want to focus on one or the other in order to bring a differentiated product to market.

Another force to consider is the threat of substitutes, and we will now examine two distinct potential substitutes: Apache Hadoop and quantum computing. Apache Hadoop is an open source software framework developed by the Apache Software Foundation. This framework is a tool used to process big data. Hadoop works by breaking a larger problem down into smaller blocks and distributing the computation amongst a large number of nodes. This allows very large computations to be completed more quickly by splitting the work amongst many processors. The product's success is evidenced by its widespread adoption in the current market. Almost every major company that deals with big data, including Google, Amazon, and Facebook, uses the Hadoop framework.

Hadoop, however, comes with a number of problems. Hadoop is a software solution that shifts the complexity of doing parallel computations from hardware to software. In order to use this framework, users must develop custom code and write their programs in such a way that Hadoop understands how to interpret them. A high throughput and low latency switch will eliminate this extra overhead because it is purely a hardware solution. The complexity of having multiple processors and distributed computing will be hidden and abstracted away from the end user. Hadoop is a software solution, so you still need physical switch hardware to use Hadoop, but future improvements to Hadoop or similar frameworks could potentially mitigate the need for the type of high-radix switch which we are building.

The other substitute we will look at is quantum computing. Quantum computing is a potential competing technology because it provides a different solution for obtaining better computing performance. In theory, quantum computers are fundamentally different in the way that they compute and store information, so they will not need to rely as heavily on communication compared to conventional processors. However, it is unclear whether practical implementations of quantum computers will ever be able to reach this ideal. Currently, only one company - D-Wave - has shown promising results in multiple trials, but, their claims are disputed by many scientists (Deangelis, 2014). Additionally, we expect our solution to be much more compatible with existing software and programming paradigms compared to quantum computers, which are hypothesized to be very good for running only certain classes of applications. Therefore, switching costs are expected to be much higher with quantum computers. Because quantum computing is such a potentially disruptive technology, it is important to consider and be aware of advancements in this field.

IV. MARKET

Next, we will examine two different methods of commercializing our product: selling our design as intellectual property (IP), or selling a standalone chip. Many hardware designs are written in a hardware description language such as Verilog. This code describes circuits as logical functions. Using VLSI (Very Large Scale Integration) and EDA (Electronic Design Automation) tools, a Verilog design can be converted into standard cells and manufactured into a silicon chip by foundries. If we were to license our IP, a customer would be able to purchase our switch and integrate it into the Verilog code of their own design.

Some key customers for licensing our IP are microprocessor producers. The big players in this space are Intel, AMD, NVIDIA, and ARM. Intel owns the largest share of microprocessor manufacturing, and it possesses a total market share of 18% in semiconductor manufacturing (Ulama, 2014:30). Microprocessors represent 76% of Intel's total revenue, making it the largest potential customer in the microprocessor space (Ulama, 2014:30). AMD owns 1.4% of the total market share, making it a weaker buyer (Ulama, 2014:31). While Intel represents a very strong force as a buyer because of its power and size, they are still an attractive customer. If our IP is integrated into their design, we will have a significant share in the market.

Another potential market is EDA companies themselves. We can license our product to EDA companies who can include our IP as a part of their libraries. This can potentially create a very strong distribution channel because all chip producers use these EDA tools to design and manufacture their products. Currently, EDA is a $2.1 billion industry, with Synopsys (34.7%) and Cadence (18.3%) representing 53% of the total market share (Boyland, 2014:20). Having our

switch in one of these EDA libraries would result in immediate recognition of our product by a large percentage of the market.

Another option for going to market would be selling a standalone product. This means that we will design a chip, send our design to foundries to manufacture it, and finally sell it to companies who will then integrate the chip into their products. This contrasts with licensing our design to other semiconductor companies. Licensing our design would allow our customers to directly embed our IP into their own chips. One downside of manufacturing our own chip is the high cost. Barriers to entry in this industry are high and increasing, due to the high cost of production facilities and low negotiation powers of smaller companies (Ulama, 2014:28). Selling a standalone chip versus licensing an IP also targets two very different customers—companies who buy parts and integrate them, or companies who manufacturer and sell integrated circuits.

The main application of our product is in warehouse scale computing. The growth in cloud computing and media delivered over the internet means that demand for servers will see considerable growth (Ulama, 2014:8). High-speed high-radix switches will be essential in the future for distributed computing to scale (Binkert, 2012:100). In a data center, thousands of servers work together to perform computations and move data. Our product can be integrated in network routers connecting these servers together. Companies such as Cisco and Juniper, who supply networking routers, are our potential buyers. They purchase chips and use them to build systems that are sold to data centers. Our product can also be integrated directly inside the servers themselves. Major companies producing these servers include Oracle, Dell, and Hewlett-Packard. These companies design and sell custom servers to meet the needs of data

centers. As the number of processing units and memories increase in each of these servers, a high-radix switch is needed to allow efficient communication between all of these subsystems.

In order to enter the market strategically, we need to consider our positioning. The market share of the four largest players in the networking equipment industry—our target customers—has fallen by 5.2% over the past five years (Kahn, 2014:20). The competition is steadily increasing, and the barriers to entry are currently high but decreasing (Kahn, 2014:22). With the influx of specialist companies offering integrated circuits, new companies can take advantage of this breakdown in vertical integration (Kahn, 2014:22). This means that the industry may expect to see a rise in new competitors in the near future. With the increase in competition among the buyers, their power is expected to decrease. Thus, if we have a desirable technology, we may be in a strong position to make sales. Competition in server manufacturing is also high and increasing with low barriers of entry (Ulama, 2014:22). This competitive field in both networking equipment and data center servers is advantageous for us because these companies are all looking for any competitive edge to outperform each other. A technology that will give one of these companies an advantage would be very valuable.

In order to create a chip, we will need to pay a foundry to manufacture our product. Unfortunately, although there is healthy competition among the top companies in the semiconductor manufacturing industry, prices have remained relatively stable because of high manufacturing costs and low margins (Ulama, 2014:24). Because custom and unique tools are required for producing every chip, there are very high fixed costs associated with manufacturing a design. Unless we need to produce very large volumes of our product, the power of the foundries, our suppliers, is very strong. The barriers of entry for this industry are extremely high,

and we don't expect to see much new competition soon. EDA tools developed by companies such as Synopsys and Cadence are also required to create and develop our product. As discussed in previous sections, these two companies represent more than half of the market share. As a result, small startups have weak negotiation power. Both our suppliers, foundries who manufacture chips and EDA companies that provide tools to design chips, possess very strong power largely in the form of fixed costs.

V. CONCLUSION

In this paper, we have thoroughly examined a set of relevant trends in the market and, using Porter's Five Forces as a framework, conducted an analysis of the semiconductor industry and our target market. We have concluded that our project will provide a solution for a very important problem, and is well positioned to capitalize on projected industry trends in the near future. We have proposed and analyzed two different market approaches - IP licensing and selling discrete chips - and weighed the pros and cons of each. We have surveyed the competitive landscape by looking at industry behaviors and researching a few key competitors, as well as thinking about potential substitutes. With all of this in mind, we can carefully tailor our market approach in a way that leverages our understanding of the bigger picture surrounding our technology.

Works Cited

Binkert, N.; Davis, A.; Jouppi, N.; McLaren, M.; Muralimanohar, N.; Schreiber, R.; Ahn, Jung-Ho, "Optical high radix switch design," Micro, IEEE , vol.32, no.3, pp.100,109, May-June 2012.

Boyland, Kevin. 2013 IBISWorld Industry Report OD4540: Electronic Design Automation Software Developers in the US. http://www.ibis.com, accessed February 13, 2015

Broadcom. Broadcom Delivers Industry's First High-Density 25/100 Gigabit Ethernet Switch for Cloud-Scale Networks. Press Release. N.p., 24 Sept. 2014. Web. 1 Mar. 2015. <http://www.broadcom.com/press/release.php?id=s872349>.

“Computing: Battle of the Clouds”. The Economist. 15 October 2009. http://www.economist.com/node/14644393?zid=291&ah=906e69ad01d2ee51960100b7fa502595

Deangelis, Stephen F. “Closing in on Quantum Computing”. Wired. 16 October 2014. http://www.wired.com/2014/10/quantum-computing-close/

Handy, Jim. 2014 Semiconductors A Crazy Industry. http://www.forbes.com/sites/jimhandy/2014/02/11/semiconductorsacrazyindustry2/, accessed February 16, 2015

Hoover’s. “Semiconductor and other electronic component manufacturing: Sales Quick Report”. http://subscriber.hoovers.com/H/industry360/printReport.html?industryId=1859&reportType=sales, accessed February 11, 2015

Hulkower Billy, “ Consumer Cloud Computing- Issues in the market”, Mintel (2012)

Kahn, Sarah. 2014 IBISWorld Industry Report 33421: Telecommunication Networking Equipment Manufacturing in the US. http://www.ibis.com, accessed February 10, 2015

Kahn, Sarah. 2014 IBISWorld Industry Report 51721: Wireless Telecommunications Carriers in the US. http://www.ibis.com, accessed February 26, 2015

Porter, Michael. “The Five Competitive Forces That Shape Strategy”. Harvard Business Review. January 2008.

“The Server Market: Shifting Sands”. The Economist. 1 June 2013. http://www.economist.com/news/business/21578678-upheaval-less-visible-end-computer-industry-shifting-sands

Ulama, Darryle. 2014 IBISWorld Industry Report 33441a: Semiconductor & Circuit Manufacturing in the US. http://www.ibis.com, accessed February 10, 2015

Ulama, Darryle. 2014 IBISWorld Industry Report 33329a: Semiconductor Machinery Manufacturing in the US. http://www.ibis.com, accessed February 10, 2015

Ulama, Darryle. 2014 IBISWorld Industry Report 33411a: Computer Manufacturing in the US. http://www.ibis.com, accessed February 10, 2015

IP Strategy

Distributed computing is rapidly growing due to demand for high performance computation. Today, computers have multiple cores to divide and solve complex computational problems. In the near future, they will have many more cores which will need to work in unison. In this project, we are designing a high-radix router which will serve as an interconnect between processor cores and memory arrays in data centers. Our project addresses the problem of transferring large amounts of data between processors and memories to achieve high speed computation. It is a part of ongoing research in Berkeley Wireless Research Center (BWRC) for building hardware for next generation data centers.

The router we are designing is unique among other routers available today in several ways. First, it is a high-radix router which means it can be used to direct traffic to and from a large number of endpoints. Second, the router can support very high bandwidth. We have designed such a high-performing router by proposing a novel system architecture based on a few key design decisions from the results of our design space exploration. These design decisions differentiate our router from existing designs in the commercial and research domains, and would form the core of our patent application.

If we are successful in implementing our proposed design changes, then the router design can qualify for a patent. We would apply for a utility patent since the router will produce a useful tangible result like increased bandwidth. One of our marketing strategies is to sell the router as a standalone chip, which means we will be mass producing the router from a chip foundry. This makes it an article of manufacture, another quality of a utility patent. In addition to qualifying for one of the patent categories, our router can be considered novel invention since it is a high radix

router with up to 256 ports. This is much higher than any others that we have come across during our literature review.

Patenting our novel design will give us a huge competitive advantage because we would be the first to develop a petabit bandwidth router. In general, the semiconductor industry is highly litigiousness because of rapid change in the technology each year. Many lawsuits are filed every year between rivals like Broadcom, Qualcomm, and Samsung. Furthermore, many of these companies have very deep pockets, along the motivation and resources to rigorously protect their patent portfolio. Therefore, before commercializing our technology, we must to exercise careful scrutiny to ensure we do not infringe on anyone else's patents. In this environment, it also becomes necessary for us to hold our own patents, both to keep others from copying our technology and to prevent them from coming after us with lawsuits. However, as a small startup, we would have to weigh any sort of legal action very carefully, as we would likely not have sufficient funding to carry out protracted legal battles.

The primary risk of choosing not to patent our novel router architecture would be forfeiting the legal protections that a patent grants. As a small company starting out, we would not provide much value as to our customers beyond our technological advantage. Without a patent, we risk allowing a much larger company to copy our technology. Combined with their vast resources, this could effectively put us out of business. While we might not actually be able to defend our patent, having one would at least deter others from blatantly copying us.

Something else to consider here would be how easy we think it would be for our technology to be reverse engineered. Since our project is conducted in a research setting under BWRC, any major breakthroughs would most likely be published and peer reviewed, rather than

kept as a trade secret. Furthermore, since our technology would be based on a novel architecture rather than an implementation detail, others would almost certainly be able to engineer their own solutions based on our architecture, depending on how much we decide to publish. Thus, without a patent, we would have no way of controlling or profiting from our technology.

A potential secondary risk of not patenting might be that we would be passing on the chance to attract potential investors. In addition to the legal protection described above, holding a patent could have the additional effect of demonstrating strength to investors in multiple ways. First, the patent would differentiate us from our competitors; it gives us a sustainable, legally enforceable competitive advantage. Second, the patent would signal a high level of expertise to investors; it can signal that we are truly experts in our particular domain. Finally, the patent could provide assurances to investors that other companies will not be able to patent something similar and attempt to come after us for infringement.

With all of this in mind, we would most definitely want to obtain a patent for our novel technology. Practically, the extent of legal protection we might receive remains questionable given our limited financial resources, but a patent still grants us many other advantages which could provide a huge boost to a company in its early stages. From this preliminary analysis, the benefits far outweigh to costs, and we would thus want to pursue a patent as soon as possible. We will conduct a thorough patent search with assistance from a patent attorney to make sure our invention has not previously been patented and does not infringe on any existing patents.

Individual Paper

Technical Contributions

I. OVERVIEW

Given the current trends in computing and the shift toward multi-core and multi-chip systems, the capacity for moving bits back and forth between discrete processing units has become a limiting factor for system performance. An efficient communication network is therefore key to continued improvement. Professor Krste Asanovic of the University of California, Berkeley recently laid out the Berkeley vision for the datacenter of the future. In his keynote address at the Usenix FAST 2014 conference, he described large clusters of CPUs and memories connected through optical fibers and high-radix switches and routers (Asanovic, 2014). High-radix switches and routers become increasingly attractive as they can dramatically decrease the number of router hops in the network, and therefore significantly cut down on the network latency (Dorren et al., 2015: 1117) In this context, our project is to characterize the design space for a high radix router. This is part of ongoing research at the Berkeley Wireless Research Center (BWRC), and our results will be used by other researchers to make decisions as they design future systems.

In order to accomplish this, we have collectively identified and divided the project into a series of major subtasks. These include literature study, code study, tool setup, initial data collection, tool optimization, and further architectural optimizations. The literature study was highly segmented amongst group members, with each person choosing a particular sub-block in the overall design to study in depth; as such, the “Knowledge Domains” section below will reflect this. For the rest of the project, our team tackled many of the major tasks together, so much of the content in our papers will overlap, especially in the “Methods and Materials”

sections. For these tasks, I will focus on my technical contributions towards each goal, while also providing some context within the overall project objective. In the "Results and Discussion" sections, each team member will discuss the results corresponding to the parts which he or she contributed most heavily toward. Each individual paper should provide a unique perspective on the individual contributions of the various team members, and taken together, they should provide a more in depth understanding of our project goals and outcomes.

As both the project manager and a strong technical contributor, my individual work has been key to moving the project along. In addition to facilitating discussions during our weekly meetings, and assigning tasks to other group members, I have also made several key strides in both the setup and optimization of the software tool infrastructure needed for our project, along with key contributions toward integrating the SRAMs as an optimization. Additionally, each week I have engaged team members in discussions about issues encountered with their subtasks, and together we have come up with solutions to move forward.

II. KNOWLEDGE DOMAINS

Existing research in building high-radix switches include a multitude of novel architectures proposed by researchers around the world. One example is the Hierarchical Asymmetric Crossbar (HAC) proposed by Fang and Wang. This scheme makes use of asymmetric crossbars with more output ports than input ports followed by a level of output multiplexers to achieve the functionality of a symmetric crossbar. Their work shows promising results, but their evaluation was limited to a switch of radix 32, which is still well below the sizes we are interested in (Fang and Wang, 2011). Another interesting architecture proposed by Stapathy et al. makes use of a transistor-circuit level matrix of input and output busses. They

claim to have fabricated a radix 64 switch fabric with throughput of 4.5 Tb/s, which is in the range of throughput we are looking to achieve (Stapathy et al., 2012). One major difference here is that their architecture is based on a matrix of shared input and output busses rather than fully connected crossbars like our design. This can potentially lend itself to a different set of applications, as bus-based architectures would only perform well when endpoints only rarely need to communicate, such that there will not be frequent resource contention for the bus. Others have proposed even more exotic schemes based on optical crossbars (Binkert et al., 2012).

As noted above, we divided the router architecture into several major blocks which each team member could study in depth, and each team member's "Knowledge Domains" section will focus specifically on a different part of the design. Yale Chen will discuss arbiters, Jay Mistry will talk about allocators, Bhavana Chaurasia will focus on switch allocators, Surabhi Kumar will talk about buffer management, and I will discuss virtual channel allocation. I will discuss the different parameters we potentially have control over, and especially what kinds of tradeoffs we can expect when modifying those parameters. The objective of this collective overview is to provide a general understanding of how a router works, as well as what kinds of improvements can be made with what tradeoffs. This will help to motivate the directions which we have chosen to explore with the rest of the project.

In order to discuss virtual channel allocation, we must first discuss flow control and virtual channels. The flow control mechanism determines how routers can talk to each other by dictating when a packet or flit (part of a packet) can be forwarded on to the next router. In particular, it determines how buffers and channel bandwidth are granted to packets, as well as how resource conflicts are resolved (Becker, 2012). In other words, when multiple packets need

to use a single channel, the flow control mechanism will determine how the winner gets chosen. There are many different types of flow control schemes, but virtual channel flow control is the most popular method used in router designs.

A virtual channel (VC) holds all of the state needed to coordinate the handling of flits over a channel. One of the key properties of a virtual channel is that it decouples buffer allocation from channel allocation (Daly, 1992:194). Virtual channel flow control is a particular type of flow control, which makes use of virtual channels to eliminate some of the problems of more naive schemes. Rather than only allowing a single packet to occupy a physical channel while it forwards all of its flits, having multiple virtual channels allows you to effectively multiplex the physical channel (Daly and Towles, 2004:239).

In many cases, multiple packets will want to forward their flits to the same destination port; when this happens, if a single packet is occupying the physical channel, it must wait until the destination is ready to receive; meanwhile, the physical channel lies idle. This is known as head-of-line (HOL) blocking, and is one of the primary issues that virtual channel flow control solves, since another VC can make use of the physical channel to forward to a different destination while the blocked one waits. Additionally, virtual channel flow control helps to avoid deadlock and also gives a convenient mechanism to implement multiple Quality-of-Service (QoS) classes (Becker 2012).

The VC allocator is responsible for matching an incoming request from an input virtual channel with an available output virtual channel. The primary tradeoff here comes in choosing the range of potential output VC's. A larger set of potential output VC's could mean a more efficient use of resources since a VC would be less likely to remain idle; however, the size and

complexity of the allocator grows exponentially with the number of things it has to allocate amongst, so it becomes bigger and slower. As with most performance metrics, this will be highly workload dependent. Shim et al. argue that in some applications, statically allocating VC's can greatly simplify the allocation, while also providing most of the benefits. They do acknowledge, however, that this may lead to suboptimal performance since the allocation scheme cannot account for dynamic behavior (Shim et al., 2009). The approach that Daniel Becker recommends, and which we have chosen to follow, combines this thinking with more traditional dynamic allocation: he assigns a set of VC's to belong to each different type of packet, while still performing the allocation dynamically amongst that set (Becker, 2012). When the VC allocator needs to select an output VC for a request from an input packet, it only needs to select from among the available VC's for that packet type.

Another choice we can make with regard to VC allocators is which type of allocator structure we want to use. There are three primary types of allocators we can employ: separable input first, separable output first, and wavefront. For a more in-depth description on these structures, refer to Jay's paper (Mistry, 2015). In general, there are tradeoffs associated with each of the three choices. There are also various schemes for performing the virtual channel allocation speculatively, or combining the VC allocation with other allocation steps (Nguyen and Oyanagi, 2010). However, specifically for use in VC allocation, Becker finds that the differences in matching quality are very small under normal usage. Thus, we should instead focus on other metrics like speed and area. Based on his analysis, in both of these metrics separable input first is the clear winner. We have therefore concluded from this literature review that we should focus our efforts on other blocks.

III. METHODS AND MATERIALS

The design space exploration we conducted consisted of taking various design points - collections of various parameters - and pushing them through the ASIC tool flow to get a fully placed and routed design that could be sent off to a foundry and fabricated as a real chip. For the purposes of this project, we did not send our designs off to be fabricated. Rather, once we obtained this final output, we then collected statistics such as area, power consumption, and critical path timing. Given enough of these statistics, we can then examine the types of tradeoffs available for different parameter combinations. Then, we can draw conclusions about which tradeoffs are most favorable given our desired application of using the high radix router to connect large clusters of processors and memories.

There are several essential tools which make this sort of design space exploration possible. First, we make extensive use of the Verilog hardware description language. This enables us to represent digital circuits using code rather than circuit diagrams, and use software tools to transform our code into an actual physical layout implementation. Verilog is an industry standard language used by most companies making real chips. The Synopsys software tools we use to transform the code into circuits are also common in industry. Through the resources provided by BWRC, we have access to the most recent Synopsys tools for synthesis and place-and-route, as well as the simulation tools used to verify our designs.

While we could have written our own implementation of a router from scratch, which might have been a good learning experience, given the time constraints of this project, it simply would have taken too long to get a design working. Instead, we used Daniel Becker's open source, parameterized router code as the starting point for our project. We then tweaked and

augmented the existing code to suit our purposes. One of the challenges with this approach was that his code was written for a router in the context of a network-on-chip design, while our investigation has primarily focused on a single router. However, the benefits of using his design still far outweigh the time investment we would have had to otherwise make in order to create our own.

During the first semester, I took the lead in setting up the tool flow. Using the EE141 class project as a reference, I adapted the infrastructure to work with Becker's code so that we could run the design through the flow and collect the necessary statistics to map out the initial design space. The class project code was a very good starting point, but it still required a few key tweaks in order to work with Becker's code. While the aggregate total of all these tweaks was not actually a lot of code, finding out what exactly needed modification was a very tedious task of looking through the Synopsys user guides and asking other grad students for help. I also took the lead with the setup and debugging of the RTL, post-synthesis, and post-PAR simulation frameworks. All of this was extremely important because it laid the groundwork for all of our future work during the second semester. Without a working software tool setup, we would not have been able to move on with the project.

Once we had the tools working, we began to scale up Becker's design to more higher radix configurations. Since his thesis focused on routers for use in network-on-chip systems, his design was intended to produce low radix routers of around 16 ports or less. Due to our target throughput of several terabits per second, we needed to try to make much higher radix designs with at least 64 ports. One of the big challenges that arose next was dealing with the long runtimes of synthesizing a design with so many ports. As we increase the number of ports, the

complexity of the router scales exponentially, so in stepping up from 16 ports to 64 ports, the tool runtime increased from a few hours to a few days. This was unacceptable for us, since we wanted to be able to try running many different designs with many different parameter combinations. Ultimately, this made scaling up to 128 ports completely impractical given the time constraints.

In order to reduce the runtimes, there were a couple of approaches that we can took. First, we modified the tool setup scripts in order to try to get the runtime down. In particular, the reference flow from the class project had been configured with many optimization steps because that design was very small, and so the tools could complete the many levels of optimization in a reasonable amount of time. In general, the synthesis and place-and-route software can trade off runtime for quality. In other words, it should be able to perform more optimization steps to yield a better design, but those extra steps will take longer to complete. For this approach, I spent a lot of time going through user guides to figure out how to prune the optimization process down to the bare minimum, while also enabling multithreaded synthesis and place-and-route for extra speed. Jay also contributed in this regard, and his paper will discuss his steps in more detail (Mistry, 2015). Unfortunately, the combination of these steps was not enough to get the runtime into an acceptable range, so we had to try other methods in order to get lower runtimes.

The next solution we came up with to decrease tool runtime was to separate out the crossbar's place-and-route step from the rest of the design. We realized that the layout and functionality of the crossbar depend only upon the number of ports. Rather than re-doing the whole place-and-route every time, we should be able to just make a single layout for a crossbar with a set radix and then reuse that layout for all variants of the router with the same number of

ports. Since only the logic around the crossbar should be changing given our other parameter tweaks and additional logic, it made sense to set the layout of the crossbar to reduce the amount of necessary work for the tool. Yale was responsible for implementing this, and will discuss the steps he performed with more detail in his paper (Chen, 2015).

In parallel with the runtime optimization efforts, we were able to gather data for a few 64-port configurations. With this data, we focused our optimization primarily on ways to reduce area and decrease the critical path. By looking at the timing reports, we deduced which modules were causing the critical path and attempted to shorten those paths by making changes at the RTL level. Bhavana focused on ways to improve critical path and will discuss more in her paper (Chaurasia, 2015).

We also looked at the area reports and determined that the input buffers were consuming a large portion of the total area. We therefore tried to cut down on the chip area by replacing the flip-flop arrays with SRAMs generated from the CACTI SRAM generator tool. For the SRAM integration, I worked closely with Surabhi, who was responsible for figuring out how to properly use CACTI, as well as modifying the RTL to replace the FF arrays with the SRAM macros. Please refer to her paper for further discussion of these topics (Kumar, 2015).

The main technical implementation challenges in implementing these changes were caused by the fact that the SRAMs are synchronous read, while the flip-flop arrays are asynchronous read. Simply replacing the flip-flops with SRAMs thus yielded a non-working router. I was responsible for finding a solution to this problem. I added extra pipeline registers and refactored some of the existing update logic to maintain functional correctness. I then implemented and debugged these RTL changes and verified their correctness in RTL,

post-synthesis, and post-PAR simulation. This required many hours of waveform debugging. Once I had verified functional correctness, we then synthesized a couple of 64 port designs and collected data. The results of these runs will be discussed in the following section.

IV. RESULTS AND DISCUSSION

A major accomplishment for the group was successfully configuring a tool flow for pushing the design all the way through place and route. While this was not the original goal for our project, it was an extremely important step towards the larger objective, and much more difficult than initially expected. Through this process, we developed a much deeper understanding of how the Synopsys tools work, how to troubleshoot many of the errors, and how to manipulate the software at a much more detailed level than anything we have learned in class. We have implemented a tool flow that runs significantly faster than the original setup, along with a minimally functioning hierarchical flow that demonstrates the approach we would need to take for getting further speedup. This will enable faster iterations for trying new designs, getting results, and determining new directions to explore. Having this flow in place will allow future researchers to more effectively explore this design space beyond what we have been able to do with the limited time constraints.

From our initial investigations, we learned that approximately a third of the area from the 64 port configurations was due to the area occupied by just the input buffers. From this data, we decided that there is a lot of potential for improvement in this area, and reduction of the size of the input buffers would be very beneficial to overall area. As previously discussed, we made necessary changes at the RTL level to replace the flip-flop arrays used for the input buffers with SRAMs. The results of this study are shown below in Tables 1 and 2.

	area (um^2)	clk period (ns)	power (uW)
FF regfile	6.86E+06	11.02	7.89E+05
SRAM regfile	5.89E+06	10.72	7.03E+05

Table 1: Comparison of 64 port, 32 flit width router, post-PAR

	area (um^2)	clk period (ns)	power (uW)
FF regfile	1.14E+07	11.63	1.21E+06
SRAM regfile	9.93E+06	11.95	1.09E+06

Table 2: Comparison of 64 port, 64 flit width router, post-PAR

The data clearly shows that, as expected, the addition of SRAMs decreases both area and power consumption. Interestingly, though, we also notice that for the 32 flit width router, the SRAM decreases the cycle time, while increasing cycle time for the 64 flit width router. One likely explanation of this is simply that the cycle time has less to do with the actual access time of the SRAM vs FF array, and more to do with the relative optimality of the placement and subsequent routing delays for any given place-and-route run. In both cases, the difference between the cycle times for the two designs is on the order of a few tenths of nanoseconds, which is well within the variability of the ICC place-and-route tool. We would need to collect more data points, possibly just re-running the same configurations multiple times, to get an idea for the variance in critical path inherent in the tool.

One particularly interesting trend we can see from the data is that in terms of both power and area, the SRAMs improve the design significantly for the 32 flit width design, but show much less favorable improvement for the 64 flit width design. This could be because the logic surrounding the SRAM requires more registers to deal with the synchronous read behavior of the SRAM. These additional registers certainly would reduce the amount of savings from replacing the FF arrays with SRAMs. If this is the case, then it suggests that the SRAMs provide

diminishing returns at larger sizes because of the additional overhead required to make them work within the router. Future work could further verify this hypothesis with more data points.

Another factor which might contribute to the diminishing returns could be the fact that we used relatively small SRAMs of size 16x32, where each instantiation carries the overhead of the decoder and output logic. We could also use much larger SRAMs where the overhead gets amortized over more bits of actual storage. It would be an interesting area of future inquiry to explore using fewer, larger SRAM block sizes to further decrease area, most likely at a cost of longer access times. A caveat to keep in mind with all of these results is that the SRAMs we used were generated from CACTI and should only be considered a reasonable first-order approximation; future improvements on this study would include using a real-world SRAM compiler.

While these results do not cover the range of design space exploration we had initially hoped for, we have still collected meaningful data which can serve as a solid foundation for future work. The tool flow we set up will allow future design space exploration to iterate much more quickly, and the data we have collected provides insights which can guide further exploration. Given more time, we would like to continue to broaden our search space and gain a more thorough understanding of the tradeoffs in this design space.

Works Cited

Asanovic, Krste. “FireBox: A Hardware Building Block for 2020 Warehouse-Scale Computers”. *Usenix FAST 2014: 12th Usenix Conference on File and Storage Technologies*. Santa Clara, CA. Keynote Address. 17 February 2014.

Becker, Daniel. “Efficient microarchitecture for network-on-chip routers”. Doctoral dissertation submitted to Stanford University. August 2012. http://purl.stanford.edu/wr368td5072

Binkert, Norman, and Al Davis, Norman P. Jouppi, Moray McLaren, Naveen Muralimanohar, Robert Schreiber, and Jung Ho Ahn. “Optical High-Radix Switch Design”. *IEEE Micro, Vol. 32 No. 3*: 100 - 109. 3 April 2012.

Chaurasia, Bhavana. Contributions to “Petabit Switch-Fabric Design”. May 2015.

Chen, Yale. Contributions to “Petabit Switch-Fabric Design”. May 2015.

Daly, William. “Virtual-Channel Flow Control”. *IEEE Transactions on Parallel and Distributed Systems, Vol. 3 No. 2*: 192 - 405. March 1992.

Daly, William, and Brian Towles. *Principles and Practices of Interconnection Networks.* San Francisco: Morgan Kaufmann Publishers, 2004.

Dorren, H. J. S., and Erik H. M. Wittebol, Rob de Kluijver, Gonzalo Guelbenzu de Villota, Pinxiang Duan, and Oded Raz. “Challenges for Optically Enabled High-Radix Switches for Data Center Networks”. *Journal of Lightwave Technology, Vol. 33 No. 5*: 1117 - 1125. 1 March 2015.

Fang, Ming, and Kefei Wang. “Achieving High Throughput in High-Radix Switch”. *2011 International Joint Conference of IEEE (TrustCon)*: 1452 - 1456. Changsha, China. November 2011.

Kumar, Surabhi. Contributions to “Petabit Switch-Fabric Design”. May 2015.

Mistry, Jay. Contributions to “Petabit Switch-Fabric Design”. May 2015.

Nguyen, Son Truong, and Shigeru Oyanagi. “The Design of On-the-fly Virtual Channel Allocation for Low Cost High Performance On-Chip Routers”. *First International Conference on Networking and Computing, 2010*: 88 - 94. Higashi-Hiroshima, Japan. November 2010.

Satpathy, Sudhir, and Reetuparna Das, Ronald Dreslinski, Trevor Mudge, Dennis Sylvester, and David Blaauw. “High Radix Self-Arbitrating Switch Fabric with Multiple Arbitration Schemes and Quality of Service”. *Design Automation Conference 2012*: 406 - 411 . San Francisco, CA. 3 June 2012.

Shim, Keun Sup, and Myong Hyon Cho, Michel Kinsy, Tina Wen, Mieszko Lis, G. Edward Suh, and Srinivas Devadas. “Static Virtual Channel Allocation in Oblivious Routing”. *3rd ACM/IEEE International Symposium on Networks-on-Chip, 2009*: 38 - 43. San Diego, CA. 10 May 2009.

Concluding Reflections

The results we have collected represent an incomplete set of steps toward a larger goal. With the work we have done, someone should be able to simply clone a copy of our code base and continue experimenting with different parameter combinations to more fully characterize the design space for high-radix routers. As previously mentioned, directions for future work could include collecting more data points to get a better understanding of tradeoffs, using different SRAM sizes as building blocks for our input buffers, and other architectural changes to the router design.

In addition to the technical knowledge, this project has also yielded a number of lessons in project management. While I have had group projects before that dealt with teams of five students, the scale and duration of this capstone project presented many new challenges. Most notably, I found that each team member came from a very different undergraduate background and had varying levels of proficiency with the tools used for the project. At the beginning of the project, we attempted to distribute tasks for the tool setup evenly, but it soon became apparent that similar tasks would take two different team members vastly different amounts of time to complete. The takeaway for me was that we should have spent more time initially getting familiar with each others' skills, and that sometimes it is simply impractical to try to partition work evenly.

Another takeaway from this experience has been that people generally provide poor estimates of the amount of time it will take to get a task finished. This was especially true for many of the tasks for which no member of the group had any prior experience, thus making the initial estimates little more than guesswork; unsurprisingly, these proved to be wildly inaccurate

and caused us to remain behind schedule for the rest of the project. For future projects, I would invest much more planning effort into getting accurate estimates for the expected completion time of individual tasks.

Overall, our final outcomes fell somewhat short our initial goals somewhat as we found the learning curve of using the tools was much higher than expected. However, although we did not complete the design space exploration to the extent we would have liked, we still made tangible progress toward the goal. We gained some truly valuable experience working in a team with diverse backgrounds, and in an environment very different from the curated classroom experience with which most of us are most familiar.