Petabit Switch Fabric Design

Yale Chen
Bhavana Chaurasia
Ian Juch
Surabhi Kumar
Jay Mistry

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-93
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-93.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

Yale Chen

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.

Abstract

Because of physical limitations on how fast we can operate modern digital integrated circuits, trends point towards increasing the number of processing cores by leveraging parallelism. Fast and efficient communication between all of these cores is paramount for optimal performance. In our report, we explore the design space of high radix switches up to 64x64 ports. We also examine alternative methods of building the switch using Synopsys design tools.

Table of Contents

Shared Team Paper

Problem Statement ----------1

Industry and Market Trends ----------3

IP Strategy ----------13

Individual Paper

Technical Contributions ----------16

Concluding Reflections ----------29

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.

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.

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.

Technical Contributions

I. OVERVIEW

The main goal of the Petabit Switch Fabric capstone project was to explore the design space of a high radix router. Using an open sourced router design from Daniel U. Becker's Stanford dissertation, "Efficient Microarchitecture for Network-on-Chip Routers," we explored tradeoffs between power, area, and bandwidth by adjusting parameters and implementing custom design changes. For example, increasing the number of input and output ports might result in higher bandwidth because more bits can travel across the router. However, this would increase our power and area metrics because the design would be larger. Alternatively, we might attempt to decrease power and area by reducing the complexity of control algorithms. As expected, the performance of the router is likely to decrease because of less efficient controllers. Our project aims to study and quantify these tradeoffs.

In order to collect the power, area, and bandwidth data of our chip, we needed to push our design through a set of Synopsys Design Compiler tools. These tools will be discussed in further detail in the Methods and Materials section of this paper. The first task of the project was to set up these tools to work properly for our specific design. This involved setting up our compute environment, configuring the tools, and modifying custom build scripts. Whenever a new parameter was changed or the design was updated, the router needed to be re-compiled and pushed through the tool-flow again. Early in the project, we realized that the large router designs we were planning on implementing were taking far too long for the tools to process. Each design was taking many days to complete. If we were to iterate over a combination of parameters, it would be infeasible to finish collecting the necessary data within the timeframe of the project.

Therefore, we quickly realized that the majority of work would actually be dedicated to improving the turn-around time of the Synopsys tools.

Because of the highly linear progression of tasks that were required before we could actually begin to build our chip and run simulations, our group worked in parallel for a significant portion of the project without clearly separated tasks. This means that there was a significant amount of collaboration that was required because our tasks were overlapping. Therefore, in our set of technical contribution papers, we will each be discussing our individual contributions that helped our group accomplish common goals, rather than discussing isolated tasks.

In the Results and Discussion section, I will be discussing my contributions: initial design space exploration of small routers, hierarchical design implementation, and scripts for data parsing and running tools. First, the initial design space exploration with small designs was important for us to make sure our assumptions based on our literature reviews were correct. It helped us gain intuition for how changing certain parameters would impact our router's power, area, and bandwidth. This set of data points was also an indicator that the tools were behaving correctly. For example, if we increased our number of ports but found a decrease in total area, it would be obvious that we had an error in our method. Next, using hierarchical design was crucial for allowing us to complete builds with fast turn-around times. Hierarchical design allowed us to use unchanged portions of previous builds in our new build. This reduced redundancy in our work, and the tools could complete new designs more efficiently. Finally, Python and Bash scripts were extremely useful for parsing reports and aggregating data automatically. These scripts were a shared resource that everyone in the group could access, and they further improved the ability of the group to collect data quickly.

II. KNOWLEDGE DOMAINS

In order to explore the design space of a router, we must first understand the different parts that make up a router, and how these modules are integrated together. On a high level, a router is a component that moves data from any input port to any output port. This switch allows different parts of a system to communicate with each other. How efficiently and quickly data can be moved from an input port to an output port depends on the design and implementation of the router. In this section, I will be discussing arbiters, while the rest of my group will be focusing on the other submodules of the router.

One of the fundamental operations performed by the control logic in a router is arbitration, or the mediation between multiple agents that want to access a shared resource (Becker, 2012). In other words, arbiters ensure that if one or more agents request access to a particular resource, one of the agents will receive a grant. Arbitration algorithms occur in buffer management as well as switch allocators, which will be discussed in Bhavana Chaurasia's as well as Surabhi Kumar's literature review sections. In the following paragraphs, I will be discussing five different arbitration techniques and each of their tradeoffs.

First, we will consider the fixed priority arbiter. This is the simplest form of arbitration where access to resources is granted to agents based on a predetermined fixed priority order. A simple schematic of this implementation is shown in Figure 1. This straightforward approach uses a linear array of cells, granting g_i if request r_i is asserted. This design minimizes complexity, resulting in minimized power and area. However, this implementation incorporates no fairness because some agents will always receive priority over others.

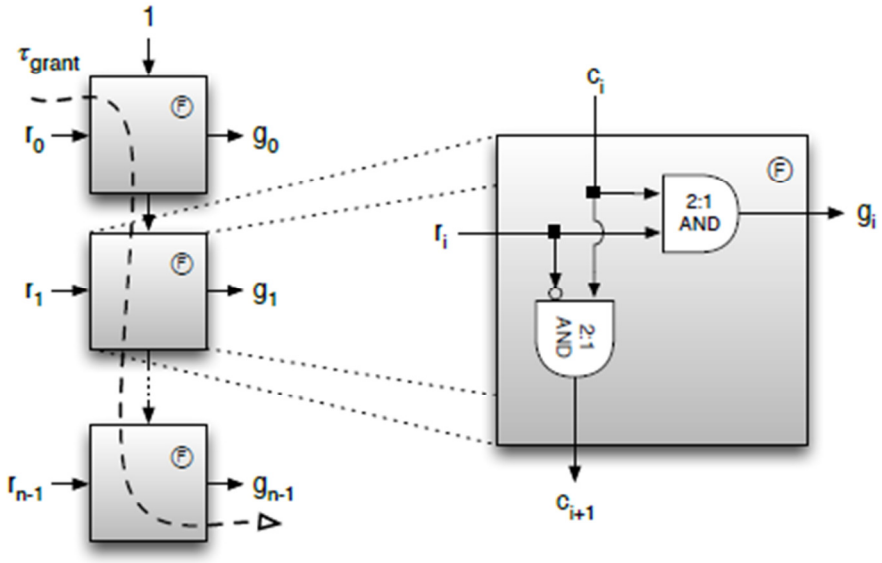

Figure 1: Fixed Priority Arbiter (Becker, 2012)

Another approach is using round-robin arbiters. This implementation is similar to the fixed-priority scheme, except that the priority queue is shifted after every request is granted. This means that agent with the highest priority, the head, will be moved to the end of the queue after any request has been granted. This is accomplished by passing a token around that determines the head of the fixed priority arbiter. This implementation ensures better fairness than a fixed priority arbiter. However, additional complexity increases power and area. A linear implementation of the round robin arbiter is shown in Figure 2.

In order to achieve the maximum fairness, a matrix arbiter can be used. The matrix arbiter implements a least-recently-served policy. For every pair of inputs, a precedence indicator is assigned that determines which agent has higher priority among that pair. These precedence indicators are stored in a matrix of registers. Anytime an input is granted, it sets the matrix to

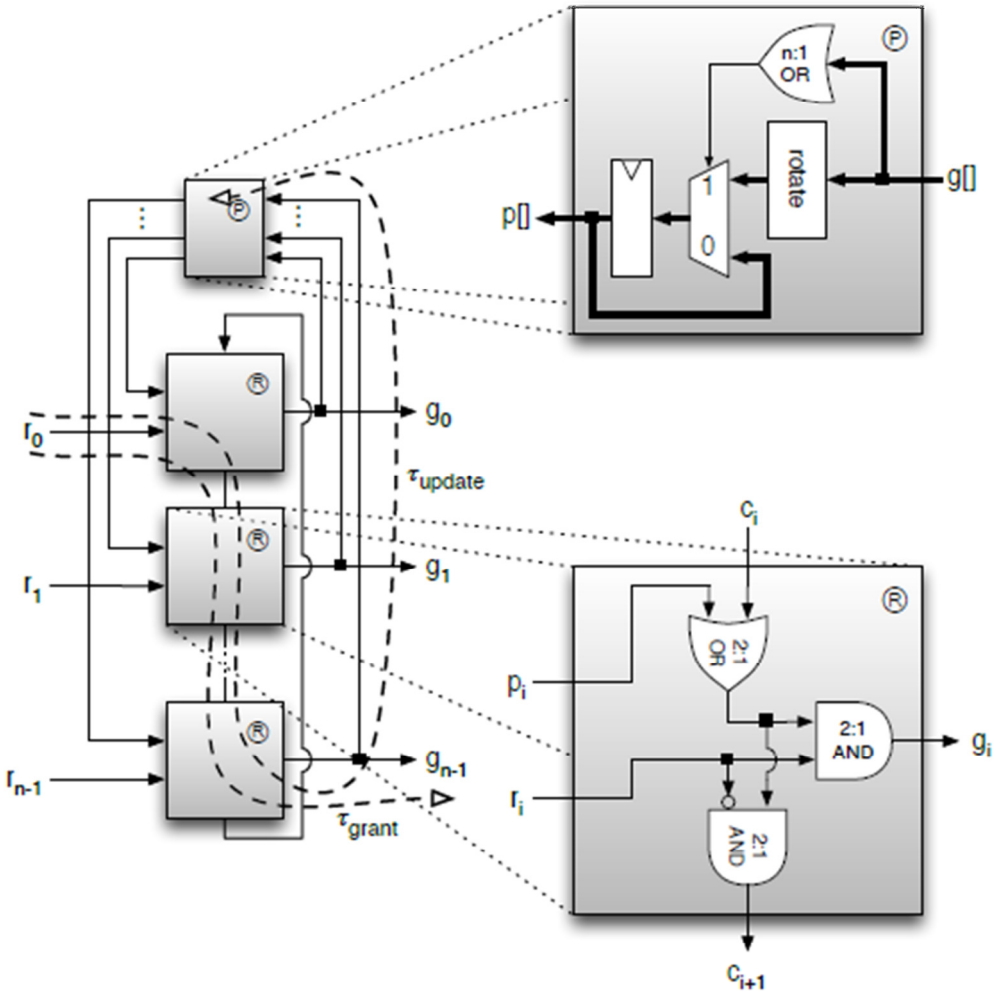

Figure 2: Round Robin Arbiter (Becker, 2012)

ensure that the most recently granted agent receives the lowest precedence among all pairs for future requests. The number of registers required to hold the precedence matrix is the primary factor in the allocator's implementation cost (Becker, 2012). Furthermore, the size of the matrix scales quadratically. This means that power and area is also expected to scale quadratically. Therefore, although matrix arbiters exhibit the most fairness, it is only attractive for a small number of inputs.

Another implementation is the tree arbiter. In some applications, agents are organized into multiple groups (Becker, 2012). In these cases, it may be advantageous to grant requests fairly among different groups rather than to individual agents. This can be achieved by implementing a hierarchical tree structure of arbiters that grants requests to groups. Tree arbiters can significantly reduce the power and area while maintaining fairness. However, these arbiters are only relevant for specific applications and do not make sense without a well-defined groups of agents.

Finally, multi-priority arbiters can be implemented in order to dynamically prioritize a set of requests (Becker, 2012). This design allows us to support a small number of priority levels similar to what we would have done in a fixed priority arbiter design. The agents within these priority groups are then arbitrated fairly. Much like the tree arbiter implementation, the multi-priority arbiter requires well-defined groups. This design has the capability of further reducing power and area, but will only work well for a specific set of agents with fine-grained priorities (Becker, 2012).

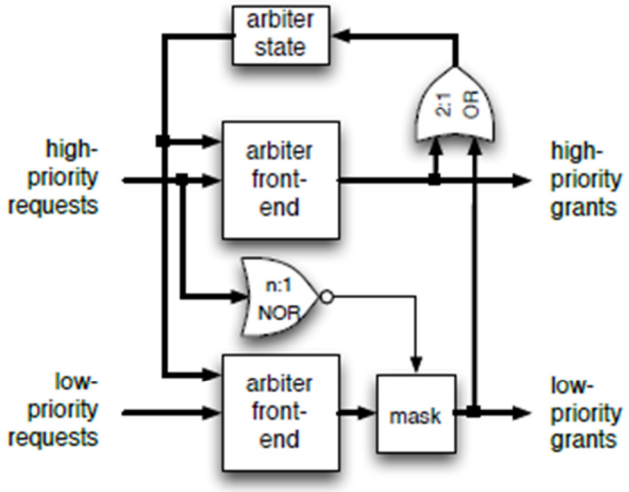

Figure 3: Dual-Priority Example of Multi-Priority Arbiter (Becker, 2012)

III. METHODS AND MATERIALS

In this section, I will discuss the necessary steps in designing and fabricating a silicon chip. A digital circuit can be represented by a hardware description language such as Verilog. This code describes circuits as logical functions. Using EDA tools, this code can be synthesized into standard cells. This synthesized design is then given to other tools to do place-and-route, the placement, wiring, and sizing of these cells, to finally produce a layout that can be manufactured by foundries. In our project, we are using Synopsys Design Compiler (DC) to run synthesis, and Synopsys IC Compiler (ICC) to run place-and-route. Because of nondisclosure agreements, we are using a Synopsys educational library of standard cells that is not a real technology. However, this library is extremely close to mimicking the real 32 nanometer technology, and our design is expected to carry over to the real libraries without major changes.

Because these tools are used to build any kind of digital circuit ranging from complex processors to single decoders, it is crucial to set up the tools properly to design optimal circuits. This involves understanding which of the myriad of features to enable or turn off, defining custom constraints on wiring and placement so the tools will use less effort, and exploring different design approaches such as hierarchical or custom designs. Much of the effort early on in the project was devoted to setting up these tools to perform optimally, and exploring different ideas to decrease the turn-around time of our designs.

As described in the overview section, large designs were initially taking far too long to complete. This was attributed to the very large amount of wiring that is required in large switches. As the number of input and output ports increase, the amount of wiring required grows quadratically because every input port needs to be connected to every output port. Furthermore, the control logic also grows quadratically. Allocators have an increasing number of agents to

handle as the number of ports increases. However, we only have a fixed number of wiring metal layers. This means that as we increase the number of input and output ports in our design, the wiring and routing between all the cells becomes more difficult quadratically.

One approach to decrease our turn-around time is using hierarchical design. Hierarchical design allows us to use unchanged portions from previous builds in our new design. This means that we can tell the tools to reuse modules from previous synthesis and place-and-route steps. If part of a design is unchanged, it is much more efficient for the tools to use the completed design from a previous build and incorporate it into the current design with minimal changes. In the Results and Discussion section, we will explore further why using hierarchical design is expected to greatly improve the efficiency of the DC Compiler as well as IC Compiler for our design.

Finally, we use another Synopsys tool called Primetime for power estimates. After place-and-route, our design is ready to be sent to a foundry for production. However, it is useful to measure and analyze the power consumption of our design. Power is an important factor to consider because efficiency and heat dissipation are becoming paramount concerns in the industry. After exercising our design by giving it stimulus, in our case packets being sent from input ports to various output ports, Primetime will report power estimates in different parts of the chip.

Using all of the tools described in this section, we have the capability to map out how different design schemes impact power, area, and throughput.

IV. RESULTS AND DISCUSSION

A. Design space for small routers

First, I will discuss the results area, timing, and throughput results collected from routers with input and output port sizes ranging from five to twenty ports. Figure 4 shows the results of the frequency plotted versus the number of ports. The four different series of data represent different number of virtual channels. Data is missing for the higher number of virtual channels because those design points lie outside of our design space. As expected, the frequency decreases approximately linearly as the number of ports increase. As the number of ports increase, the routing complexity increases, resulting in an increase in critical path and decrease in the maximum frequency of the circuit.

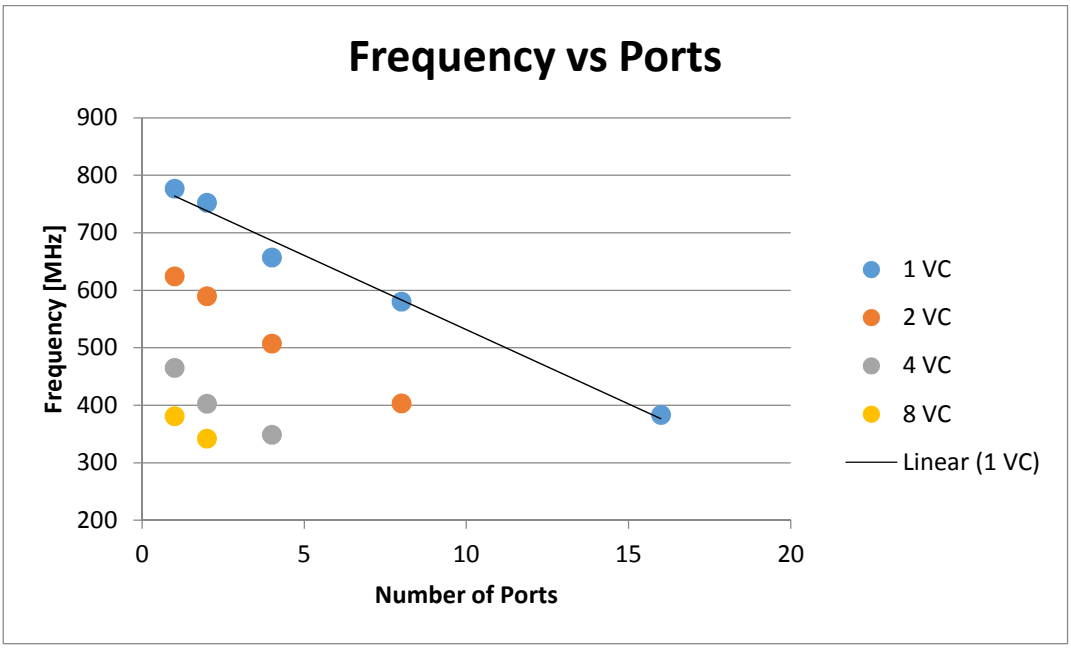

Figure 4: Frequency vs Number of Ports

Figure 5 shows the total area of designs of different port numbers and virtual channels. As the number of ports increase, the area scales quadratically. This is also expected because increasing the number of ports increases the number of standard cells and wiring required quadratically as well.

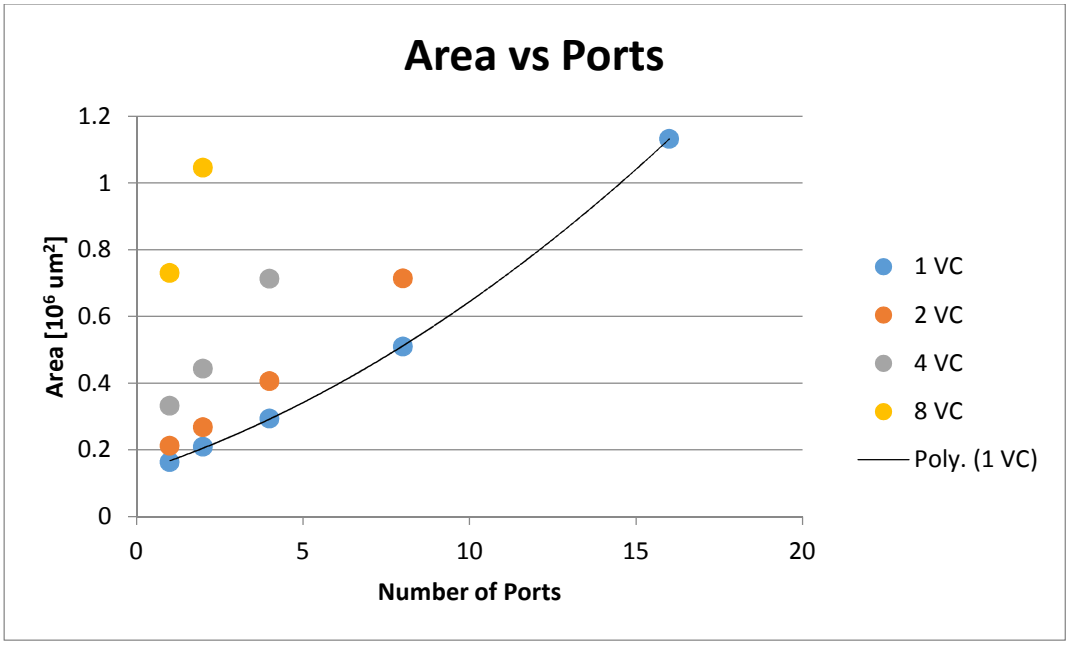

Figure 5: Area vs Number of Ports

Figure 6 shows the theoretical maximum throughput of the router. This measure of throughput assumes that every input and output port of our switch is busy, meaning that data is moving from every input port to a different output port. The throughput increases linearly as we increase the number of ports. However, we observe that the throughput begins to taper off as we move to higher port numbers. This is expected because as our design grows larger, we experience a decrease in frequency. Consequently, our circuit will run slower, and this decrease in speed overshadows the improvements of a high number of ports. This means that in order to

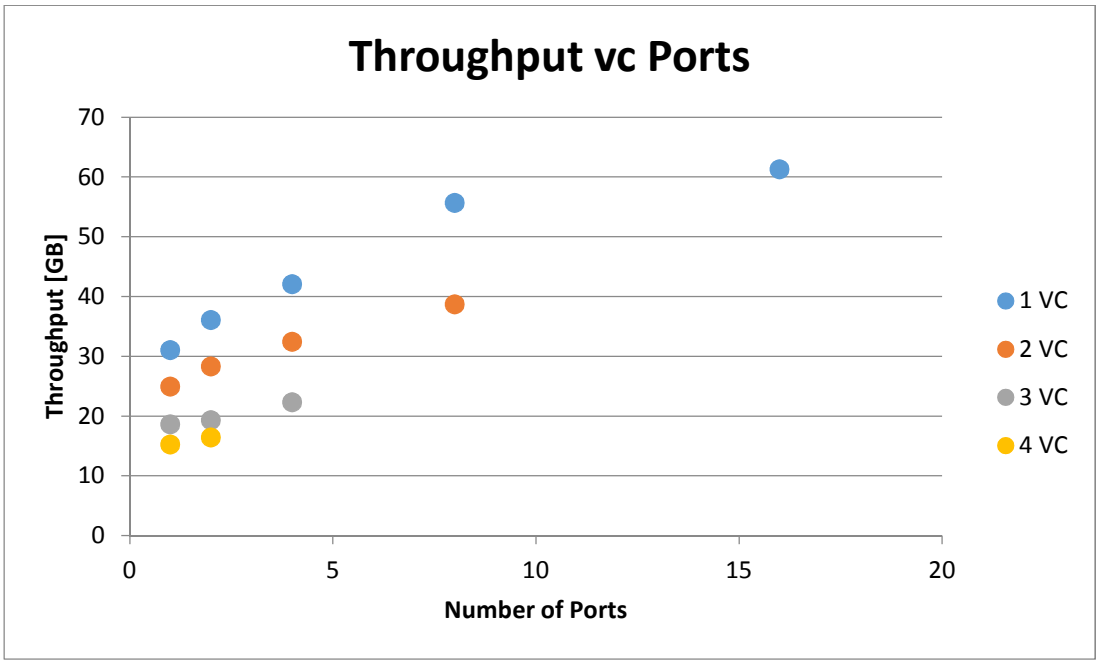

Figure 6: Theoretical Maximum Throughput vs Number of Ports

achieve high throughput, we will need to concentrate on improving frequency by decreasing our critical paths.

B. Hierarchical Design

After running incremental designs of individual modules of the switch design separately, we concluded that hierarchical design could potentially be very helpful for running our builds more efficiently. Figure 7 shows the comparison between the build times of 4 separate modules that make up a switch. These modules were all designed with the same parameters for a 64 port switch. Because of the additional time associated with running through the tools, not all the percentages are expected to add up to 100%. This analysis also ignores the overhead of placing and routing the different modules together. However, the analysis still give us a good understanding of what parts of the switch are taking the tools the longest to process.

	Area [um^2]	Area %	Build Time [hours]	Build Time %
Complete Design	13255916	100	86	100
Crossbar	3967254	29.92817697	84	97.6744186
Allocators	5625329	42.43636577	12	13.95348837
Input Control	2529408	19.08135205	4.3	4.96124031
Output Control	69632	0.525289991	1.6	1.860465116

Figure 7: Table comparing area and build time for different modules of router

One striking observation is that the build time of the crossbar alone takes over 95% of the build time of the entire switch. This was replicated over many runs. Additionally, the crossbar design only depends on the number of ports, and does not depend on any other additional parameters. This means that it is possible to build a few crossbars of different sizes once, and reuse them when we are iterating over the design space. By implementing hierarchical design we expect to improve our build runtime.

After running a small 8x8 router design through the hierarchical design flow, I discovered the following results shown in Figure 8 below. Interestingly, the build time increased for the smaller design. Because the design is relatively small, the crossbar is simple and uses a small number of gates. The benefits of using hierarchical design is overshadowed by the overhead introduced by implementing the hierarchical approach into the design flow. As expected, the critical path and area both show negative consequences. Because the physical cell for the

8x8 router design	Normal Build	Hierarchical	% Change
Build Time [minutes]	49	91	186%
Critical Path [ns]	5.18	6.09	118%
Area [um^2]	290578.675	312319.194	107%

Figure 8: Table comparing build time, critical path, and area for normal and hierarchical builds of an 8x8 router

crossbar was designed without knowledge of the loading on the input and output ports, the sizing of these gates at the boundaries are not optimized. In order to better design for timing, the loads on the crossbar ports can be estimated and used for sizing the gates appropriately. The area is also larger for the hierarchical design. This can be attributed to the sub-optimal boundaries. The crossbar is now a rectangular box, and the routing becomes heavily constrained depending on where the input and output ports of this box are located.

The results from a 64x64 router design is summarized in the Figure 9 below. Note that the build times are in hours instead of minutes. We observe a 70% decrease in build time. However, we also see a large negative impact on critical path (increase by 55%) and area (increase by 14%). This hierarchical switch design was achieved by designing a crossbar with an estimated capacitive loading on the output. When we implemented the crossbar without any loading, as we had done in the previous smaller 8x8 router design, the tools would not finish in a reasonable amount of time. Because of irregular placement and wire routing by the tools, many output ports of the crossbar actually had very different capacitances ranging from 31 fF to 120 fF. The crossbar was designed assuming average capacitive loading at every output. For these reasons, the timing and area are suboptimal. In order to achieve better timing, area, and build time, we need to give better prediction of output loading capacitances.

64x64 router design	Normal Build	Hierarchical	% Change
Build Time [hours]	86	61	71%
Critical Path [ns]	7.41	11.5	155%
Area [um^2]	$13.26x10^6$	$15.11x10^6$	114%

Figure 8: Table comparing build time, critical path, and area for normal and hierarchical builds of an 64x64 router

Although we observe negative consequences for timing and area, hierarchical design approach still allows us to decrease our total build time and iterate through designs more quickly. With better modelling and prediction of the output port loading capacitances, we can expect to improve the timing and area metrics, as well as further decrease the build time.

C. Python and Bash scripts

Finally, I wrote various Python and Bash scripts that were useful for gathering data and running the tools. These scripts improved our group's efficiency and provided a common method for us to collect and analyze data. One Python script was a data collector that parsed reports from the Synopsys tools and aggregated them into a single file. This automated process meant that we did not need to manually search through various reports and record numbers, improving our accuracy and reducing the chance for human error. Another Python script was developed to automatically change parameters in the Verilog source code. This was particularly useful when I collected data during incremental design, building individual modules separately. During this phase, specific parameters from individual modules needed to be modified because the top level module was continually changing. Finally, I also developed Bash scripts that would run the tools on the compute server farm, automatically utilizing the maximum number of cores and creating logs for the progress of the job. This script allowed us to easily monitor our jobs and analyze how much time the tools spent on each step of the build.

Concluding Reflections

Originally, my capstone team planned to map out a Pareto optimal curve of the design space for high radix routers, comparing throughput, area, and timing. However, given time constraints, we did not succeed in designing optimal switches that lie on this curve. Much of our work was focused on customizing the tool-flow in order to efficiently iterate through designs. Instead, we explored the tradeoffs of specific implementation schemes, such as types of arbiters, types of allocators, and types of buffers. This design space exploration is valuable in order to gain intuition for how a complete design would perform.

Using a hierarchical design flow is helpful for decreasing the build time for larger switch designs. This approach allows us to build modules separately and include them in our switch as a black box. Hierarchical design is important for iterating through switch implementations quickly to find optimal design schemes. Although we observe negative effects on timing and area, the results are still meaningful during design space exploration. When a final design is reached, the entire switch can be flattened and run through the tools without hierarchy to achieve higher operating frequency and smaller area. For future work, the high radix switches beyond 64 ports may be explored using the tool infrastructure we have established.

Works Cited

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

2. 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.

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

4. 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>.

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

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

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

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

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

10. 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

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

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

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

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

15. “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

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

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

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