TECHNIQUES FOR FAST, ENERGY EFFICIENT ON-DIE GLOBAL COMMUNICATION

by

Shomit Das

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

The University of Utah

August 2017

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Shomit Das**

has been approved by the following supervisory committee members:

Kenneth S. Stevens	, Chair	**07/10/2015** Date Approved
Alan L. Davis	, Member	**07/10/2015** Date Approved
Cynthia Furse	, Member	**07/10/2015** Date Approved
Roberto Suaya	, Member	Date Approved
Timothy Hollis	, Member	Date Approved

and by **Gianluca Lazzi**, Chair/Dean of

the Department/College/School of **Electrical and Computer Engineering**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Communication surpasses computation as the power and performance bottleneck in forthcoming exascale processors. Scaling has made transistors cheap, but on-chip wires have grown more expensive, both in terms of latency as well as energy. Therefore, the need for low energy, high performance interconnects is highly pronounced, especially for long distance communication. In this work, we examine two aspects of the global signaling problem. The first part of the thesis focuses on a high bandwidth asynchronous signaling protocol for long distance communication. Asynchrony among intellectual property (IP) cores on a chip has become necessary in a System on Chip (SoC) environment. Traditional asynchronous handshaking protocol suffers from loss of throughput due to the added latency of sending the acknowledge signal back to the sender. We demonstrate a method that supports end-to-end communication across links with arbitrarily large latency, without limiting the bandwidth, so long as line variation can be reliably controlled. We also evaluate the energy and latency improvements as a result of the design choices made available by this protocol. The use of transmission lines as a physical interconnect medium shows promise for deep submicron technologies. In our evaluations, we notice a lower energy footprint, as well as vastly reduced wire latency for transmission line interconnects. We approach this problem from two sides. Using field solvers, we investigate the physical design choices to determine the optimal way to implement these lines for a given back-end-of-line (BEOL) stack. We also approach the problem from a system designer's viewpoint, looking at ways to optimize the lines for different performance targets. This work analyzes the advantages and pitfalls of implementing asynchronous channel protocols for communication over long distances. Finally, the innovations resulting from this work are applied to a network-on-chip design example and the resulting power-performance benefits are reported.

To the insane bard, the deranged artist, the unhinged maestro,
and to the scientist, who is the maddest of them all.

CONTENTS

ABSTRACT . . . . . **iii**

LIST OF FIGURES . . . . . **vii**

LIST OF TABLES . . . . . **ix**

ACKNOWLEDGEMENTS . . . . . **x**

CHAPTERS

1. INTRODUCTION **1**

1.1 Exascale Computing 1
1.2 Interconnect Challenges 2
1.3 System Design 4
1.4 Thesis Statement and Contributions 5
1.5 Thesis Organization 6

2. BACKGROUND AND RELATED WORK **9**

2.1 Transmission Line Interconnect 9
2.2 Multifrequency Asynchronous Design 12
2.3 Networks-on-Chip 15
2.4 Summary 17

3. SOURCE ASYNCHRONOUS SIGNALING **20**

3.1 Introduction 20
3.2 Background 21
3.3 Source Asynchronous Signaling 24
3.4 SAS Models 27
3.5 Evaluation 30
3.6 Conclusion 32

4. CHARACTERIZATION OF ON-CHIP TRANSMISSION LINES **36**

4.1 Background 36
4.2 Simulation Setup 38
4.3 Design Choices 38
4.4 Circuit Design 39
4.5 Evaluation 41
4.6 Summary 42

5. COMPARISON OF CHANNEL PROTOCOLS **53**
5.1 Motivation 53
5.2 Overview 54
5.3 Models 58
5.4 Comparisons 61
5.5 Summary 63

6. A NETWORK-ON-CHIP EXAMPLE **73**
6.1 Introduction 73
6.2 Related Work 74
6.3 NoC Architecture and Circuits 75
6.4 NoC Simulator 77
6.5 Evaluation 79
6.6 Conclusion 80

7. CONCLUSION **89**
7.1 Inferences 89
7.2 Future Work 90

REFERENCES **92**

LIST OF FIGURES

1.1 The exascale challenge 7
1.2 Communication costs 7
1.3 Communication fabric choices 8
1.4 Clocked and asynchronous methods 8
2.1 On-chip TL configurations 17
2.2 Reflection in transmission lines 18
2.3 GALS module 18
2.4 2-phase protocol. 18
2.5 4-phase protocol. 19
2.6 Simplified relative timing design flow 19
2.7 Regular network topologies 19
3.1 Traditional and SAS communication pipelines with large wire delay of x. 33
3.2 NRZ SAS $n = 2$ vs traditional channel protocol 33
3.3 SAS two-phase handshaking protocol $n = 2$ 34
3.4 SAS sender block 34
3.5 SAS receiver block 34
3.6 Complete SAS channel 34
3.7 Energy comparison: traditional vs SAS links 35
3.8 Area comparison 35
3.9 Latency comparison 35
4.1 Characterization process flow 43
4.2 Proximity effect at high frequencies 43
4.3 Effect of conductor width 44
4.4 Capacitance limitation of increasing conductor width 45
4.5 Implications of increasing the spacing 46
4.6 Differential vs single ended lines 47
4.7 Voltage mode driver 48
4.8 Comparator receiver 48

4.9 Serializer 49
4.10 Deserializer 49
4.11 Latency of a TL link 50
4.12 Energy of a TL link 51
4.13 Energy-delay squared product of a TL link 51
4.14 Comparing RC and TL 52
5.1 No repeaters in TL channels 63
5.2 Synchronous and asynchronous topologies 63
5.3 Simulated implementation for bundled data 4-phase protocol 64
5.4 Simulated implementation for bundled data 2-phase protocol 64
5.5 Simulated DI 1-of-2 protocol implementation 64
5.6 Cycle time for various protocols 65
5.7 Latency for various protocols 65
5.8 Energy for various protocols 66
5.9 Cycle time RC vs TL 66
5.10 Latency RC vs TL 67
5.11 Energy RC vs TL 67
5.12 Wirelength effect on cycle time 68
5.13 Contribution of wire delay to cycle time (RC) 68
5.14 Contribution of wire delay to cycle time (TL) 69
5.15 Wirelength effect on energy 69
6.1 Mesh based routing examples 81
6.2 Router architecture 82
6.3 Switch circuit 83
6.4 Merge circuit 84
6.5 Latency of RC and TL mesh networks 85
6.6 Throughput and energy of RC and TL mesh networks 86
6.7 Latency of RC and TL mesh networks 87
6.8 Throughput and energy of RC and TL mesh networks 88

LIST OF TABLES

3.1 Bandwidth and energy comparison 34
5.1 Parameter variables and derivatives. 70
5.2 Operating parameters 70
5.3 Asynchronous control parameters from 65nm SPICE simulations of the designs 71
5.4 Cycle time equations measured in FO4 delays. 71
5.5 Latency per stage measured in FO4 delays. 71
5.6 Models for average energy per transaction per pipe stage. 72
6.1 TL evaluation 81
6.2 Repeated RC evaluation 81

ACKNOWLEDGEMENTS

This document, six years in the making, has several primary contributors. First and foremost, my advisor Ken Stevens whose almost boyish enthusiasm over all things technical was a constant source of inspiration. He provided clever insights to seemingly insurmountable problems, offered counsel during times of negative energy, and displayed pride at the eventual success of my endeavor. I look forward to a long, fruitful collaboration with him. Roberto was my mentor during excellent summers in Grenoble, even letting me use his home during my internship with Mentor Graphics. I cannot thank him enough for going above and beyond normal mentor duties in order to enable me to work with him in France. Tim, Cindy, and Al carefully monitored my progress over several years so I could get to this point without any major hiccups. They were always quick to respond to even trivial questions that I had, despite their incredibly busy schedules. Georgios Manetas was my electromagnetic guru during and beyond my intern days. I will always appreciate his professional and personal help during some tough transition times in Grenoble. A big shout-out to the Greek gang of Grenoble as well as Zohaib for enhancing my time spent in Europe. Apart from the three internships with Mentor Graphics, I was also very fortunate to have worked with Ram Krishnamurthy's group at Intel's Circuit Research Labs. Greg, Himanshu, Mark, and others helped me shape my understanding of physical interconnect design. Salt Lake City is one of the best places I have ever lived in. My time as a grad student would not have been the same without the help and support of DK, Para, Sanket, Rashmi, Push, Raj, Rahul, NP, Preethi, Aadi, Rungta, and other members of the JLT Gang. I will always cherish our friendship. I have had the fortune of working along side some incredible minds at the U. My sincere thanks to Eliyah, Dan, Junbok, Vikas, Krishnaji, Willy, Dipanjan, Jotham, Tannu, Mac, and Dheeraj for their help and camaraderie during my time at the lab. Julie has been a source of tremendous support during the final third of grad life. I look forward to many adventures in the coming days. My extended family of aunts, uncles, and cousins have always boosted my self-confidence and provided unwavering love all my life. I am very grateful for that. Lastly, nothing in my life would have been possible without my parents, Meeta and Nirban Das. I owe everything to them, including this dissertation.

CHAPTER 1

INTRODUCTION

This chapter sets up the motivation for the problem tackled in this dissertation. A summary of proposed solutions is provided, as well as the outline for the remaining manuscript.

1.1 Exascale Computing

As big data analytics becomes the cornerstone of contemporary commercial organizations, high performance computing is making the transition from a niche market to an everyday one. The next major milestone in the journey to shape future supercomputers is to achieve one billion billion floating point operations per second (1 exaflops). Such mammoth computing machines are dubbed *exascale computers*. Apart from predictive analytics for businesses, there are several potential scientific applications for exascale computing including:

- Mapping the human brain
- Modeling regional climate changes and severe weather patterns
- Simulation of the motion of every atom in a biochemical system
- Design of advanced materials and renewable energy sources
- Modeling fusion and fission reactions

Microprocessor clock speeds are expected to remain fairly constant in the 1 to 2 GHz range due to the linear dependence of power consumption to operating frequency. Therefore, the required performance boost has to be facilitated by a substantial increase in on-chip parallelism. This will include hardware parallelism along with supporting software framework. It is estimated that the degree of parallelism on a single chip will compare to the number of discrete nodes of massively parallel supercomputers from the 1980s [1]. However, exascale computing is not merely about assembling a large number of nodes on a die. It requires a complete reimagination of the way we compute, how we communicate among nodes, and also how we design our applications. Figure 1.1 shows the gap between exascale requirements and current state-of-the-art. Some of the assumptions that need to be updated are as follows:

- Power has to replace maximum clock frequency as the principal design objective
- Communication overheads have to outweigh computation costs in keeping with the motto of *flops are free, wires are expensive*
- Heterogeneity introduced by the rising inclusion of graphics processing units (GPUs) and accelerators in modern architectures needs to replace the prevailing ideology of adding more uniform cores to a supercomputer.
- The growth in concurrency introduced by the increased parallelism demands a migration from the the clocked to an asynchronous regime

Power is the primary roadblock to exascale systems. The apex position in the Top 500 supercomputers list is currently held by the Tianhe-2 system at the National Super Computer Center in Guangzhou, China [2]. Using 3.12 million cores, it achieves 33.86 petaflops while burning a massive 17.8 MW. If we simply scale the performance for our desired 1 exaflops, it results in a power requirement of 534 MW, which is higher than the power output of the Fort Calhoun nuclear power plant in Nebraska, U.S.A. Obviously, this is a preposterous demand; future exascale supercomputers will need to achieve their performance at more or less the same order of magnitude as current power budgets.

Similarly, connecting thousands of processing elements on a chip so that they communicate reliably among themselves is a difficult problem. To do so with a limited power budget is a daunting task.

1.2 Interconnect Challenges

This work investigates the matter of high bandwidth, low latency, energy efficient on-chip data communication. There are multiple approaches to tackling this issue. In this section, we first outline the problem, and appraise the various existing solutions. Finally, we justify the choice of path we take to approach the challenges.

1.2.1 The Cost of Moving Data

While computation energy costs are reducing, communicating data on a chip has become absurdly expensive. In a 35nm process node, energy expenditures necessary to propagate a 64b signal 2mm are approximately equal to performing a 64b floating point operation [3]. The bandwidth of a conventional diffusive wire varies inversely as the square of the wire length. Higher throughputs are possible by increasing the drive voltage, but this comes at the price of increased energy consumption. Repeaters are included in long wires to boost the signal and maintain a linear power-distance relationship; however, adding these extra active devices also increases the power consumed by

the chip. Decreasing feature sizes in modern processes increases the per unit resistance of the copper wires, which leads to two deleterious effects. Not only does the wire draw more power while moving data bits, it also performs this movement more slowly due to the increased RC constant. This means higher bandwidth imposes a higher premium on the power cost, while also keeping the throughput localized due to the power-distance correlation. It is clear that realizing exascale computing will require a redesign of the communication methodology. This dissertation examines alternative techniques for on-chip data movement. Figure 1.2 shows the energy cost of data movement at different interconnect levels relative to a floating point operation [4].

1.2.2 Existing Techniques

The interconnect problem can be dealt with using process technology to obtain more suitable dielectric materials, such as low-*k* and air-gap dielectrics [5]. New wiring materials such as carbon nanotubes [6] and graphene nanoribbons [7] have also been proposed. While these solutions have potential, we look at using design techniques for existing materials to solve the problem.

Power consumption is proportional to the squared value of the signal voltage. Therefore, **low voltage swing signaling** using differential wires has tremendous potential for power savings. However, such an apparatus is highly susceptible to noise induced by variation, coupling, reflections, and other sources. Also, sense amplifiers used in such techniques are often found to be power hungry. Another solution to offset the effects of long wires is to avoid them by vertically stacking thinned die on top of each other in a **3D** configuration [8]. The long wires are then replaced by through silicon vias (TSV) that pass through chip substrates, but reduce global wire length considerably. However, this technology is plagued by yield issues and also requires complicated design and computer aided design (CAD) tool support. Researchers have also studied the feasibility of augmenting on-chip networks with single hop, long range **wireless links** [9]. Wireless point-to-point links help overcome the shortcomings of lossy metallized global interconnects while also mitigating routing congestion. These links require additional complex circuitry such as on-chip antennae and suffer from reliability issues arising from manufacturing. **Optical interconnects** provide low latency, high bandwidth data transfer using light for signal propagation and dense wave division multiplexing (DWDM) [10], [11]. Wavelength selective ring resonators are used to distinguish different signals sent as waves with different wavelengths on a single channel. This facilitates a substantially large throughput per unit wiring area as compared to conventional electrical links. However, the ring resonators need to be constantly heated, in a process called trimming so as to be tuned to their desired wavelengths. This results in a large static power overhead. Optical-electrical interfaces and laser drivers also amplify the power requirements of this signaling scheme. There is also a large cost associated with

fabricating the necessary optical devices, such as laser sources and photo detectors on silicon. In this dissertation, we explore the use of on-chip transmission lines as a suitable replacement for diffusive wires on long links.

1.2.3 Our Approach

RLC transmission lines (TL) are fundamentally different from diffusive RC wires. Signals on a transmission line propagate as electromagnetic waves on the dielectric surrounding the lines. This means that very low latency of signal propagation is possible with this technique. Additionally, since the transmission lines essentially act as waveguides, data bits can travel farther distances as compared to RC lines, without making use of repeaters, resulting in much lower energy requirements. Transmission lines are implemented as full swing communication links in regular back-end-of-line (BEOL) metal stack without the need for exotic TSV structures or complicated circuitry like antennae and lasers, among others. However, a number of design constraints need to hold for proper on-chip deployment of RLC interconnects. These consist of ensuring precise dimensions, topologies, and aspect ratios for the lines. Transmission lines are susceptible to reflections caused by impedance mismatches at discontinuities along the line. They are also vulnerable to crosstalk induced by coupling. Therefore, care must be taken at every stage of the TL link design, from circuit design to layout, to accurately implement these long links on a chip.

1.3 System Design

The choice of signaling technique used for data transport is just one parameter in the overall implementation of the communication fabric. This work also appraises other design choices available to the system architect, such as signaling protocols and network-on-chip designs. Figure 1.3 shows an example of the communication fabric chosen for implementation from the alternatives available.

1.3.1 Signaling Protocols

Several clocked and self-timed pipeline methodologies are available for reliable communication across a channel. In synchronous schemes, data is transmitted every clock cycle between latches or flip-flops. On the other hand, asynchronous signaling utilizes handshaking for data transfer. Bundled data protocols use the same datapath as clocked techniques, but use an *ack* signal to enforce data validity. In contrast, synchronous protocols restrict data changes via setup and hold timing constraints. Figure 1.4 shows the inherent differences between clocked and bundled data asynchronous configurations. Another self-timed protocol uses 1-of-N codes to encode data validity within data wires. Handshaking signals add an overhead to the throughput and energy properties of a communication system. Naturally, clocked pipeline protocols are expected to perform better for

purely data transfer purposes. However, sometimes it becomes useful, and often times necessary, to avoid a shared global clock in a system. In this dissertation, we evaluate an asynchronous channel protocol that emulates the properties of synchronous signaling protocols, while retaining its self-timed properties and associated capabilities. We also compare the performance and power characteristics of these protocols when applied to transmission line global channels.

1.3.2 Network-on-Chip

As the spotlight moves from computation to communication in modern-day systems-on-chip, a majority of the design effort goes towards constructing an efficient, reliable, low power communication fabric [12]. A good on-chip network must transmit large amounts of data with low latency, consume almost no static power and little dynamic power, route packets with negligible error, manage congestion, and perform well even at heavy workloads, and for different applications. It must perform these tasks while connecting hundreds of nodes. It is nearly impossible to achieve all these goals with a single design; therefore, engineers settle for Pareto optimality. There are many design choices while implementing a NoC, including choice of topology, routing scheme, router architecture, flow control, etc. There are trade-offs associated with each of these options. Experts agree that network power consumption is one of the primary concerns in future exascale trends. In this dissertation, we evaluate the use of reactive, self-timed routers and signaling schemes for achieving low power NoC operation. We also explore the use of low latency, low energy, high bandwidth transmission lines as global interconnects for these NoCs.

1.4 Thesis Statement and Contributions

Given the immediate and rising demand for energy efficient, high performance, long range interconnects, this work provides solutions using three main approaches. Low latency transmission lines are designed and characterized for high frequency, low power operation. Various pipeline protocols are examined for communication over long links and a novel method is presented to duplicate the benefits of clocked signaling in an asynchronous handshaking method. Finally, a network-on-chip design is presented that incorporates the modularity of multi-frequency asynchronous operation and the signaling properties of RLC interconnects to demonstrate the efficacy of our proposed methods at the system level.

This thesis makes several contributions towards advancing our understanding of the challenges and opportunities related to building long range on-chip communication links. The contributions range from applying very low level electromagnetic principles to design better interconnects to understanding software constructs to build accurate simulators. Specifically, some of them are:

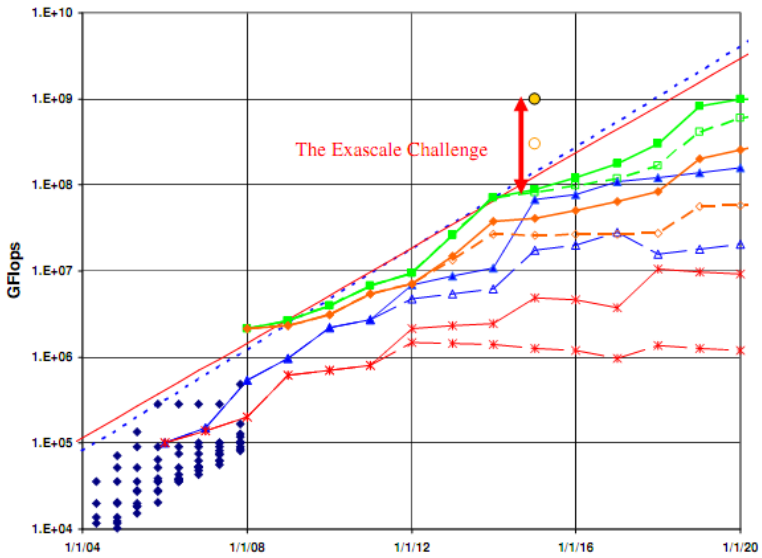
- A method to enable full communication bandwidth on wires with arbitrary delay when employing handshake communication. This method supports end-to-end communication across links with arbitrarily large but finite latency without limiting the bandwidth, so long as line variation can be reliably controlled. This technique decouples wire latency and wire lengths in an unconventional manner that is very useful in modern chips.
- Characterization of RLC interconnects for use as long haul NoC links. The effects of design choices on link performance are studied. Various circuit elements that comprise the transmission line transceiver system are compared and contrasted. Guidelines for efficient transmission line operation are provided for designers.
- A comparative study of synchronous and asynchronous pipeline protocols applied to communication over a transmission line channel. Various metrics are evaluated and the most efficient signaling protocols are identified. Detailed communication models are presented and verified with circuit simulations.
- A transmission line enabled asynchronous NoC is designed and simulated. An accurate Verilog based NoC simulator is built to appraise the effect of including transmission lines in NoC topologies. Deadlock avoidance in circular asynchronous routing topologies is studied and solutions are proposed.

1.5 Thesis Organization

This thesis is organized as follows:

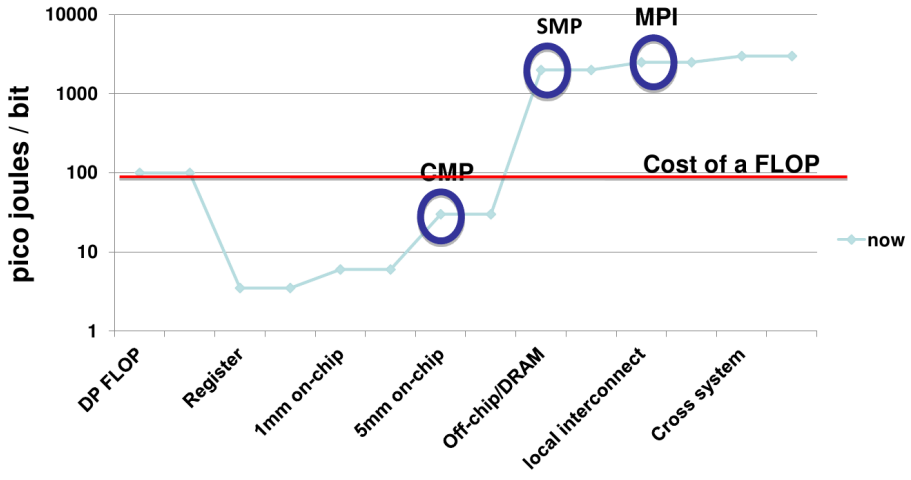
- **Chapter 2** provides some background information pertaining to the major themes used in this thesis. These include concepts in transmission line signaling, asynchronous circuit design, and networks-on-chip.
- A new channel protocol for high bandwidth communication, called source asynchronous signaling is introduced in **Chapter 3**. Various conditions to ensure correct operation are derived. The protocol is then evaluated against traditional asynchronous channel protocol.
- On-chip transmission lines are designed and characterized in **Chapter 4**. The simulation process and circuits used are described. Trade-offs of choosing one line over another in terms of channel performance are discussed.
- **Chapter 5** compares numerous synchronous and asynchronous protocols for use over long range transmission lines. First order models are derived for cycle-time, latency, and energy.
- A network-on-chip design is presented in **Chapter 6** that incorporates the techniques from the preceding chapters to evaluate their effects on system performance. This design is then compared against traditional approaches.

- **Chapter 7** provides concluding remarks and ideas for future work that will hopefully spawn more focused research in this field.



Source: "DARPA summons researchers to reinvent computing", www.extremetech.com 2012

Figure 1.1: The exascale challenge



Source: "Exascale Computing Technology Challenges [1]"

Figure 1.2: Communication costs

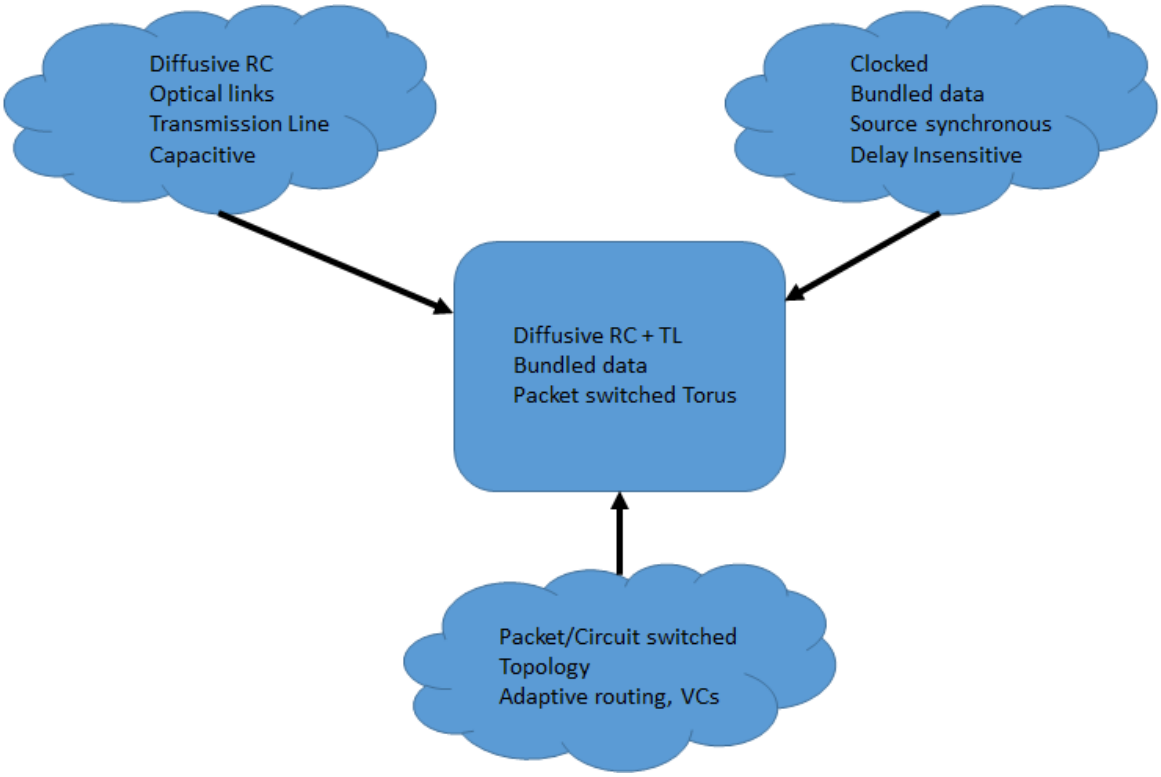
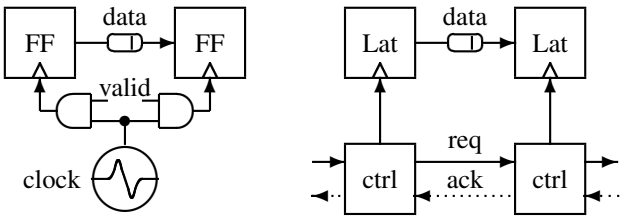


Figure 1.3: Communication fabric choices



(a) Synchronous (b) Asynchronous

Figure 1.4: Clocked and asynchronous methods

CHAPTER 2

BACKGROUND AND RELATED WORK

Overcoming design challenges in modern high performance VLSI design requires a cross-disciplinary approach. Low level circuit designers need to be aware of system implications of their design choices, while system architects need to understand the complex tweaks done at the transistor level that can affect power and performance. Mixed signal design is gaining traction as variation and other analog effects manifest themselves in deep submicron regimes. Engineering teams can no longer live in the vacua of their domains. This thesis delves into three major keystones of the communication fabric – the transmission line physical medium, the asynchronous signaling protocols, and the high level network design. In this chapter, some background in each of these spheres is presented, along with examples of previous work relevant to our approach.

2.1 Transmission Line Interconnect

By definition, transmission lines (TL) are quite simply a pair of conductors (signal and return) with nonzero length [13]. They possess some electrical and magnetic properties that are native only to them. Although classified as a passive element, TL behavior is very removed from other passive elements such as resistance (R), capacitance (C), and inductance (L). A lossless ideal transmission line can propagate electromagnetic waves at the speed of light in the dielectric medium surrounding the conductor pair. In practice, however, many factors determine the signal carrying capabilities of a transmission line as we see in the following sections.

2.1.1 Signal Propagation

A signal propagating along a transmission line uses both the signal and return paths. Close coupling between these two conductors results in superior channel performance. Therefore, if the cross-section of the transmission line remains fairly constant along its entire length, the line carries the signal better. Such lines are called uniform, or controlled impedance transmission lines. This should be the design goal of the physical designer implementing on-chip TLs. Depending on the relative arrangement of the signal and return paths, on-chip transmission lines are classified as

coplanar or microstrip as shown in Figure 2.1. The difference between single ended and differential TLs will be covered in a later chapter. Depending on the geometry, and metal and dielectric properties, transmission lines propagate electromagnetic (EM) waves in the following modes:

- TE: In the *Transverse Electric* mode, only the magnetic field waves oscillate in the direction of signal propagation
- TM: Conversely, in the *Transverse Magnetic* mode, only the electric field waves oscillate in the direction of signal propagation
- TEM: *Transverse Electromagnetic* mode dictates that both electric and magnetic field lines are normal to the direction of EM wave propagation. Due to its nondispersive nature, TEM is the preferred mode of operation for transmission lines. However, this mode imposes many unreasonable conditions on the material properties of the TL, such as infinite conductor conductivity and lossless, uniform dielectric.
- Quasi-TEM: In a non-ideal, hybrid environment such as silicon die consisting of multiple boundary regions, a *quasi TEM* mode exists where most, but not all of the EM waves are normal to the direction of signal propagation.

2.1.2 Definitions

As a passive, lossy element, a transmission line offers impedance to the signal propagating across it. At each time step, the ratio of the signal voltage to the current flowing through the line is referred to as the instantaneous impedance. If the voltage and the charge injected into the line remain unchanged, the signal encounters a fixed impedance. The only two variables that determine the impedance of the transmission line are the signal speed and per unit line capacitance. The signal speed depends on the dielectric material surrounding the wires, while the capacitance depends on the cross section of the line. It is interesting to note that the impedance does not depend on the length of the line or the signal voltage propagating along it. Therefore, for a uniform transmission line, the signal observes a constant instantaneous impedance at every point in the line. This is the **characteristic impedance** (Z_0) of the transmission line. If the line is not uniform, any perturbation in the wire geometry will cause changes in the capacitance. The resulting variation in the instantaneous impedance causes reflections and loss of signal integrity.

Signals propagating along a transmission line decays exponentially with distance. This signal degradation manifests in two ways – attenuation of the signal voltage level, and shift in the signal phase. The **propagation constant** (γ) metric is used to denote this effect. It consists of a real part, attenuation constant (α), and an imaginary part, phase constant (β). α represents the signal attenuation and is usually measured by dB/m. The phase of the signal along the line is indicated by β and

has the unit of radian/m. Z_0 and γ are often sufficient to describe the behavior of a transmission line. Other physical and electromagnetic properties as well as 2-port network (S-parameters) behavior are also used to characterize transmission lines.

2.1.3 Transmission Line Signal Distortion

Since the transmission line channel does not make use of periodic repeaters to boost the signal, extra care must be given to the matter of signal integrity along the long link. There are three main contributors to signal distortion along an RLC channel. In this section, we discuss these culprits and ways to mitigate their effects.

- Reflection: As mentioned in the previous section, any time a wave encounters a change in instantaneous impedance down a transmission line, part of the wave will be reflected back to the source as seen in Figure 2.2. The reflected wave travels back to the source, and is reflected back again. This back-and-forth continues until the reflected wave attenuates completely. At the ends of the transmission line, these spurious signals cause overshoot and undershoot of the desired signal; this effect is called ringing. Ringing is harmful because it increases power consumption due the added current drawn by the reflected signals. Also, at the receiving end, the extra signal transitions may lead to logic and timing errors, thereby compromising the passivity of the line. Reflections can be avoided, to some degree, by keeping the TL link uniform. However, this may not always be possible due to the presence of vias, bends, forks, and joins in an on-chip wiring topology. Terminating one or both ends of the transmission line with resistive or capacitive elements minimizes reflections in the line. The high resistance of metal wires in modern processes serve as a damping mechanism for reflected signals, thereby providing an intrinsic series resistive termination.
- Attenuation: Signal loss in a transmission line is frequency dependent. Higher frequency signal components get selectively attenuated more than the low frequency components. This uneven frequency domain attenuation causes an increase in the signal rise time, and subsequently, a decrease in signal bandwidth. Also, simple boosting repeaters do not work in this case due to the selective nature of signal loss. Attenuation can be attributed to the losses occurring mainly in the conductor and the dielectric. Coupling to other lines and radiation also contribute to signal loss, albeit insignificantly. The nonlinear frequency response of the channel means that different components of the same signal arrive at the receiver at different times, leading to multipath propagation. This leads to blurring together of successive symbols at the receiving end, known as inter symbol interference (ISI). The intuitive solutions to solving this problem is to either boost the high frequencies at the sending end (**preemphasis**),

or to filter out low frequency components at the receiving end (**equalization**). This approach, however, introduces extra circuitry to the TL communication system.

- Crosstalk: Crosstalk is the superposition of a spurious signal from one net to another due to coupling between the two wires. Coupling is caused by the fringe electric and magnetic fields surrounding a conductor. Changing electromagnetic fields from one wire induce current in the adjacent wire. This manifests as distortion in the signal at both near and far ends of the line. The obvious method is to increase the separation between the affected wires and/or introduce guard bands between them. However, this reduces the per unit wiring area throughput of the link. Other engineering solutions include using wide area return planes instead of return lines, decrease the characteristic impedance of the line, increasing the signal rise time, and reducing the coupling length of the wires.

2.1.4 Existing Work

Transmission line signaling is the primary medium of data transfer in off-chip environments. There have been a few attempts at engineering on-chip RLC channels. Beckmann [14] first suggested using transmission lines to reduce the communication latency incurred while accessing L2 caches. In earlier process nodes, using TL links in upper metal layers in lieu of diffusive wires translated to a direct 30$\times$ communication speedup. The numbers in current technology are not so optimistic. Suaya outlined the use of transmission lines for clock tree networks [15]. Chang [16] used modulation to up-convert data signals to a carrier wave and transmit over a microstrip TL. However, the added RF mixer circuitry increased the power budget and complexity of the communication link. A current mode logic transceiver was demonstrated by Ito et.al. [17]. This work demonstrated reliable TL signaling on a fabricated die using simple circuitry and differential transmission lines. A complete serial link with SERDES circuits was implemented by Flynn [18]. The architectural advantages of using transmission lines were reported by Carpenter [19]. In this domain, it is clear that we stand on the shoulder of giants.

2.2 Multifrequency Asynchronous Design

The complex nature of modern VLSI design has led to the proliferation of *intellectual property* (IP) blocks on a chip. These IP blocks may be designed by different teams and optimized for different goals, resulting in different operating frequencies. The overall System on Chip (SoC) is comprised of these blocks stitched together using a communication fabric. The predicted heterogenous nature of future exascale systems indicates that this organization is here to stay. Clock domain crossing required to communicate between different parts of the same chip is not only expensive

in terms of latency, power, and area, but also impacts the reliability of the system. Asynchronous design, by its very nature is intrinsically suited to adoption in this multi-frequency regime.

2.2.1 Globally Asynchronous Locally Synchronous

Globally Asynchronous Locally Synchronous is a design style that allows IP blocks to operate in synchrony with their associated local clocks, yet employs a self-timed communication fabric to facilitate *talking* among the modules [20]. Clock domain crossing interfaces are wrapped around each block that translate between the asynchronous fabric and the synchronous domain. This enables a kind of 'plug-and-play' modularity in the system. Clocked design is well understood and widely adopted in both industry and academia. GALS allows engineers to make use of highly optimized IP blocks without having to synchronize to a global chip clock. On the other hand, there are several misgivings regarding asynchronous circuit design such as inadequate computer aided design (CAD) tool support, potential races and hazards in the final design, trouble with testability, among others. GALS technique allows circuit designers to remain in their comfort zone while unlocking the advantages of modular, multifrequency system design. However, this comes at a price – each block is driven from a local pausible clock generator to stretch or pause the clock during data transfer, as shown in Figure 2.3. This is usually achieved with the help of ring oscillators. Metastability is a concern in such implementations, and must be accounted for [21]. Other options to realize GALS designs are to use synchronizers to communicate between completely asynchronous or loosely synchronous timing domains.

2.2.2 Basics of Asynchronous Circuit Design

An asynchronous design is not tied to a single clock frequency. Instead, it is reactive and is operational only when valid data presents itself. The role of timing in digital design is essentially to impose sequencing. Using a central timing reference for this purpose is inefficient in terms of power and performance. However, it makes design simpler and lends support to CAD tools that exploit the *lockstep* arrangement imposed on the signals. Self-timed designs enforce sequencing and flow control with simple handshaking. In cases when high performance and low energy are demanded from the system, self-timed designs stand out over traditional clocked design due to their adherence to the average-case performance in contrast to the worst-case performance enforced by the clocked design style. Power constrained designs especially are tailored for asynchronous circuits since they immediately get rid of the power spent in the clock tree. Asynchronous designs also possess intrinsic modularity and concurrency that make them suitable for use in highly parallel architectures.

There are several ways to design asynchronous circuits. The easiest way to differentiate design styles is to consider the type of timing assumptions made regarding the design. **delay insensitive**

(DI) designs are completely agnostic to gate and wire delays. This highly conservative nature of timing means that the resulting circuits are slow and power hungry. One simple relaxation made to the DI assumption, namely that wire forks have the exact delay (isochronic forks), enables the circuits to be much less bulky. This class of circuits are called **quasi delay insensitive**. If wire delays are assumed to be zero, yet gate delays are arbitrarily large, the design style is called **speed independent**. The fastest, smallest, and most energy efficient self-timed circuits are made possible using **timed circuits**. In this case, gate delays have predefined upper and lower bounds. Naturally, these extra timing constraints warrant extra effort for timing verification. Each style has its own advantages and disadvantages. For example, while DI designs are bound to be larger than their timed counterparts, they are also more robust against PVT timing variations, and far easier to design.
Depending on the number of signal transitions per valid transaction, asynchronous protocols can be classified as 2-phase or 4-phase. The 2-phase or non-return-to-zero (NRZ) protocol consists of two transitions to complete a data transaction. It is also called *transition signaling* protocol. A pair or transitions, either high-to-low or low-to-high, one each on the request and acknowledge lines indicates a valid data transfer. While this protocol is more efficient as compared to its 4-phase counterpart, circuits for this scheme are usually larger due to the edge detection requirements of the protocol and at the storage level. On the other end of the coin, return-to-zero or *level signaling* 4-phase protocols use four signal transitions per datum transferred. Consequently, there is a large overhead in acknowledging each transaction, which makes it unwieldy for communication purposes. However, circuits adhering to this formula are smaller and easy to design. Generally speaking, circuits are designed using the 4-phase strategy and converted to 2-phase for transmission over long distances. Figures 2.4 and 2.5 show the waveforms for the two protocols.

2.2.3 Relative Timing

This work uses relative timing (RT) [22] methodology as the sequencing technique for the design of all network components. The RT based design flow allows the designer to specify, characterize, and validate clockless elements using synchronous CAD tools. Instances of these characterized blocks can then be inserted in the regular VLSI CAD flow with some additional timing constraints. An RT constraint consists of a common timing reference called point-of-divergence, or pod, and a pair of events that are ordered in time for correct circuit operation called the point-of-convergence, or poc. A constraint is represented as $\mathsf{pod} \mapsto \mathsf{poc}_0 + m \prec \mathsf{poc}_1$ where poc_0 must occur in time before poc_1 with margin m. Hence, the maximum path delay from pod to poc_0 must be less than the minimum path delay from pod to poc_1. This is easily represented by two related design constraint equations `set_max_delay` and `set_min_delay`, which perform timing driven synthesis

that enforce the constraints on the logic paths. The complete RT design flow is represented in Figure 2.6

2.3 Networks-on-Chip

Traditional bus and point-to-point communication schemes do not scale well for *many-core* systems. To meet the aggressive performance specifications of most present-day applications, it is necessary to design a network that has suitable routing, flow control and topological schemes. On-chip networks differ from macroscale off-chip networks in several ways. On-chip networks are required to be light-weight due to the limited power and area budgets. Most of the communication latency of on-chip networks derives from router delay instead of wire delay since they span shorter distances. Therefore, router design plays an important role in the performance of the network. Also, NoC bandwidth is not limited by pin availability as in the case of off-chip networks. In this section, we discuss the various considerations that are factored in during the design of an on-chip network.

2.3.1 Topology

The network topology describes the arrangement and connection of various components of a network such as routers and links [23]. Direct topologies are defined as schemes where each router is connected to a node. Indirect topologies allow cases where intermediate routers are not connected to nodes. A hop is defined as a physical connection between neighboring routers. The topology of a network determines the number of hops required for a packet to travel from source to destination. The network diameter is the maximum number of hops required to traverse from one node to another such that the packet does not loop, backtrack, or detour; and is an important characteristic of the topology. Another important feature of the network is the path diversity, which is a count of the number of unique paths that exist between a pair of nodes. This is a key consideration when traffic load distribution and fault tolerance of a network are important. The network topology is decided based on these factors. Some examples of network topologies are shown in Figure 2.7.

2.3.2 Routing

The routing algorithm determines the path taken by the packet to travel from source to destination. If the path is fixed at the source, it is called *source routing*. *Destination routing* is when intermediate routers resolve the path based on the destination address. If multiple paths are available, routing algorithms choose the route based on optimizing a target metric, which can be hop count, latency, bandwidth, etc. Adaptive routing algorithms can change the paths for different packets traveling between a source-destination pair to balance load traffic, or to avoid deadlock. While employing adaptive routing, care must be taken that out-of-order packet arrival is accounted

for. In contrast, a deterministic route always uses the same link to go from a source to a destination. Care must be taken in this method to avoid deadlocks. Deadlock occurs when packets cannot progress further down the NoC due to a cyclic dependency of the shared resources (buffers and channels). In most topologies, deadlock can be avoided by restricting certain turns that can create cycles. X-Y dimension ordered routing is one such example.

2.3.3 Flow Control

There are choices for the flow control as well. **Circuit switching** first sends out a probe to reserve channels along a path for a communication flow. After the sender receives word (ack) that the route is ready, it streams data freely without having to worry about contention or deadlock. After the entire message is transmitted, another signal is sent from the source to free the reserved links. This overhead of reservation and de-reservation is justified only when a large amount of data is transmitted. In **packet switching**, the packets are injected directly into the NoC and link allocation is performed on a hop-by-hop basis at the routers. In *store-and-forward* flow control, a router stores all flits of a packet before forwarding it to the next router. This increases the required buffer capacity of the router FIFO to accommodate the entire packet. In the *virtual cut-through* technique, storage and bandwidth requirements of the router remain the same, however flits can be forwarded before the entire packet has been received. *Wormhole* reduces the buffering capacity of input ports at the router to the flit granularity level. While this greatly reduces the size of the routers, wormhole flow control is susceptible to head-of-line blocking where a flit blocked at the head of the FIFO queue due to congestion also blocks the flits behind it, even when the routes associated with the blocked flits are not congested. If different flows have separate queues at the router, the head-of-line blocking can be avoided. This is called the *virtual channel* flow control, because separate channels appear to each flow, although only one physical channel exists.

2.3.4 Simulators

Early forms of on-chip network simulators were usually single-threaded, which meant that their turn-around time was high. Unfortunately, this high run-time scales very poorly with the rising number of nodes in future exascale networks. The introduction of transaction level modeling (TLM) introduced the concept of function calls to model communication. Recent efforts have been focused on parallelizing NoC simulations using a multithreaded approach on highly parallel GPU architectures. One problem with these programs is the high synchronization cost incurred. Other efforts, based on workload decomposition have been proposed [24]. Full-system simulations provide a high degree of accuracy at the cost of higher run-time, by evaluating the correlation between the NoC and the applications. Sampling based approaches have been investigated where

a small sample is used to represent the entire application [25]. NoC simulators also need to take into account various choices in network design, such as topology, flow control, fault tolerance, etc. [26] Also, with use of unconventional links such as wireless, optical interconnect, and transmission lines, low level link accuracy gains further importance. Evidently, the problem of fast, reliable, accurate on-chip network simulators that can accommodate hundreds of nodes is a critical unsolved problem.

2.4 Summary

In this section, some background has been provided that will help in understanding the following chapters. It is highly recommended that the reader consult the references provided for the further understanding of some of the salient points in this chapter.

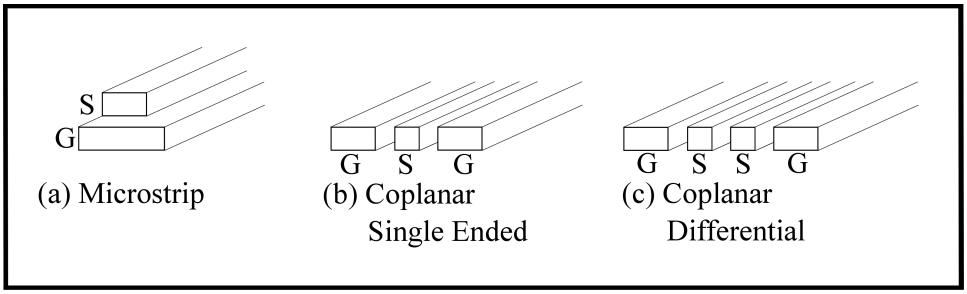


Figure 2.1: On-chip TL configurations

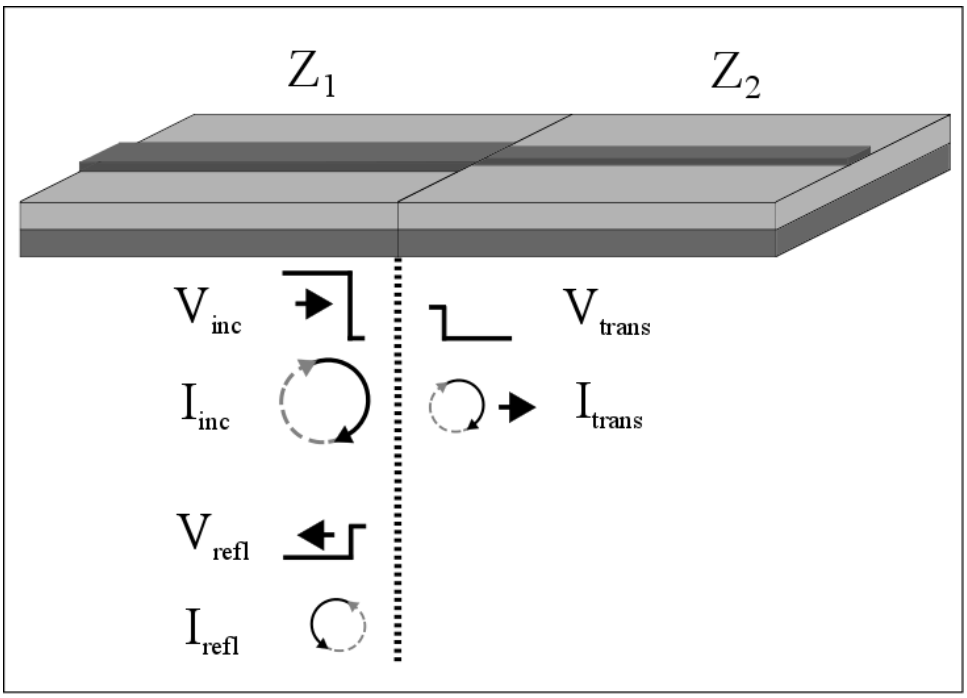


Figure 2.2: Reflection in transmission lines

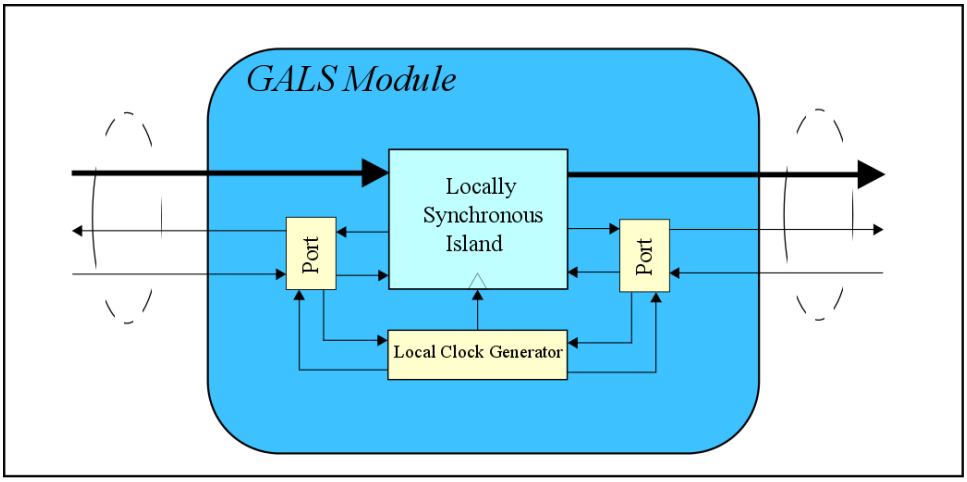


Figure 2.3: GALS module

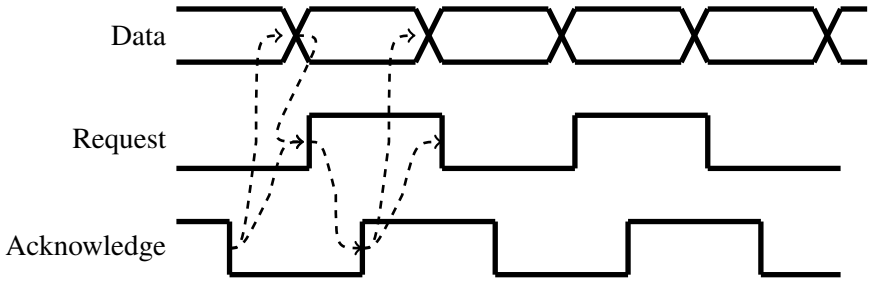


Figure 2.4: 2-phase protocol.

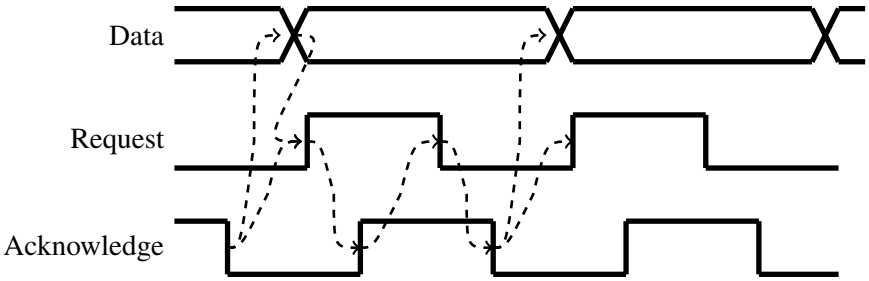


Figure 2.5: 4-phase protocol.

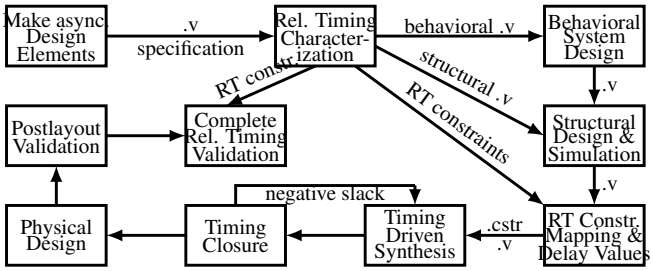


Figure 2.6: Simplified relative timing design flow

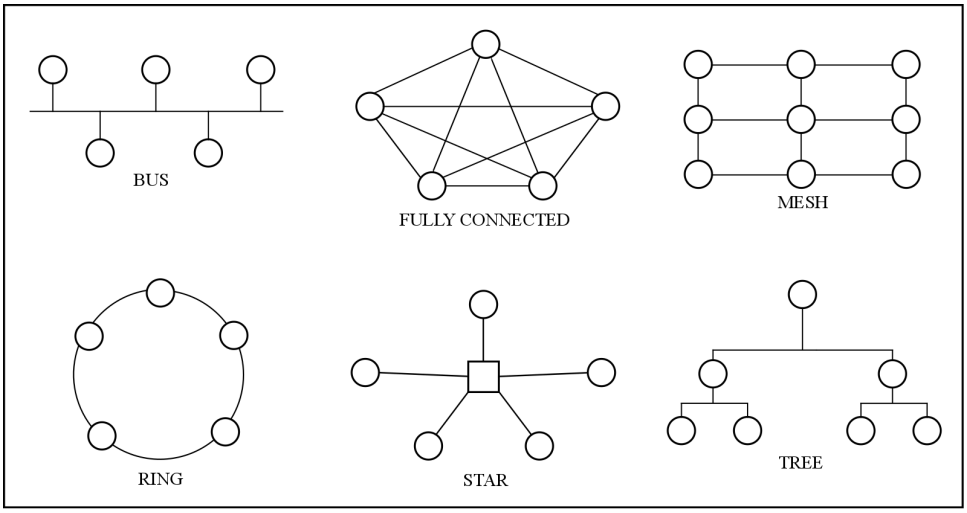


Figure 2.7: Regular network topologies

CHAPTER 3

SOURCE ASYNCHRONOUS SIGNALING

In the absence of a universal timing scheme in clockless paradigms, the choice of the communication protocol carries high importance for functional correctness as well as performance. This chapter introduces a novel handshaking protocol that imparts the desirable throughput characteristics of a clocked scheme while maintaining the modularity of an asynchronous method.

3.1 Introduction

Latency down a communication link on an integrated circuit is dependent upon the resistance, capacitance, and current carrying capabilities of the wires. Each new process generation scales the technology down by reducing the cross sections of the wires while simultaneously placing them closer together. This increases the number of communication links in a fixed millimeter squared area of an integrated circuit. However it also modifies the signal carrying properties of the wire. In a scaled wire, capacitance remains about the same but resistance substantially increases due to the reduced cross sectional area. The increased resistance produces a relative increase in the communication latency down a fixed length of wire [27]. Increasing delay and energy properties on interconnect pose design challenges, particularly with global wires or network-on-chip interconnect.

Communication employing asynchronous handshake protocols is accomplished by sending data accompanied with validity information down a communication channel. Flow control is established by acknowledging the receipt of the data, thereby enabling the transmission of new data. The handshake signals are generated using pipelined control logic that implements the protocol and synchronize two adjacent channels. The frequency of communication is determined by the delay of the control logic plus the latency down the communication wires. Maximum operating frequency for any specific controller design is established when the pipelined controllers are physically adjacent to each other, as in a first-in first-out buffer (FIFO). Increasing distance between pipeline stages increases wire delay, and decreases the operating frequency of the system.

The direct consequence of wire latency is easily observed when employing asynchronous request acknowledge handshake-based communication. Every additional picosecond of wire delay

due to controllers being placed farther apart directly results in at least two picoseconds of degradation in the cycle time (1ps each for request and acknowledge). As the communication distance between control elements increases, the communication overhead increases, with a commensurate decrease in operating frequency and communication bandwidth. At some wire length, the communication delay eventually exceeds the desired performance target.

Traditional solutions to this problem reduce the impact of wire overhead. This is accomplished by employing two-cycle communication, which reduces the number of transient communications down the channel in half compared to four cycle protocols, and placing pipeline stages closer together, which reduces communication latency [28, 29, 30].

A new protocol called source asynchronous signaling (SAS) is provided for asynchronous handshake based communication. Rather than mitigate the wire overhead, this protocol is completely independent of wire delays. Thus the SAS link can achieve the same bandwidth as a traditional pipelined link shown in Figure 3.1. The SAS protocol therefore allows for high throughput asynchronous communication for channels with large wire latency. A method of implementing SAS is provided, and results are evaluated and compared against traditional protocols.

The SAS protocol decouples the request and acknowledge handshaking signals in such a way that multiple request operations may occur without an acknowledgment operation, and multiple acknowledgment operations may occur without an intervening request operation. Such communication by definition can not be delay insensitive as handshake events are not directly acknowledged. This results in a necessary set of relative timing constraints that must hold for the circuit to be functional as well as to perform at the desired bandwidth without stalling [22].

3.2 Background

This work builds on existing handshaking based communication protocols for data transfer. This section compares existing techniques and demonstrates the bridge to this work.

3.2.1 Related Work

This work is similar to clocked “source synchronous” signaling [31]. We are aware of unpublished work as far back as the mid-1980s when Hewlett Packard began transmitting the clock along-with data between memory chips and microprocessor chips. This solved the problem of distributing low skew clocks in systems where data is transmitted over long distances. Similar properties exist today on a single die due to increasing wire delays and transistor counts. Thus, source synchronous signaling is relevant for today’s system-on-chip and network-on-chip designs [32, 33, 34].

Clocked source synchronous signaling is challenging for two reasons: it creates timing relationships between the sender and receiver nodes, and it necessitates detailed evaluation of the signal carrying properties of the wires. Applying source synchronous concepts to asynchronous handshake protocols is no different; timing relationships and signal integrity become central issues. This work defines all timing constraints necessary for the SAS protocol as applied to the design presented in this work.

The topic of signal integrity is largely left as related work. However, due to its fundamental importance we briefly touch on key aspects related to SAS. A robust method of repeater deployment on long wires must be used to maintain linear wire delay, reduce variation, and maintain signal fidelity. The critical repeater distance is the maximum distance between line drivers that must be maintained to ensure linear delay and signal fidelity. In several 65nm technologies this distance is approximately 555 μm. First order equations for communication latency and variation have been developed and validated against SPICE showing approximately 15% maximum error [30]. These models can be used to quickly estimate long communication link robustness to variation, energy, and latencies for any particular physical design configuration.

Since the SAS protocol is independent of wire delay, one can theoretically set channel frequencies to arbitrarily high values. Wave pipelining can result for very long wires and/or very high frequencies. In such cases multiple signals are concurrently in flight down a communication channel. Wave pipelining requires more care in physical design of the communication channel. However, multiple researchers have shown that high speed wave pipelined signaling is possible with proper engineering [35, 36, 37].

In order to put wave pipelining and signal integrity issues in perspective, consider our highest frequency network-on-chip design operating at 2.6 GHz in a 65nm process [38]. The delay in picoseconds of well managed interconnect for this process is modeled with the linear regression equation $len/10 + 16$, were len is the wire length in microns [39, 40]. This equates in the the network-on-chip router to the time-of-flight down a 3.69mm link. At that distance a skew of 150ps (a 39% variation) in either a delay-insensitive or bundled data protocol on a SAS channel still provides a very robust margin of 85ps.

Previous work defining a method for asynchronous handshake-based source synchronous signaling has been performed [41]. That work contains two channels, one for forward communication and one for backward communication, each with a FIFO. The design from [41] requires interface logic in order to connect traditional handshake protocols to the dual-channel internal protocol. The specifics of the internal channel protocol are not defined. In contrast, this work only contains a single channel, and no interface logic is required. The SAS channel protocol is formally defined.

One FIFO is placed at each end of the channel. FIFO sizes can be reduced to zero when cycle times are sufficiently large or wire delays sufficiently small. In such a condition the SAS protocol and provided implementation becomes equivalent to a traditional handshake channel. This base case is directly represented in our timing models when no SAS buffers are required.

3.2.2 Channel Properties

A channel consists of data, validity information identifying when the data are stable and may be sampled, backward flow control, and the controllers that implement the communication. Traditional handshake communication performance depends on the wire delays in the communication channel and the performance of the pipelined controllers. Data validity is provided by a separate req signal for bundled data channels or encoded in the data for delay insensitive channels. Backward flow control is normally provided with an ack signal. Handshake communication channels are characterized by (i) the channel protocol, (ii) the data encoding employed by the channel, (iii) how and when data are stored, (iv) the amount of concurrency implemented between adjacent channels, (v) the design of the controller that implements concurrency between channels, and clocking of the local storage elements.

Asynchronous channel protocols are very simple. Delay-insensitive channels forever repeat the sequence of a valid data encoding, ack, null data encoding, ack. Bundled data channels consist of a forever repeating req and ack events with the associated data relationship [42]. Channel protocols can take the form of a "return-to-zero" (RZ) or "four-phase" communication, or reduce the number of transitions by using "non-return-to-zero" (NRZ) or "two-phase" communication. There are many examples of different pipelined communication channels and their associated controllers in the literature. Any such controller that can be used to build FIFOs is a candidate and thus can be used in this design.

The SAS protocol introduces a new channel protocol, but otherwise supports all data encodings, data and control relationships, and concurrency between channels. The general design of the controller which interfaces between SAS channels and traditional handshake channels is provided herein. The design is based on FIFO structures with specific signal integrity and timing relationships that must hold. This paper demonstrates the SAS protocol using bundled data encoding. This encoding requires $w+1$ wires to encode w bits of data as well as a communication wire labeled ack to engineer the acknowledge handshaking. The SAS channel is described in this section using a two phase protocol.

3.2.3 Channel bandwidth

Equations 3.1 to 3.3 represent the maximum communication bandwidth for clocked and traditional asynchronous handshake protocols. Variable w is the width of the data-path in terms of the number of data bits it carries, and delays are converted into frequencies by putting them in the denominator. The maximum bandwidth of a clocked design B_{clk} is proportional to the wire latency down a communication channel (L_c) plus the setup time to the flip-flop (SU). B_2 is the maximum bandwidth of a two phase communication protocol. It is proportional to the sum of C_{Or}, the request to acknowledgment responds time of the output channel, plus twice L_c. This is because in the two phase protocol, there is one transition each on the req and ack handshake signals. The maximum bandwidth of the four phase protocol B_4 is proportional to twice the response time of the output channel C_{Or} plus four times the channel latency L_c due to the four required transitions on the handshake control signals.

$$B_{clk} = w/(L_c + SU) \tag{3.1}$$

$$B_2 = w/(2 \times L_c + C_{Or}) \tag{3.2}$$

$$B_4 = w/(4 \times L_c + 2 \times C_{Or}) \tag{3.3}$$

$$B_{SAS} = w/C_I \tag{3.4}$$

Assume delays $L_c \gg (SU \approx C_{Or})$, so that L_c is the dominant delay in the above equations, and that the clocked and asynchronous overheads (SU and C_{Or}) are effectively identical. In such a case, the clocked design has approximately twice the maximum bandwidth of a two-cycle asynchronous design, and the two-cycle asynchronous design has approximately twice the bandwidth of a four-cycle asynchronous design. This two to four penalty factor for handshake communication can seriously hamper competitiveness in terms of performance, area, or power. SAS bandwidth (Equation 3.4) is only dependent on the input channel frequency and number of data bits. Now if $C_I \approx L_c$ then the SAS channel operates at the same frequency as the clocked design. Thus the SAS protocol overcomes the communication disadvantage, but comes at the cost of adding timing constraints to the design.

3.3 Source Asynchronous Signaling

This section provides details on the novel SAS protocol.

3.3.1 Protocol Specification

Figure 3.2 shows the formal representations of a bundled data NRZ SAS channel protocol with a buffer depth of two ($n = 2$) and a traditional bundled data NRZ handshake channel protocol. Each

protocol is represented as a petri-net and can be used as a specification for the traditional and SAS communication channels.

Figure 3.3 shows an example simulation of a two phase NRZ SAS communication channel where $n = 2$. Rather than requiring that acknowledgment signal arrives before the next request signal, data transfer stalling is delayed by three ack transactions. This allows up to $n + 1$ request transactions to occur before an acknowledgment transaction. As a result, the third data value becomes valid and the third req signal asserts indicating data validity, but flow control of the SAS Channel Protocol requires that the data remain valid and the third data transfer transaction does not complete until after the first acknowledge transaction (ack) occurs. Thus for the SAS channel communication protocol, the acknowledge flow control is shifted based on the number of data items n that can be buffered in the SAS sender and SAS receiver elements.

3.3.2 Implementation of SAS Channel and Interfaces

SAS sender controllers interface a traditional channel protocol to the SAS Channel Protocol; SAS receiver controllers perform the dual operation. One can write a petri-net specification to synchronize a SAS channel and a traditional channel protocol in order to build SAS sender and receiver controllers. Our experiments synthesizing such specifications resulted in complex and slow designs. We then realized that the concurrency between the channels very closely mimics the behavior of a FIFO.

Figure 3.4 shows the block diagram representation of the bundled data SAS sender controller. The controller consists of an n-deep first-in first out (FIFO) that interfaces between a traditional handshake communication channel as an input and a SAS communication channel as the output. SAS controllers may use bundled data or delay-insensitive channels; however, this design does not work for GasP or single track communication.

The function of the FIFO is to record how many data transfers have occurred, and to not complete any transactions on the input channel that would exceed n transactions on the SAS communication channel. This is accomplished by implementing an n-deep FIFO to communicate between the traditional and the SAS communication channels. If $n = 2$, then two complete request acknowledge transactions can occur unimpeded on the input channel, and the corresponding data are sent down the SAS output channel. A third transaction may begin and the data transfer down the SAS communication channel would be initiated. However, the transaction would not complete until an acknowledge transaction occurs on the SAS communication channel. Since the depth of the FIFO is two, and two tokens have been placed in the FIFO, the third request transaction on the input port will not be acknowledged until an acknowledgment is received on the SAS channel.

Note that the data and the data validity information (req) on the traditional input channel are the same as the data and data validity information on the SAS communication channel. Other logic may be placed on these signals for various reasons such as to improve signal fidelity, delay the timing reference, change the data and timing encoding, or other standard modifications. The output request (rr) from the FIFO is left unconnected. This results in timing assumptions that must hold for the circuit to operate correctly. All transitions on the rr output of FIFO must occur before an associated response occurs on the acknowledge signal on the SAS channel that connects to the ra signal on the FIFO. This requires that the delay between edges on the ack signal is greater than the response time of FIFO.

The SAS sender element is not limited to bundled data protocols. Dual-rail, m-of-n, LEDR, or any other data encoding may be used in the channels and FIFOs. Likewise any type of asynchronous handshake controller that can implement a FIFO may be used so long as the design meets the SAS timing requirements imposed upon the design.

The SAS receiver controller is represented in Figure 3.5. This circuit is in many ways the dual of the SAS sender controller. This element contains a data FIFO that interfaces an input SAS communication channel with a traditional output asynchronous handshake channel. The input SAS channel and traditional output channel may use any data encoding, data transfer protocol, or protocol concurrency.

The function of the FIFO is to buffer and output the data received on the input SAS channel to the output channel. The size of the FIFO will be n, and this will be the same depth as the sender FIFO.

The acknowledge signal (ack) on the output handshake channel and the SAS communication channel are the same signal. The input acknowledgment (la) from the FIFO is left unconnected. This results in timing assumptions that must hold for the circuit to operate correctly. All transitions on the acknowledge signal la on the FIFO input must occur before the next transition on the req signal from the SAS communication channel. This requires that the delay between edges on the SAS Channel req signal is greater than the response time of the FIFO.

Figure 3.6 shows the block diagram of a complete SAS communication system with interfaces to traditional communication channels. The wires across the SAS Channel may be very long with substantial delay. Both the request and acknowledge timing signals across this channel are not directly acknowledged. Therefore, care must be taken to ensure that the fidelity of the signals and the relationship between the data and the timing signals hold across this channel. Repeaters will be inserted on the wires if they are longer than the critical repeater distance to ensure signal fidelity. Wave pipelining may occur on the SAS channel.

3.4 SAS Models

For a correctly designed SAS communication system, the maximum bandwidth B_{SAS} is expressed in Equation 3.4 where C_I is the cycle time of the channel. Note that this equation is strictly dependent upon the target data frequency to be sent down the channel, not upon wire delays as is the case with traditional communication channels as defined by Equations 3.2 and 3.3. Thus, the SAS channel provided can achieve wire overhead free communication down a long communication channel. However, for the SAS communication system to operate properly, a number of constraints must hold, which are now defined.

The forward and backward latency (Equations 3.7–3.10) of all FIFO designs are a function of the depth n of the FIFO. For linear FIFOs, the latency is n times the latency of each stage in the FIFO. Many designs exist that reduce the forward and backward latencies, such as parallel, tree and square configurations [43]. In these designs the latency is not calculated as n times latency per stage; rather there is a more complicated function to determine latency. For example, in an asynchronous tree FIFO, the latency is to the first order $\log_2 n$ times the latency per stage [44]. For the latency values used in this document, the forward and backward latency values $L_{b_S}(n)$, $L_{f_R}(n)$, and $L_{b_R}(n)$ for each FIFO calculate the first order approximation of the latency based on the FIFO design and structure which is a function of n.

L_{SAS}	Latency of a repeated wire down SAS Channel	(3.5)
n	Depth of sender/receiver FIFOs	(3.6)
$L_{f_S}(n)$	Forward latency of sender FIFO as a function of n	(3.7)
$L_{b_S}(n)$	Backward latency of sender FIFO as a function of n	(3.8)
$L_{f_R}(n)$	Forward latency of receiver FIFO as a function of n	(3.9)
$L_{b_R}(n)$	Backward latency of receiver FIFO as a function of n	(3.10)
C_I	Minimum cycle time of the input channel	(3.11)
C_O	Minimum cycle time of the output channel	(3.12)
C_{Or}	Output channel req-to-ack response time	(3.13)
C_{Sf}	Maximum cycle time of SAS sender FIFO	(3.14)
C_{Rf}	Maximum cycle time of SAS receiver FIFO	(3.15)

Overhead free communication only occurs when the channel operates at frequency $1/C_I$ without stalling. A few fundamental conditions must hold for this to be the case based on the above variables.

$$C_I \geq C_O \geq C_{Sf}, C_{Rf} \tag{3.16}$$

The cycle time of the input and output channels must be greater than or equal to the FIFO cycle times. Otherwise the FIFO will stall the input or output channel(s). Likewise, the cycle time of the input channel may not be less than the cycle time of the output channel, otherwise the input channel will eventually stall under full bandwidth traffic. Since the input channel request drives the SAS receiver FIFO, and the request transaction is not acknowledged by the FIFO, this FIFO must operate faster than the input channel. Likewise, the SAS sender FIFO handshake with the output channel is not acknowledged and the similar condition holds.

The cycle time of input channel C_I is the base factor all equations depend upon, as it dictates the frequency of operation of the SAS communication system. It represents the desired operational frequency for the design. This cycle time is based on the input stage controller and SAS receiver FIFO delays, input channel wire delays, and the rate at which other pipeline stages limit data to that channel. Every other component must operate at least that fast.

The following two values are necessary to define the remaining constraints of a SAS communication system. They are based on the fundamental delays and properties of the links and FIFOs defined above.

$$T_{S_n} \quad \text{Time to fill the sender FIFO with } n \text{ tokens} \tag{3.17}$$

$$T_{R_{SAS}} \quad \text{Response time of the SAS system} \tag{3.18}$$

The response time $T_{R_{SAS}}$ is the time from when one data transaction is asserted on the input channel until the entire SAS communication system becomes idle and the sender FIFO is empty when no new tokens are added to the system and none of the channels stall.

Variables T_{S_n} and $T_{R_{SAS}}$ can be represented using values from Equations 3.5 to 3.15. The SAS receiver FIFO will stall the input channel on transaction $n+1$ if no acknowledge transaction occurs on the SAS channel. The response time across a SAS communication system is approximated by the latency down the SAS channel L_{SAS}, the forward latency of the receiver FIFO $L_{f_R}(n)$, the response time of the output channel C_{Or}, the latency back down the SAS channel, and the backward latency down the sender FIFO $L_{b_S}(n)$.

$$T_{S_n} = (n+1) \times C_I \tag{3.19}$$

$$T_{R_{SAS}} = 2 \times L_{SAS} + L_{f_R}(n) + L_{b_S}(n) + C_{Or} \tag{3.20}$$

If $T_{S_n} < T_{R_{SAS}}$ the input channel will stall because FIFO in the SAS sender element is full and cannot accept more transactions until the response time $T_{R_{SAS}}$ occurs allowing a new transaction to be stored in FIFO. Thus the following equation must hold for SAS system to operate without stalling.

$$T_{S_n} \geq T_{R_{SAS}} \tag{3.21}$$

Substituting in the values for the above equations results in the following fundamental inequality for SAS systems. The minimum buffer size n for any SAS system design can be calculated from Equation 3.22 when a valid solution exists. A valid SAS solution will exist if the left side of the inequality increases as n grows. This occurs when the cycle time of the input channel is greater than the sum of the forward latency of the receiver FIFO and the backward latency of the sender FIFO when one more stage is added to each.

$$(n+1) \times C_I - (L_{f_R}(n) + L_{b_S}(n)) \geq 2 \times L_{SAS} + C_{Or} \tag{3.22}$$

This inequality shows that the depth of the FIFOs n is critically dependent on channel cycle time and FIFO latencies. The most efficient designs will select a FIFO in the SAS sender element that has a small backward latency L_{b_S} and a FIFO in the SAS receiver element that has a small forward latency L_{f_R}. Another key parameter for SAS communication system designs is the input channel cycle time C_I. As the data frequency increases (cycle time C_I decreases) the depth n of the FIFOs increase substantially. The challenge of building a robust system also increases as the input channel frequency increases due to issues of signal fidelity down the SAS channel as well as the tighter constraints on the FIFOs. Area also increases due to the larger FIFO depths n.

As the input channel cycle time C_I increases, the inequality demands fewer stages in the FIFOs. When the cycle time C_I becomes sufficiently large in relation to the channel delay L_{SAS}, the solution for n becomes zero in SAS Equation 3.22. In this case, there are no SAS FIFOs in the design and Equation 3.2 and 3.22 are equivalent if $C_I = B_2$. The SAS circuit also becomes identical to a traditional 2-phase bundled data pipeline. This makes perfect sense, because the SAS timing constraints ensure that the unconnected FIFO signals obey a proper traditional handshake protocol, but they are no longer necessary once the FIFOs are no longer needed.

Based on Equation 3.22, some SAS communication system designs will not produce valid solutions. In practice this has been shown to be the case for SAS systems that use simple linear FIFOs. Assume simple linear FIFOs are used in the SAS sender and receiver blocks. These FIFOs have the property where the latency grows linearly with the number of stages and where the sum of the forward and backward latencies equals the FIFO cycle time. Therefore, one can

let $L_{f_R}(n) = n \times L_f$ and $L_{b_S}(n) = n \times L_b$ and $C_{lf} = L_f + L_b$ where L_f and L_b are the latencies of a single pipeline stage respectively and C_{lf} is the cycle time of the FIFOs. Assume input channel frequency C_I is equal to the FIFO cycle times and $C_{lf} = C_{Sf} = C_{Rf}$, which is valid according to Equation 3.16. Substituting these values into Eqaution 3.22 yields $C_I \geq 2 \times L_{SAS} + C_{Or}$. Thus when using linear FIFOs, correct operation can be independent of the FIFO size and the design will not correctly implement the SAS channel protocol. We have validated this with design and simulation examples.

$$L_{f_R} + 2 \times L_{SAS} \geq L_{f_S} \tag{3.23}$$

$$L_{b_S} + 2 \times L_{SAS} \geq L_{b_R} \tag{3.24}$$

Two additional timing constraints for SAS communication system are shown in Equations 3.23 and 3.24. This ensures that on initiation the forward latency when coming out of a stalled state, the empty token can propagate to the input of the sender FIFO by the time the receiver observes the acknowledgment and a new token arrives at the sender FIFO input.

3.5 Evaluation

Performance of the SAS communication system is evaluated and compared against a traditional channel employing a common burst-mode four-phase handshaking protocol. The analysis is done for 5,000μm and 10,000μm channels in a 65nm process with a critical repeater distance of 555μm. This corresponds to 18 repeaters over the length of the wire. Data width of both links is 64 bits. The same four-phase linear controller is used for the SAS FIFOs and in the traditional communication channel. The circuits were designed in behavioral Verilog, synthesized using Synopsys Design Compiler, and placed and routed using SoC Encounter. Circuits were simulated for timing and functional correctness using Modelsim with postlayout parasitics back-annotated. Testing was performed using predefined input vectors. Various performance parameters including forward latency, cycle time, and throughput were also generated from the simulation along with VCD (Value Change Dump). The simulation VCD file along with the parasitics of the place and routed design was used to calculate the power numbers for each design by PrimeTime PX.

3.5.1 Bandwidth

According to simulation results the unpipelined wire of length 10,000μm can support a bandwidth of 222 Mbps. This bandwidth is far too low to be practical for the evaluation of the conventional handshake channel in actual communication link of that distance. Pipelining has to be included to obtain a fair appraisal of handshake communication in a real world system-on-chip.

One example is to consider the case where all repeaters have been replaced with pipeline latches giving a pipeline link length of 555 μm. The corresponding bandwidth of a 64-bit link is provided in Table 3.1 where n refers to the number of pipeline latches along the link and depth of the SAS FIFOs. Note that the bandwidth can be improved by changing the pipeline granularity. However, this comes at a cost of energy and area. We shall investigate these scenarios later in the section.

The bandwidth of the SAS communication channel is independent of the wire latency, and therefore the link length. The rate of data transmission across the channel is determined only by how fast the sender and receiver FIFOs operate. Tree FIFOs at the sending and receiving ends with a capacity $n = 10$ are utilized to obtain a successfully operating SAS channel at high bandwidths. Care was taken to ensure that the design met with the constraints outlined in Section 3.4. The bandwidth of the bus is calculated based on the cycle time of these FIFOs and represented in Table 3.1. The energy for transmitting data across each wire in the traditional and SAS cases is also compared. We observe that the SAS communication channel allows data to be transmitted over 30% faster at nearly 20% less energy.

3.5.2 Energy

As discussed earlier, conventional asynchronous handshaking can achieve higher bandwidths if a higher degree of pipelining is applied. Higher bandwidth targets results in a finer pipeline granularity which ultimately leads to a higher energy cost. We compare the energy expended in transferring data to meet different bandwidth targets over wires of length 5,000μm and 10,000μm. The wire latency per pipeline stage that meets the bandwidth target is calculated for a four phase asynchronous handshaking channel. The corresponding link length per pipeline stage is derived from that value, which eventually provides the total number of latches required for the entire link. This number is multiplied by the energy consumed by a single pipeline latch, and the product is added to the wire energy across the entire link. Note that at higher bandwidths, the pipeline controllers need to be spaced closer than the repeaters.

In case of the SAS channel, the choice of appropriate FIFO structures were determined using Equation 3.22. The rule of thumb in this case was to choose a FIFO with cycle times less than the target frequency. The sender FIFO was chosen to have a low backward latency, and the receiver FIFO had lower forward latency. In most cases, these criteria were satisfied by a linear FIFO at the sending end and a parallel FIFO at the receiving end. The capacity of the FIFOs, n, was calculated based on the SAS equations. The total energy of the entire setup, including the wire energy is depicted in the figures below.

It can be seen in Figure 3.7 that the energy benefit of using a SAS channel is more prominent in

longer wires at higher bandwidths. For a 10,000μm wire, operating at a bandwidth of 1.8 Gbps, the SAS channel consumes 74% less energy per data transfer when compared to the traditional channel.

3.5.3 Area

The same experiment as the one used for energy analysis is used to determine the area benefits of employing the SAS protocol. The added silicon used for extra pipeline stages manifests itself in the added area numbers. On the other hand, SAS has a lumped distribution of silicon along the length of the wire, with major logic cells only at the sending and receiving ends and repeaters spaced at regular intervals. Note that the wire area remains the same; the added benefit is in the silicon area metric. The SAS channel will have fewer pipeline controllers than the traditional channel, but will require more wire repeaters.

Figure 3.8 shows the comparison of area cost of the traditional and SAS communication systems. The 1.8GHz SAS solution for a 5mm and 10mm communication channel required the same number of SAS buffers, and hence these solutions have identical area. It can be seen that the SAS channel implementation results in a reduction of 66% of the silicon area as compared to a traditional four phase protocol for a 10,000μm wire operating at 1.8 GHz.

3.5.4 Latency

Latency down a communication channel is the sum of the wire latency and the control logic overhead. The use of a large number of pipeline controllers adds a significant delay to data communication over a long wire, especially for high bandwidth requirements. The SAS channel only has a FIFO at the receiving end in the forward path, along with repeaters at regular intervals. This allows for a very low latency transfer of data over the channel. Figure 3.9 shows the comparison of forward latency values for the traditional and SAS communication links.

3.6 Conclusion

A novel high throughput asynchronous handshake signaling protocol that is unhindered by wire latency is introduced. Some simple relative timing constraints must hold for the design to operate correctly. The new technique also results in lower latency and area as well as energy savings when compared to traditional handshaking methods that achieve the same bandwidth. For the examples provided in the paper, energy, area, and latency are reduced by 75%, 66%, 77%, respectively for a 1.8GHz channel down a 10mm line. The key feature of this protocol is that it allows the bandwidth of data transfer to be independent of the link length. The added latency of longer wires is simply compensated by increasing the capacity of the sender and receiver FIFOs. The SAS channel protocol and design elements provided in this paper apply equally well to two phase or

four phase signaling. A SAS communication system also works equally well for bundled data or delay insensitive protocols. Likewise, the design can be automated due to the ability to define timing requirements and communication requirements and to characterize asynchronous FIFOs based on these requirements. Also, any channel protocol, be it more or less concurrent, may be used to implement the FIFOs and SAS communication channels and traditional channels. When channel delays become sufficiently large and signal fidelity can be sufficiently controlled, wave pipelining can be implemented with the SAS channel protocol by allowing multiple transactions to be concurrently in flight down the SAS channel.

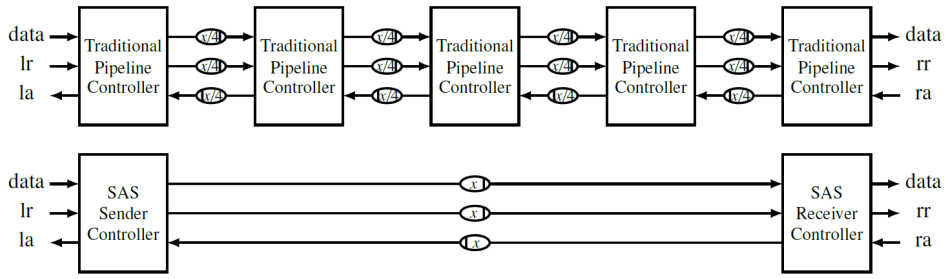


Figure 3.1: Traditional and SAS communication pipelines with large wire delay of x.

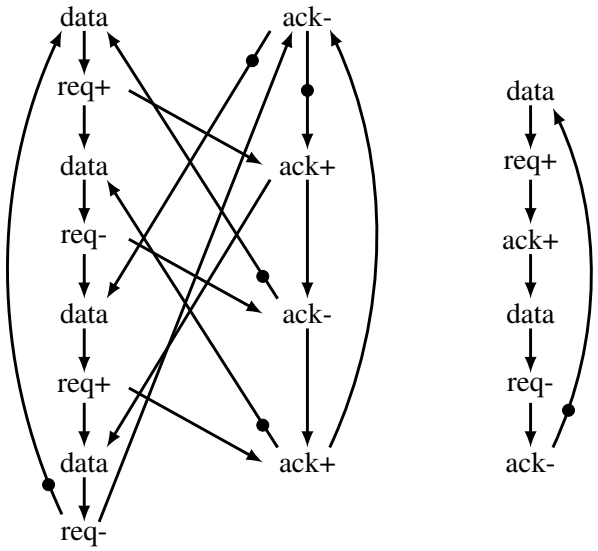


Figure 3.2: NRZ SAS $n = 2$ vs traditional channel protocol

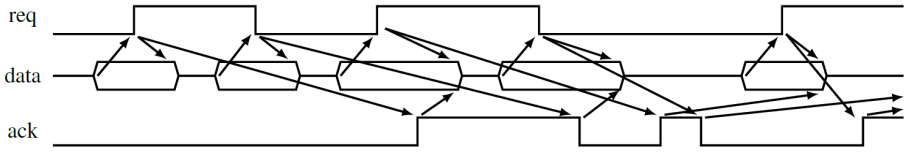


Figure 3.3: SAS two-phase handshaking protocol $n = 2$

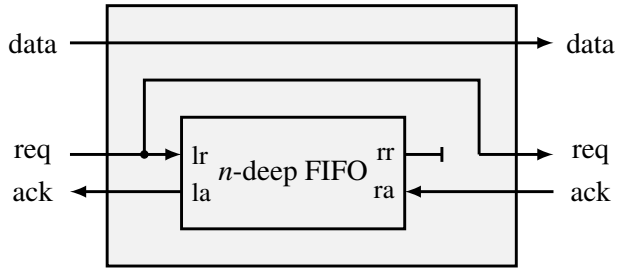


Figure 3.4: SAS sender block

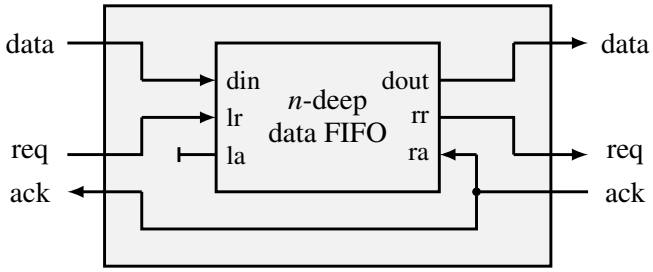


Figure 3.5: SAS receiver block

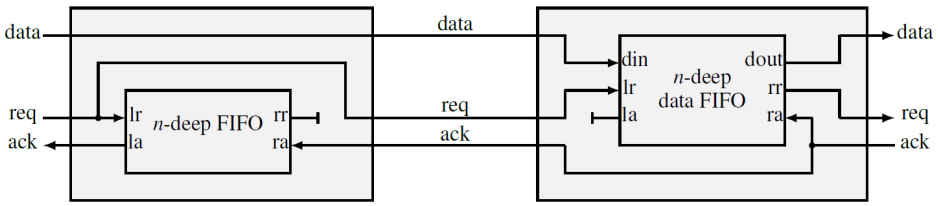


Figure 3.6: Complete SAS channel

Table 3.1: Bandwidth and energy comparison

Channel	n	**Energy (pJ)**	**Bandwidth (Gbps)**
Traditional	18	16.66	88.03
SAS	10	13.51	116.36

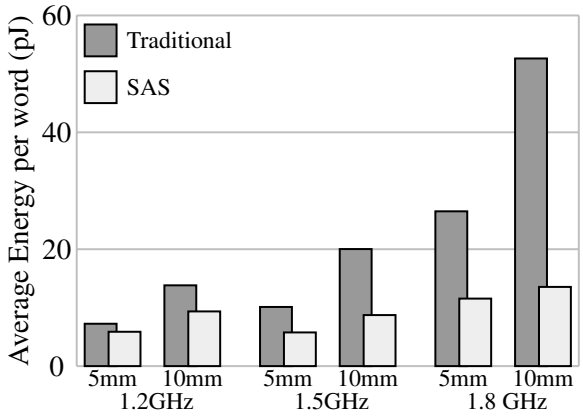


Figure 3.7: Energy comparison: traditional vs SAS links

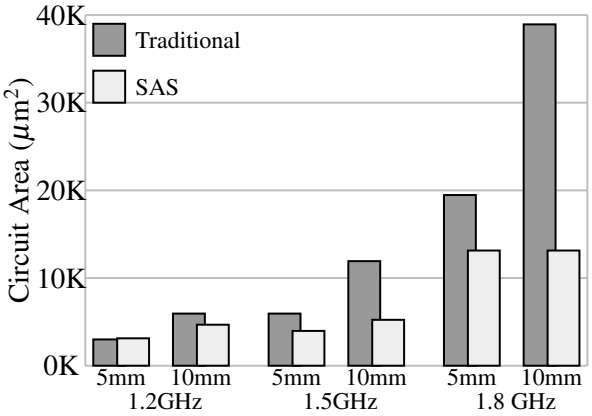


Figure 3.8: Area comparison

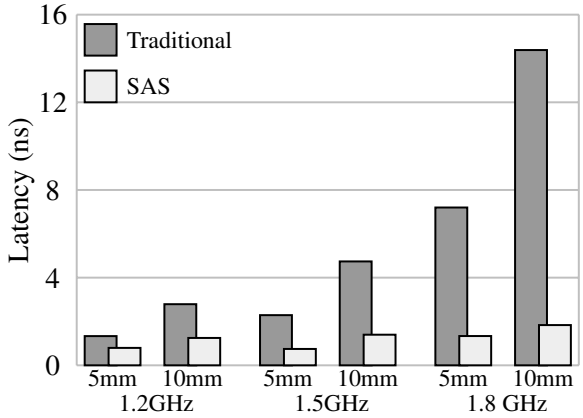


Figure 3.9: Latency comparison

CHAPTER 4

CHARACTERIZATION OF ON-CHIP TRANSMISSION LINES

Scaling shrinks wire cross-sections. The subsequent increase in the per unit wire resistance results in poor interconnect performance. As the physical distance between transistors on a chip decreases along with their size, local wires scale as well. However, global wires traveling across the length of the chip suffer massive loss in performance. This results in the performance deterioration of the entire communication network. Repeaters are employed at regular intervals to boost the signal and maintain a linear delay-length relation. Similarly, pipeline stages are introduced on a long wire to preserve cycle time requirements. However, adding these circuits on wires adds to the power budget. An alternative approach is to use low latency transmission line waveguides that can propagate signals long distances at high frequencies without repeaters. The per bit energy consumption is low due to the absence of repeaters, as well as potentially high frequency data transfer. These advantages come with the caveat of significantly higher wiring area, sensitivity to noise, and frequent use of analog circuitry to drive/sense the lines. There is often a gap in understanding between the needs of a NoC designer and the electromagnetic properties of transmission lines. In this chapter, we attempt to bridge that divide by linking interconnect performance with TL design choices. This will allow engineers to optimize channel behavior using physical design choices.

4.1 Background

To understand the connection between the behavior of a transmission line and its geometry, we first need to map the physical parameters such as line widths and thickness to electrical terms like resistance, capacitance, and inductance.

4.1.1 Resistance

Like any physical material, the transmission line has a resistance (R) associated with it that describes its opposition to the flow of current. For a conductor block like the transmission line, the resistance is proportional directly to the length of the wire and inversely to its cross-sectional area.

This means that thicker wires make better conductors. Therefore, it is always desirable, and often necessary to implement transmission lines on the top metal layers of the BEOL stack. The resistance also depends on the wiring material, which means that copper interconnects are more suitable for TL links. The primary cause for signal attenuation and inter-symbol interference is the behavior of the resistance of the transmission line, especially at higher frequencies.

4.1.2 Capacitance

The capacitance (C) is the measure of how much current can flow between a pair of conductors per unit voltage difference between them. For an arrangement of conductors similar to those found in chip environments, the capacitance can be approximated to vary directly with the amount of area overlap between the conductors, and inversely with the separation between them. Since transmission lines consist of a signal-return pair, closely coupled lines have a higher capacitance. The dielectric material between the lines also plays an important role in determining the capacitance. High-k dielectrics, like those used in top metal layers imply higher capacitance in those layers. The electric field lines that exist between conductors when a current flows across them consists of fringes. Unfortunately, this means that there is considerable unwanted coupling between conductors in a wire-rich on-chip environment. This is the primary cause of crosstalk between signal wires in transmission lines.

4.1.3 Inductance

The dual of capacitance and electric fields is inductance with magnetic field lines. Each current carrying conductor has magnetic field lines around it. Inductance (L) is a measure of number of field lines around a wire per unit current flowing through it. If the current is flowing in the same conductor whose inductance is being measured, it is called self inductance. Mutual inductance is the measure of the number of field lines due to current in a neighboring wire. Changing magnetic fields induce a voltage across the conductor. This magnetic coupling can introduce spurious signals very easily within a conductor. Closely coupled signal and return paths decrease the effective inductance of the wires by boosting the mutual inductance between them.

4.1.4 Proximity Effect

Proximity effect refers to the phenomenon that occurs when two conductors that are physically close to one another carry high frequency signals. Current flowing through a neighboring conductor causes the current distribution in the original conductor to be confined to smaller regions. This *current crowding* increases the effective resistance of the wire at high frequencies. The cause of this current crowding are the eddy effects induced by the alternating magnetic fields of the adjacent

wire. Since this added resistance applies only to the high frequency components of the signal, it causes distortion and closing of the signal ”eye”.

4.2 Simulation Setup

Accurate simulations of transmission lines at high frequencies is not possible with either lumped or distributed models. For uniform geometries, a 2D field solver is used to calculate per unit R, L, and C values for different frequencies. A field solver solves Maxwell's equations with the geometry of the conductors as boundary conditions. In this work, we use the built-in 2D field solver in HSPICE. This solver is based on an improved version of the boundary-element method and the filament method used in Raphael [45].

The characterization process is illustrated in Figure 4.1. First, the BEOL stack of interest is modeled as accurately as possible by drawing a geometric arrangement using metal conductivity, metal layer and inter-layer dielectric (ILD) values, metal and ILD thickness and others as inputs. These values are obtained from the process foundry guides. The materials, layer stacks, X-Y locations of the conductors, and their shapes are all specified as inputs to the field solver. For current process nodes, the technology files are often locked and unavailable to the designer. In such cases, best approximations are deemed sufficient. The geometry of interest is described in the control file. The transmission line is modeled as a W element in HSPICE. This choice yields far better accuracy than comparable U models without tuning the optional parameters. The W element also provides support for a large number of coupled conductors, while placing no restriction on the sparsity of the RLGC matrices. The parameters specified for the field simulation include the number of conductors, node names for the signal and reference terminals, and the field solver model used. A range of frequencies is then listed along with other simulation information such as duration and time-step of the simulation run. The RLGC values are then calculated and stored in an output data file. The efficacy of using the transmission line is then checked using equations from [46]. If found suitable, SPICE models are extracted from these files, to be used for transient simulation. The transient simulation uses the field solver output W model as the device under test (DUT). Circuit structures to drive and detect signals over the transmission lines are specified in the stimulus file, along with load information. Finally, results are accumulated to characterize the transmission line geometries.

4.3 Design Choices

This section is intended to serve as a designer's guide to choosing the optimal transmission line from a set of available geometries. We explain the various transmission line effects manifesting

in our simulations and elucidate the choices we make during the design process. We start with a single ended line flanked on both sides by return paths connected to ground. From the observed R and L values, we notice the presence of the proximity effect, as suspected. This is reflected in Figure 4.2. It can be seen that as we move from 1 GHz to 50 GHz, the per unit resistance increases by 44%. We then go on to observe the effect of changing the signal wire width on the resistance and inductance. As seen in Figure 4.3, both low frequency and high frequency resistance decreases with increasing signal width. Inductance remains fairly unperturbed. However, we cannot make the signal conductor width arbitrarily large. Wider conductors have larger fringe fields which makes the transmission line increasingly susceptible to wires present in the metal layers immediately above and below it. Since there are no guard bands in the vertical direction, there is no way to protect against this scenario. This effect is reflected by the increase in capacitance, as shown in Figure 4.4.

It is interesting to note that a similar effect can be actuated, albeit on a smaller scale by increasing return path widths, while keeping every other parameter constant. However, the marginal decrease in resistance does not warrant the additional increase in wiring pitch. Therefore, this solution is only considered in desperate situations. Increasing the spacing between the signal and return wires decreases the capacitance. However, this effect is negated by the increase in inductance resulting from the uncoupling of the signal-return pair, as shown in Figure 4.5.

There is another design option to obtain better transmission line performance. Using differential signaling in place of the single-ended method substantially improves the wire properties. In differential signaling, there are two output drivers that drive complementary signals over two separate signal wires. Guard bands enclose this arrangement. The presence of tightly coupled lines carrying current in opposite directions uses the proximity effect to our advantage. Figure 4.6 shows significantly lower resistance and inductance values as compared to single ended lines. There are several other advantages of using a differential pair over a single ended TL, such as:

- The signal is self-referenced
- It provides excellent common mode noise rejection
- Sense amplifiers at receiving end can have a higher gain
- More immunity to power supply noise

This work uses a differential transmission line with signal and return path width of 1.6μm and spacing of 0.4μm in the top metal layer of an IBM10sf 65nm BEOL stack for baseline simulations.

4.4 Circuit Design

So far, we have explored the wire properties of transmission lines. In this section, we assess the various circuit choices available to the designer and their advantages and shortcomings. All

circuits used in the simulations have been designed and simulated in HSPICE for the IBM10SF 65nm process.

4.4.1 Drivers

The two main classes of driver circuits are current mode and voltage mode drivers [47]. Current mode drivers use a Norton equivalent termination to produce output voltage. A constant current is driven at a predetermined value of output termination (usually 50 ohm). The easily controlled output impedance is a desirable trait for a TL driver, especially if the line is prone to reflections. Also, the dI/dt noise is kept to minimum levels due to the invariant nature of the current. Voltage mode drivers, on the other hand, use a Thevenin's equivalent termination to drive a constant output voltage. They are much more energy efficient than current mode drivers since they draw variable current. The large voltage swing at the receiving end enable better noise immunity and smaller receivers. Also, reflections in on-die transmission lines are a much less significant problem for contemporary technology nodes due to the inherent damping provided by the high resistance wires. We use the circuit shown in Figure 4.7 for our simulations. The circuit was adapted from [18].

Based on the data polarity, the predriver selects one of the four MOS drivers to send data on the line. Drivers A and B operate on opposite phases of a single clock, CK. The use of CMOS devices enables a high swing output. Transistor sizing is performed each time for different TL lengths and frequencies.

4.4.2 Receivers

A receiver must have a high gain in order to sense small signals at its input while also maintaining a high bandwidth to allow very fast signals to be correctly detected. A dynamic comparator as shown in Figure 4.8 is used for these simulations. The comparator operates in an interleaved fashion, similar to that of the driver. Transistor sizing is performed so that the receiver can detect at least a 400mV signal at its input.

4.4.3 SERDES

Transmission lines suffer from poor throughput normalized to wiring area due to their wide pitch. To offset this problem, they are operated as serial links at very high frequencies. A simple 4:1 serializer-deserializer system as shown in Figures 4.9 and 4.10 is used in this work. Multiplexers and D flip-flops are used for implementation. The clock generation circuitry is outside the scope of this work.

4.5 Evaluation

In this section, we evaluate the signal carrying characteristics of a transmission line link, complete with a driver and receiver.

4.5.1 Latency

The latency of a transmission line is comprised of two parts – the link latency and the latency introduced by the driver-receiver pair. As observed in Figure 4.11, the overall latency increases as line lengths increase. However, if we observe the latency per unit length, it becomes clear that the delay is dominated by the circuits. Therefore, to amortize the latency cost of driving the transmission line, longer lines must be preferred as long as signals can be reliably propagated along the TL by the line drivers.

4.5.2 Energy

The same argument holds true for the energy cost. In fact, due to the absence of intermediate repeaters, the only power consuming devices exist at the ends of the line. Figure 4.12 illustrates this phenomenon. This goes to show that transmission lines exhibit a different behavior as compared to RC wires. For example, doubling the line length does not double the latency and energy budgets of the link.

4.5.3 A different metric

Martin et.al. [48] introduce a metric to gauge the performance of a design. They show that for normal range of operation (not near threshold), the product of the energy (e) and the square of the time delay t, et^2 is independent of the voltage. This allows designers to optimize the circuit for either e or t, and adjust the voltage to obtain the desired trade-off. This makes the metric independent of constant field scaling. The metric is evaluated for the transmission line and represented in Figure 4.13. It is clear that the lower power-delay overhead provided by longer transmission lines improves the per unit length performance significantly.

4.5.4 Comparison with RC lines

Finally, we compare the energy and latency numbers for the transmission line example and a diffusive wire in an intermediate metal layer used for global signaling. Repeaters are added at regular intervals to maintain signal integrity and optimize the power $\times$ product. Figure 4.14 shows the comparison of the two wires for latency, energy, and the et^2 metric. The RC wires perform better for lower lengths, where the costs of the extra circuitry in transmission lines has not been amortized. For longer lengths, the extra energy consumption of repeaters and the inherent slow nature of RC

wires severely degrades its performance. The et^2 is constant for the diffusive wire due to the linear energy-length and latency-length relationship facilitated by the use of repeaters.

4.6 Summary

This chapter provides guidelines for the design and characterization of on-chip transmission lines. The simulation setup is described, and driver and receiver circuits are presented. Transmission lines are then compared against diffusive RC wires and their relative superiority is demonstrated for long distance communication.

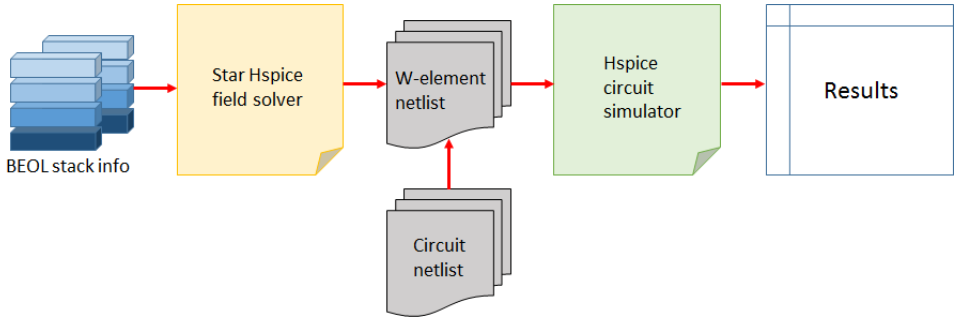


Figure 4.1: Characterization process flow

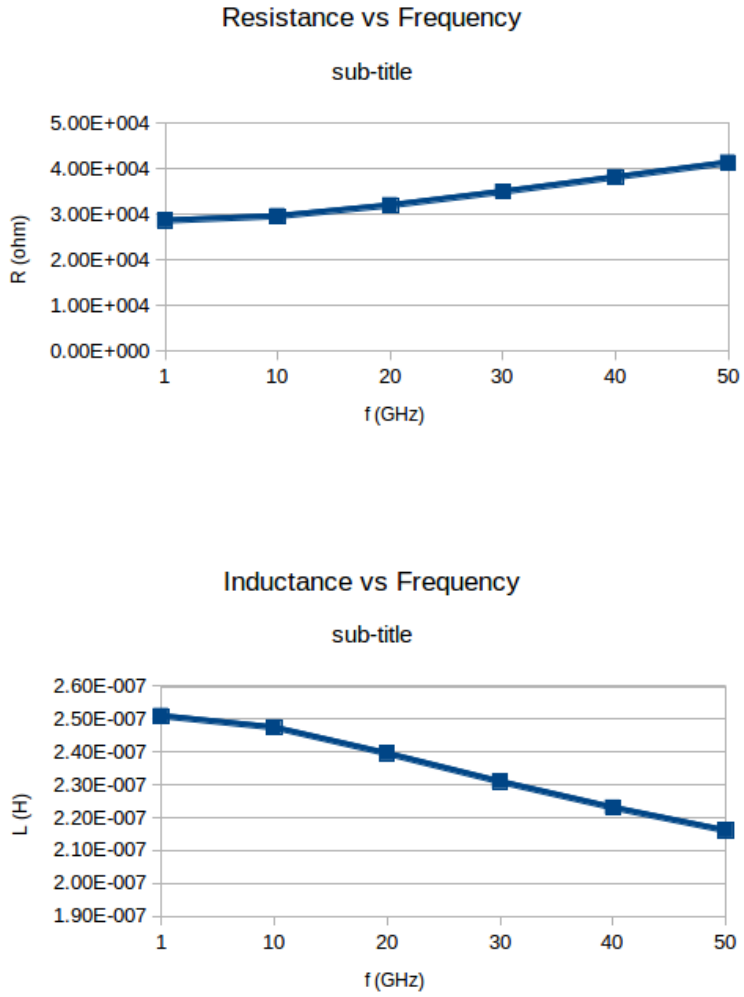


Figure 4.2: Proximity effect at high frequencies

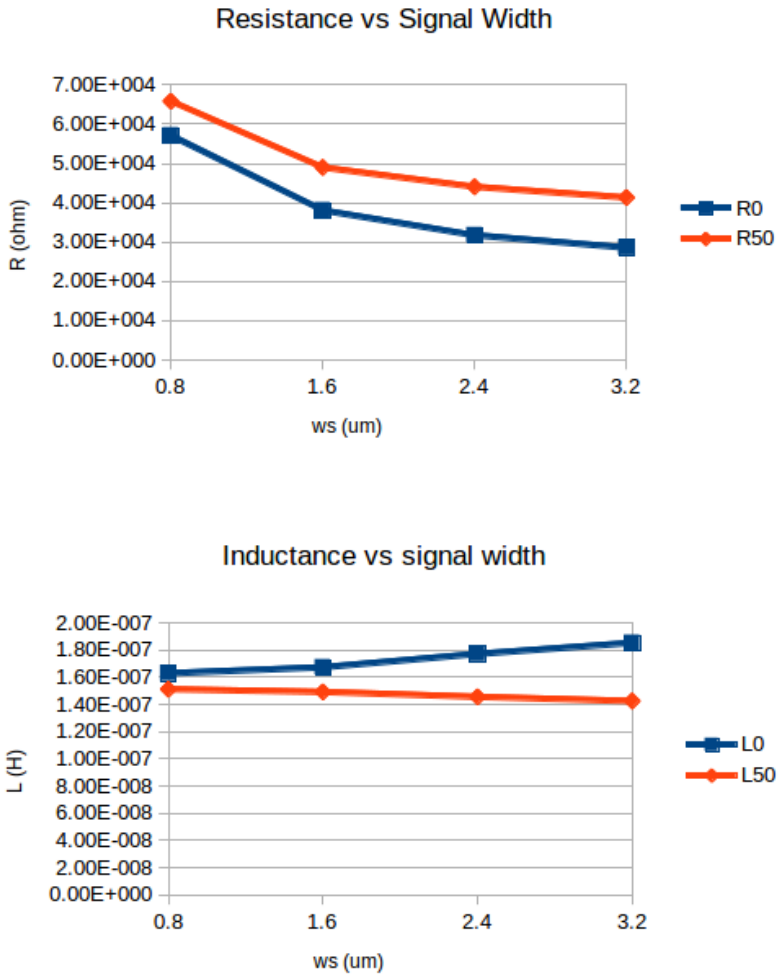


Figure 4.3: Effect of conductor width

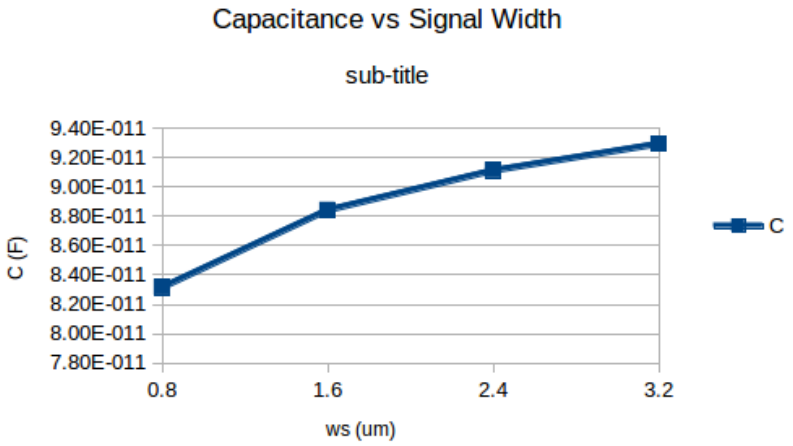


Figure 4.4: Capacitance limitation of increasing conductor width

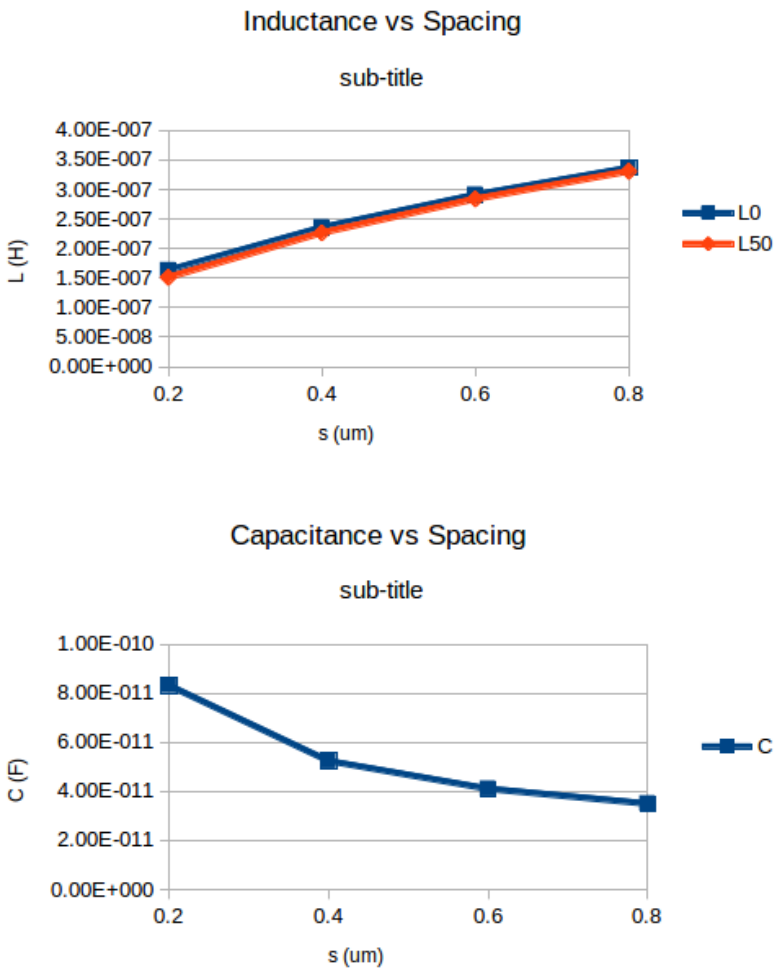


Figure 4.5: Implications of increasing the spacing

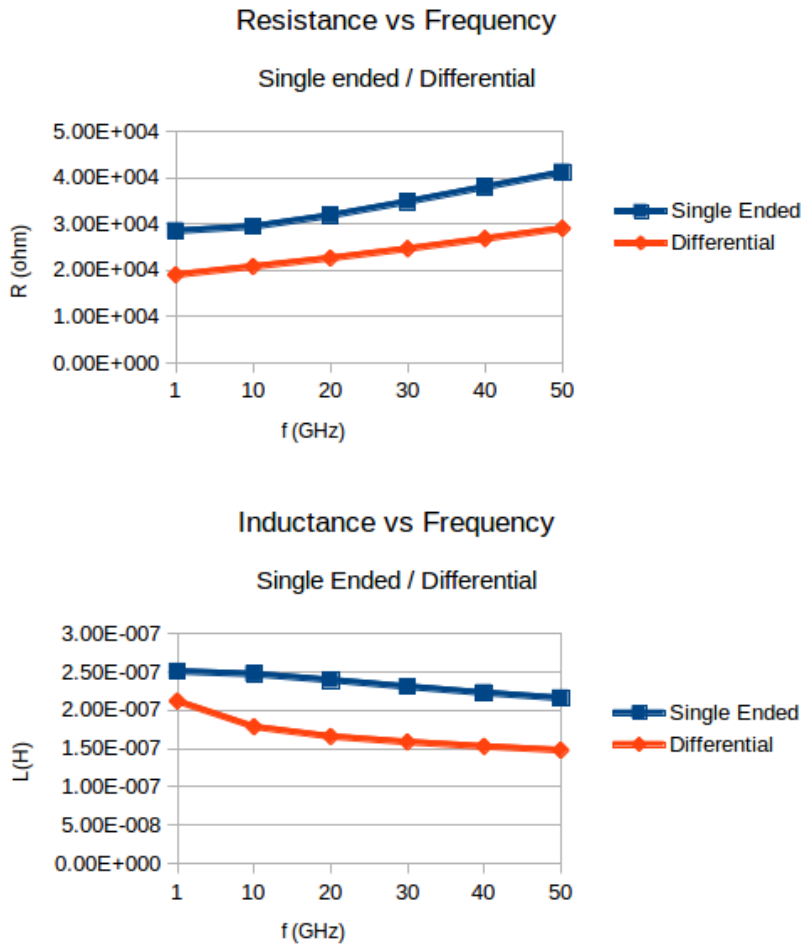


Figure 4.6: Differential vs single ended lines

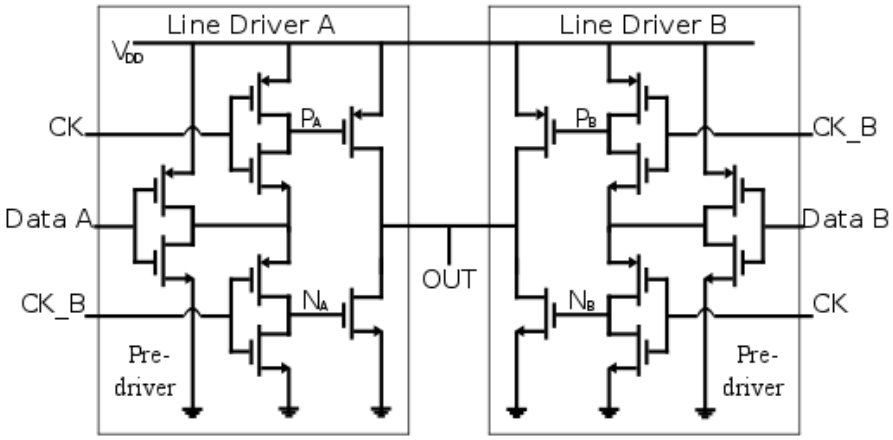


Figure 4.7: Voltage mode driver

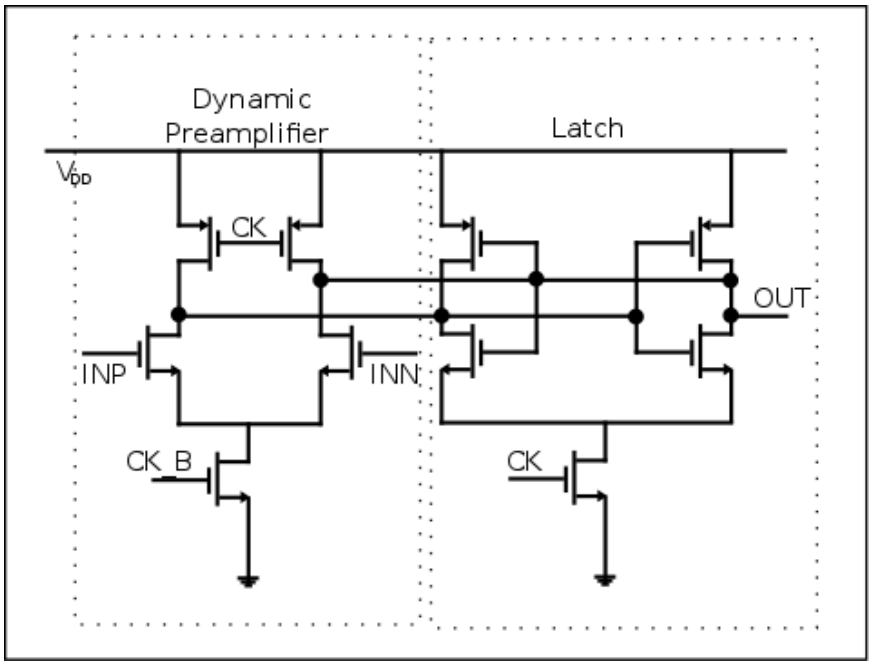


Figure 4.8: Comparator receiver

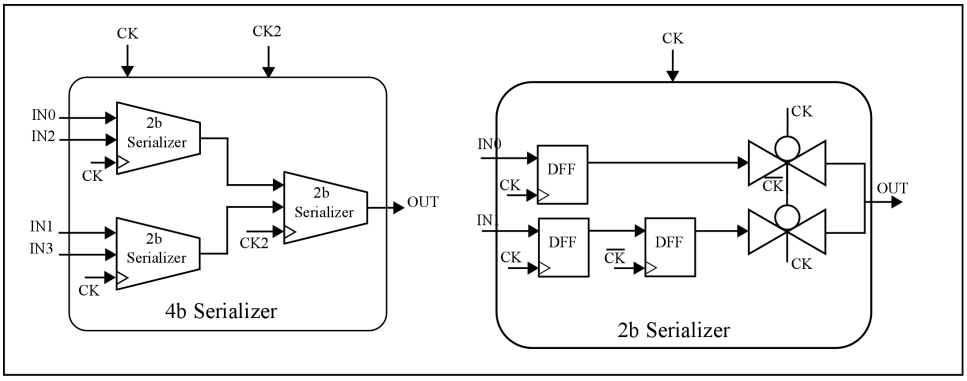


Figure 4.9: Serializer

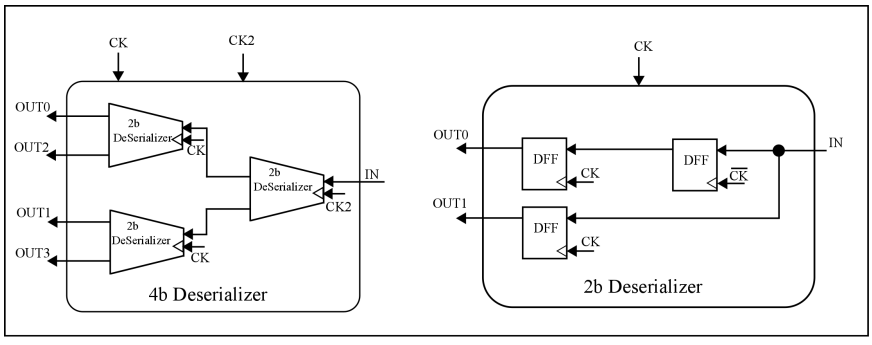


Figure 4.10: Deserializer

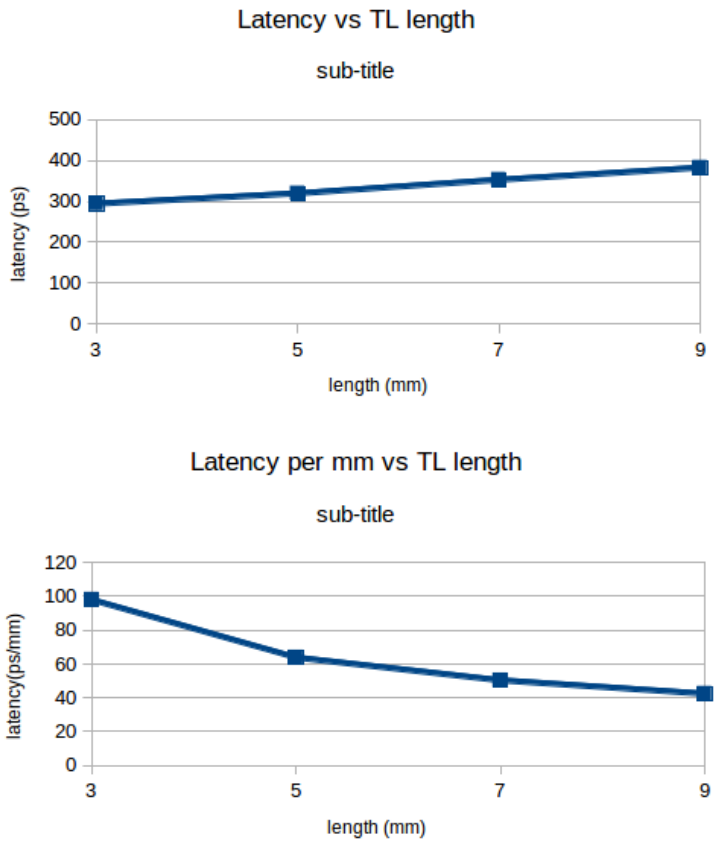


Figure 4.11: Latency of a TL link

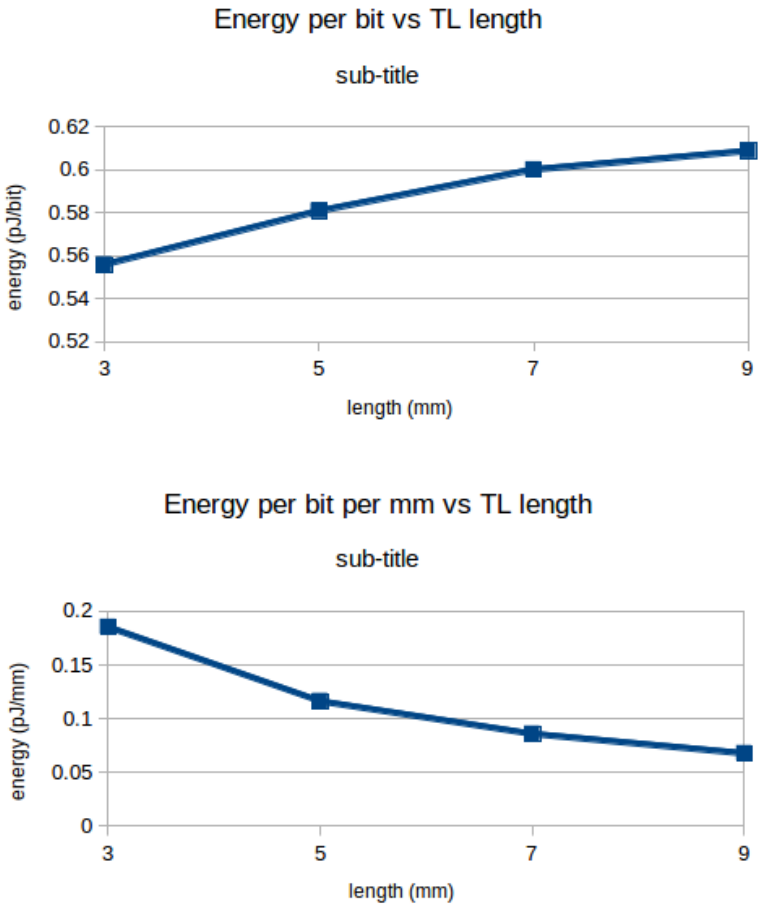


Figure 4.12: Energy of a TL link

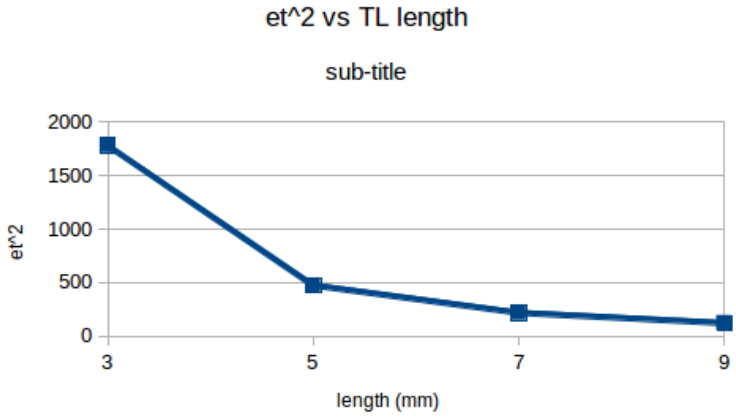


Figure 4.13: Energy-delay squared product of a TL link

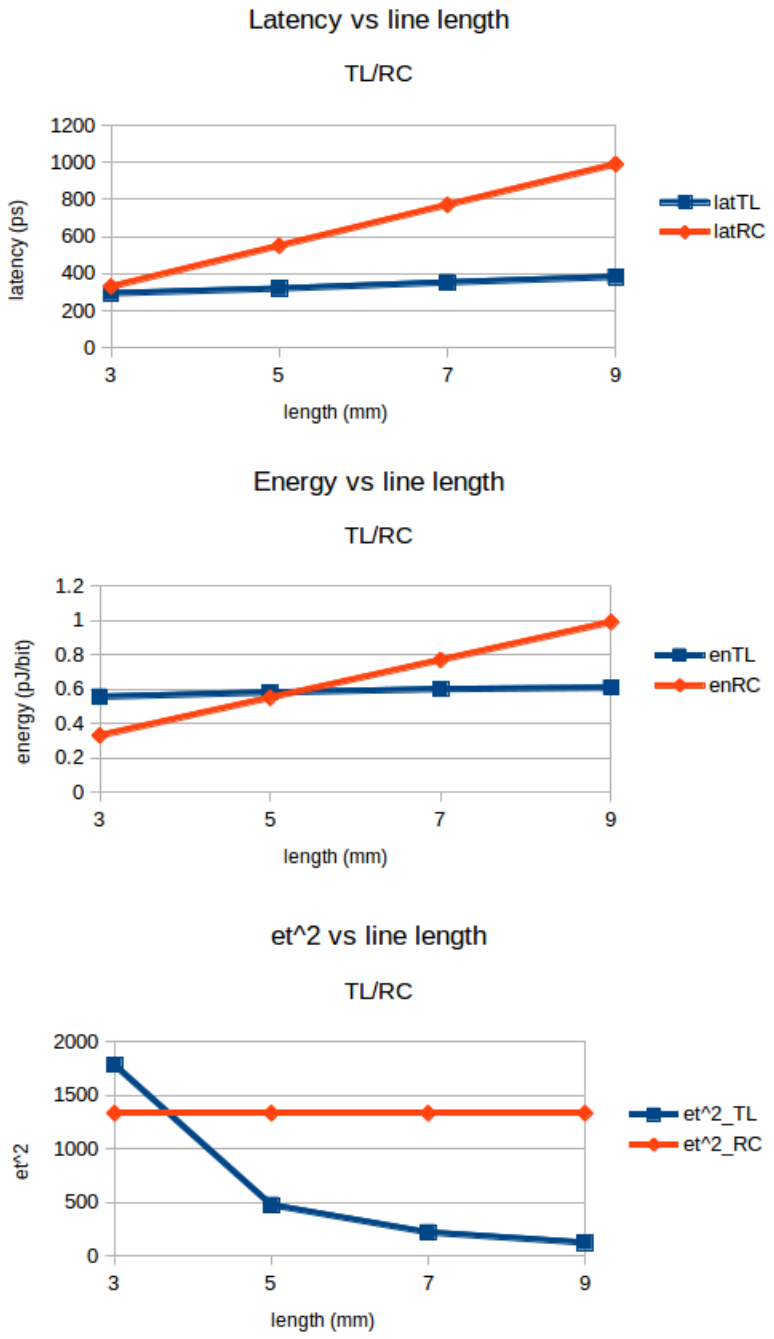


Figure 4.14: Comparing RC and TL

CHAPTER 5

COMPARISON OF CHANNEL PROTOCOLS

Use of unconventional low latency transmission line interconnects changes the timing assumptions in link design. Performance of both clocked and asynchronous communication protocols is dependent on the wire latency between stages. This chapter examines the power-performance characteristics of various protocols in the presence of high speed transmission line interconnects.

5.1 Motivation

Comparison of channel protocols for NoC links using transmission lines is a useful study in order to determine the best data transfer scheme for a given design constraint.

5.1.1 Scaling Effects

One of the major consequences of physical scaling and increased performance targets is the reversal of the cost of communication and computation. To put the magnitude of these effects in perspective, consider the delay difference scaling has produced for a transistor to calculate a logic function compared to the propagation of a signal over 1mm of interconnect. One can compare 10 scaling generations by examining the 1.0m and 35nm technology nodes. Native transistor delay (τ) in the 1.0μm process is 30ps, while propagation delay down a 1mm wire is a small fraction of the transistor delay at 1ps. This 30:1 delay relationship has been dramatically reversed in the 35nm node. Transistor delay is projected to be 1ps while wire delay for the same 1mm wire balloons to 250ps. This is nearly four orders of magnitude difference. The energy relation is equally stunning. In the 35nm process technology, energy expenditures necessary to propagate a 64b signal 2mm is approximately equal to performing a 64b floating point operation [3]. One of the major changes in design methodology brought about by scaling is the addition of pipeline stages in the interconnect. This communication pipelining results in some circuit and architecture advantages and overheads. Various communication methodologies have differing local and architectural performance characteristics. It is useful to model pipelines using various synchronous and asynchronous protocols and compare their power-performance relations [30].

5.1.2 Transmission Lines and Diffusive Wires

Transmission line interconnects transmit data at high bandwidths with low latency while consuming much less energy as compared to diffusive RC wires. There are fundamental differences in the technique of signal propagation between RC and RLC interconnects. Diffusive wires in modern process nodes have poor signal carrying properties. They have high resistance, which escalates the latency and diminishes the available bandwidth. Lesser supported bandwidth means that pipeline stages need to be added along the line to maintain throughput. Also, repeaters need to be introduced at regular intervals to limit signal distortion and maintain a linear delay-length relationship. This increases the power drawn by the wires, especially for long distances. On the other hand, transmission lines can support higher frequencies, and have better characteristics when they remain free of discontinuities, including vias. As seen in Figure 5.1, the absence of signal boosting repeaters and pipeline stages are an important change while utilizing TL signaling links over RC wires. Communication protocols have been modeled and evaluated extensively for long RC wires. In this work, we update the models to apply to transmission line channels and report the effects of using various synchronous and asynchronous communication protocols on these disruptive interconnects.

5.2 Overview

A quick background along with some definitions are provided in this section for the benefit of the reader.

5.2.1 Handshaking

Clock tree networks consisting of a low skew clock for a global synchronous system are inefficient in terms of area and power consumption. Given the heterogenous and multi-frequency nature of modern SoCs, imposing a single timing reference on the entire chip leaves a lot of performance on the table. On the other hand, there are multiple benefits of using asynchronous protocols for communication. Self-timed blocks can be designed to operate at optimal frequency and power without the shackles of a global clock. The inherent modularity of asynchronous circuit blocks enables higher design reuse, faster time to market, and proliferance of custom IP blocks. Also, asynchronous designs are more robust due to their adaptive nature. However, there is a bandwidth penalty due to the overhead introduced by handshaking. In synchronous design, data is normally transmitted every clock cycle between latches or flops. In asynchronous design, the clock signal is replaced by handshaking between neighboring latches to indicate that data is ready to be transmitted. Data can be represented as tokens flowing in conjunction with handshake channels, transparent to combinatorial circuits. This data-flow abstraction separates the circuit behavior from

its implementation details. Therefore, there may be multiple ways to implement each handshake protocol.

5.2.2 Protocols

Data transmission in asynchronous domains is based on handshake protocols (`req` and `ack`), allowing transactions to be dynamic based on data validity and space availability. Single-rail (bundled data) protocols, shown in Figure 5.2, use the same data path as clocked designs but employ latches rather than flip-flops for storage [49]. Data can also be transmitted using 1-of-N codes which encode data validity within data wires but use a shared acknowledgment. When data is not available, the communication links remain idle [50]. This is emulated in a synchronous design by applying clock gating, which passes a data valid bit with similar functionality to the asynchronous `req` signal indicating the validity of the data, allowing one to disable or "gate" the clock to the flops. This implicit clock gating that asynchronous design provides improves the energy efficiency of the data communication, especially when the sender is not ready to transfer data every clock cycle. The backward acknowledge (`ack`) signal in the asynchronous domain restricts the max cycle time in a design dependent way that ensures min-delay hold-time failures can not occur or are explicitly modeled.

Synchronous methodologies and bundled data protocols place restrictions on how and when data can change relative to the clock or handshake signals [51]. The data must have sufficient time to become valid before the clock or request signal enables its usage. The `ack` signal is used in asynchronous protocols to indicate the data has been received, and that new data can be transmitted. The protocols are maintained in a synchronous system with setup and hold requirements.

Most asynchronous protocols have backwards handshakes and return-to-zero (RZ) phases. Two-phase (or NRZ) protocols halve the number of transitions per handshake but often require more logic overhead and delay. Timing assumptions can be made using pulsed and source synchronous protocols to entirely remove the backward handshake path [52].

When controlling local data paths, the communication delays for control signals are small and the handshaking overhead can largely occur concurrently with the data function, hiding the overhead. However, for communication, the `req` and `ack` control signals are exposed to the same power and delay costs as the data because they must also propagate from sender to receiver. Thus propagation of the control lines is an expensive operation for most asynchronous communication protocols in both power and delay when compared to clock methodologies where the distribution costs can be shared amongst many logic blocks. This makes data communication the most *inefficient* application for asynchronous protocols and creates a worst-case comparison between synchronous

and asynchronous implementations.

Wire sizing and shielding can be used to somewhat mitigate control wire delay overhead by trading off increased area. Nonetheless, the handshaking significantly limits the throughput of asynchronous. For example, when the delays of the data and control wires are roughly the same, then a four-phase protocol can at best transmit data at a rate one-fourth the propagation time between the latches.

There are seven protocols studied in this work. Three synchronous ones – flopped and latched clocking, and source synchronous. The asynchronous family of protocols is represented by the 2-phase and 4-phase bundled data, dual rail delay insensitive design, and source asynchronous methodology introduced in an earlier chapter.

5.2.3 Metrics

We calculate throughput, latency, and energy cost of implementing clocked and unclocked communication protocols over a 5000μm long transmission line. The various parameters used in our models have been obtained using circuit simulations with HSPICE. Ideal pipelines would have no variation, and the flops would have zero latency and no overhead. This would allow a new data item to be propagated through these pipe stages every four gate delays. We compare the actual throughput and overheads of the protocols as calculated by closed-form equations developed in this work. In case of ideal pipelines there are no overheads associated with the memory elements, so the latency is the same as wire delay. However, with channel protocols, delay is introduced due to various factors such as set-up delay, added latency of the control logic, serialization costs, etc. The extra circuitry added due to the protocol based communication also adds to the energy costs. We compare the total energy of the pipelined system, which is the sum of the latch and the controller energy. Models are developed that are used to compare cycle time, latency and energy in a hierarchical manner. These models include most first order effects such as clock gating, coupling, process variation, voltage variations, data and bus activity factors.

5.2.4 Terminology

The timing of all fabricated circuit elements will be faster or slower than the scalar value of an "ideal" element due to process variations, capacitative coupling, and other effects. Variation from ideal devices and wires are considered an overhead in this work, whether it manifests itself as clock skew or device and wire delay variation.

Tables 5.1 and 5.2 show the parameters and their associated values that are used in this paper. All values come from simulation of the designs in 65nm technology using predictive spice models [53] [54]. The parameters in the top section of Table 5.1 model variations from ideal delays that

occur on wires and devices. First-order effects are modeled, including coupling, process variation, voltage and temperature variations (V_c, V_p, and V_{vt}, respectively).

Values in the right column of Table 5.1 are scalar delays. These delays are all relative to the fanout of 4 (FO4) of a typical inverter, for example worst case clock skew is assigned a scalar delay of 1 FO4. Similarly, all energy values are relative to the energy of a single repeater.

The transmission line link delay (T_c) is specified for three different values – 3, 5, and 7 mm. These values were calculated using SPICE simulations for a fixed drive strength and frequency. The T_{fsm} parameter in Table 5.1 is based on the worst case cycle time of the asynchronous controller. The total cycle time is averaged between the four (two) phases of the four (two) phase asynchronous controller. Different protocols exhibit different cycle time overheads. For example, 2-phase NRZ protocols generally require more complicated control with larger delays compared to 4-phase protocols. The T_{lat} parameter represent the forward propagation latency of the handshake control signals in the asynchronous protocols. Intuitively this represents the delay associated with propagation of the incoming request signal to outgoing request signal. T_{cfifo+} and $T_{latfifo+}$ represent the cycle time and forward latency of the tree FIFO designs used for the implementation of the SAS protocol. The E_{ctl} parameter represents the average energy per phase per bit for the control logic for the asynchronous protocols. The energy spent per bit during a transfer over a transmission line is denoted by E_{tl}. A 4:1 serialization ratio is chosen for the data bus implementation. Consequently, the total number of transmission lines is $W_b/4$. The energy overhead added by the SERDES is represented by E_{serdes}. The energy consumed by the flop / latch to transfer data the critical distance of a communication link is represented as the parameter E_{dataf} / E_{datal}. The average energy consumed per transition by the flop / latch due to switching of the clock is represented as parameter E_{clkf} / E_{clkl}. Similar to control delays, different asynchronous protocols exhibit different energy requirements. For a fair comparison T_{fsm}, T_{lat}, and E_{ctl} have been characterized through simulation for the different protocols and are listed in Table 5.3. The sizing and spacing of the control wires in the asynchronous protocols may be optimized differently than the data wires. The parameters O_{wd} allow a reduced control wire delay for a larger area O_{wa}.

The parameter A_b is the activity factor of the bus, or the percentage of clock cycles during which data is transmitted across the bus. Each flop is considered to be gated during idle cycles in the clocked protocols. A_w is the activity factor of data on the bus, or the probability that any bit will switch values for an average bus transaction.

5.3 Models

In this section, models for the cycle time, latency, and energy for different protocols under various operating conditions are presented. Tables 5.4, 5.5, and 5.6 presented in this section reflect these expressions.

5.3.1 Cycle Time

The minimum cycle time of a flop represented by equation C_{ff+} presented in Table 5.4 will be bounded by one of two conditions. First, the clock cycle cannot be shorter than twice the width of the minimum sized pulse $2T_{pw}$ that can be safely engineered and propagated. Otherwise, the delay through a stage will be the sum of the maximum communication delay from the source flop to the destination flop T_{c+}. Synchronous systems have additional overheads of setup time, clock skew and jitter $T_{su} + T_{skj}$. Upon arrival of the clock edge, the data must also propagate through the flop T_{cq+}. Setup, skew, and flop delays create the overhead for clocking.

Equation C_{l+} in Table 5.4 is the minimum cycle time for a synchronous latch protocol. Similar to the flop protocol the cycle time is bounded by one of the following two conditions, twice the width of the minimum sized pulse $2T_{pw}$ and the delay through a stage which is equal to the communication delay from the source latch to the destination latch $T_{c+} + 2T_{dql}$. Use of latches instead of flops helps in reducing clock skew and jitter overheads because of the time borrowing allowed between stages. However the data must propagate through two latches $2T_{dql}$ to account for the alternate phase control of the latches.

The datapath in a bundled data asynchronous communication is similar to the datapath in synchronous communication. Controllers generate the local clock signals for the latches instead of the global clock, as shown in Figure 5.3. The bundled data protocol can be implemented in both four-phase and two-phase handshaking protocols. The key timing assumption (also known as *bundling data constraint*) in a bundled data protocol is that the data should be valid before there is an associated transition on the request line. To satisfy this constraint the request signal needs to be delayed using a delay line such that this delay is greater than the worst case delay of the data plus the setup time of the latch.

Equation C_{bd2+} and C_{bd4+} in Table 5.4 gives the minimum cycle time for a two phase and four phase bundled data protocol respectively. The same controller is used for both the 4-phase and 2-phase designs shown in Figures 5.3 and 5.4. The key difference in these designs is the use of double edge triggered flip flops [55] rather than simple latches for 2-phase design. Parameter T_{fsm+} represents the cycle time of the controller divided by the number of phases. It is equal to 4.25FO4 for a 4-phase protocol and 4.78FO4 for 2 phase protocol. The parameter T_{ps} in Table 5.4 is the

worst case path separation or margin in terms of FO4 delays between the control and data paths. T_{cdel} is the number of gate delays that must be added to the control path to guarantee that we satisfy the bundling data constraint. The worst-case values must be taken assuming that, as pipe stages are composed, it is possible to have worst-case skew between data and control in all stages. This creates a safe margin for the control and data race at the expense of throughput.

The data path in a DI protocol is comprised of data encoded as 1-of-N rails where N wires are used to represent $log_2 N$ bits of data and an additional wire which acts as the acknowledgment signal. The most well known of this class of protocols is dual rail (1-of-2) which uses two data wires per data bit. Delay insensitive protocols are typically implemented using four-phase handshake protocols. Hence there are four iterations through the data acknowledge detection and propagation logic $4T_{fsm+}$. Figure 5.5 shows a 1 bit WCHB [56] controller implementing a 1-of-2 DI protocol handshake. The acknowledgment wire can be made wider to reduce delay on its transitions $2O_{wd}T_{c+}$.

Another mode of data communication is wave-pipelining [57] where several "waves", i.e. data signals, can concurrently propagate through a pipeline with proper timing separation between two consecutive waves. This timing separation is created by a "constructive clock skew" in conventional wave-pipelining, by using a separate timing reference in source synchronous protocols and a "fast" signal in an asynchronous "surfing" protocols [58][59]. The timing reference signal is routed in parallel with the data. A novel characteristic of asynchronous surfing is that when logic blocks receive the fast reference pulse their computation delay drops, thus creating a surfing effect wherein events are bound in close proximity with the pulse on fast reverence signal.

The key to the performance of any wave-pipelining method is the time separation of waves which defines their cycle time. In this paper, we have explicitly modeled the cycle time of source-synchronous communication with equation C_{ss+} in Table 5.4. In particular, the cycle time is lower bounded by three terms: the minimum pulse width that can propagate through a critical distance; the communication delay of the data through a repeated wire; and the propagation delay of the clock signal plus the delay through controller which generates the enable signal for the latch. In the source synchronous models used here, multiple values are not propagated between flop stages.

As we have seen earlier, the bandwidth of a SAS protocol is independent of the wire delay. As long as the sender FIFO can accept tokens, the system can keep sending data along the channel. Since the wire latency of a transmission line interconnect is lesser than the cycle time of the FIFO used in the experiment, no wavepipelining is required. Therefore, C_{sas+} is limited only by the FIFO cycle time, T_{cfifo+}.

5.3.2 Latency

For clocked communication systems, the latency associated with data transfer per stage is the same as the cycle time. This is reflected in Table 5.5. Since there is only one pipeline stage in our transmission line experiment, the total latency is equal to the cycle time. Latency in asynchronous communication protocols is the sum of the wire delay and the controller delay. The overhead associated with bundled data asynchronous designs comes from two sources. The first source is the delay associated with the controller used to generate local clock signals. The second is the extra delay margin inserted in the request line as shown in Figure 5.3 to satisfy bundling data constraint. In DI protocols, no extra margins are required for setup to the latches. Also, the use of fast domino logic used in the data path leads to faster propagation of data compared to latch / flop designs. A data token in a SAS protocol has to travel across the FIFO at the receiving end. Therefore, the total latency is the sum of the TL communication latency T_{c+} and the forward latency of the receiver FIFO $T_{latfifo+}$.

5.3.3 Energy

The energy consumed by the flop / latch to transfer data the critical distance of a communication link is represented as the parameter E_{datf} / E_{datl} in Table 5.6. The average energy consumed per transition by the flop / latch due to switching of the clock is represented as parameter E_{clkf} / E_{clkl}. The average energy per transaction for clock gating of a flop E_{gf} / latch E_{gl} is calculated as a single transition on the clock driver, which is one-fourth the clock load of the flop plus energy into the gating NAND for the average number of idle bus cycles per transaction. The average energy per transaction consumed by the clock distribution network is given as $\frac{1}{A_b} 2 P_{dc} E_{clktree}$ where $\frac{1}{A_b}$ accounts for the amount of energy consumed by the clock distribution network when the data bus is idle.

The energy consumed per transaction of a single flop / latch pipeline stage can be attributed to two factors: the energy consumed by the memory element (flop / latch), and energy of the repeaters in that pipeline stage. The energy consumed by a flop is $\frac{W_b}{4}(2(E_{clkf} + E_{gf}) + A_w E_{dataf})$ and the energy consumed by the repeaters $\frac{W_b}{4} A_w (E_{tl} + E_{serdes})$ to drive the data. For the case of latch based communication the energy consumed by the latches is $\frac{W_b}{4}(4(E_{clkl} + E_{gl}) + A_w(2E_{datal}))$ due to two latches per pipeline stage. Note the 4 in the denominator to indicate a 4:1 serialization scheme.

Equations E_{4e} and E_{2e} in Table 5.6 calculates the energy per pipeline stages for 4-phase and 2-phase bundled data protocols respectively. The energy required by the asynchronous controller per phase is characterized as E_{ctl}. This parameter represents the energy consumed in transmitting the control signals (req and ack) a critical distance along the communication link. The parameter E_{clkdef} in Table 5.1 represent the average energy consumed per transition by a double edge triggered

flip flop shown in Figure 5.4.

The energy consumed per transaction per pipeline stage is the sum of the controller energy and the energy consumed for data transfer through latches. The controller energy for the 4-phase protocol is equal to $4*(E_{ctl}+E_{tl})$ and $2*(E_{ctl}+E_{tl})$ for the 2-phase protocol. The energy consumed by the latch in a 4-phase protocol is calculated as $\frac{W_b}{4}(2E_{clkl}+A_w(E_{datal}+E_{tl}+E_{serdes}))$ and for a 2-phase protocol energy is calculated as $\frac{W_b}{4}(E_{clkdef}+A_w(E_{datal}+E_{tl}+E_{serdes}))$.

Dual Rail protocol energy is calculated in similar fashion and is indicated in Table 5.6. For the SAS protocol, the sum of the FIFO energies at the sender and the receiver is taken into account for calculations.

5.4 Comparisons

Using parameter values from the IBM 65nm process, we evaluate the model expressions to obtain the throughput, latency, and energy values for different protocols. All values are normalized to a single FO4 repeater.

5.4.1 Protocol Comparison

We compare various pipeline protocols to determine the optimal choice for long distance communication over transmission lines in Figures 5.6, 5.7, and 5.8.

It is clear from Figure 5.6 that the asynchronous protocols with acknowledgment are the least efficient for communication, with the 4-phase protocols being the worst. The most efficient protocols are the clocked and source-synchronous protocols. The two-phase asynchronous protocol shows significantly better performance due to the reduction in control transitions propagated between sender and receiver. The source synchronous protocol performs well because its request is propagated as a pulse and no acknowledgment is explicitly included. This protocol is marginally slower compared to synchronous protocols largely due to the conservative margin in the delay elements. In a real design, one would expect the source-synchronous circuit to out-perform the synchronous design due to its adaptive nature. SAS also performs better than other asynchronous methods because its throughput is wire-delay agnostic.

Figure 5.7 indicates that the DI protocol provides the lowest latency of all the protocols. This can be attributed to two properties of these circuits. First, since the data coding is delay-insensitive in nature no extra margins are required for setup to the latches. Second, the use of fast domino logic used in the data path leads to faster propagation of data compared to latch / flop designs. The SAS protocol suffers from a slightly higher latency due to the presence of large FIFO buffers in its forward path, albeit the effect is slightly mitigated by using non-linear FIFO arrangements like the parallel FIFO, or in our case, the tree FIFO.

In a TL signaling regime, wire energy over long distances is a smaller fraction of the total communication link energy due to the higher frequency of operation coupled with the lack of signal repeaters. The asynchronous dual rail protocol has a much poorer power profile due to its activity factor. We see in Figure 5.8 that the bundled data protocols perform much better. In a clocked scheme, energy required to correctly distribute the clock tree with low skew is an energy intensive operation with a heavy switching profile. Also, energy is required in the gating logic even during idle phases, while asynchronous protocols provides ideal clock gating at no extra cost.

5.4.2 Diffusive Wires

We now evaluate the effect of using low latency, low energy transmission lines for long range communication by comparing the metrics to an equivalent 5mm RC line consisting of a single pipeline stage, but augmented with repeaters for signal boosting [30]. Obviously, without intermediate pipeline stages, this experiment yields pessimistic numbers for the throughput metric. However, skipping intermediate pipeline stages means that the associated energy and latency overheads are drastcally reduced, especially for long wires. Figure 5.9 shows over 60% improvement in bandwidth when migrating from an RC interconnect to a transmission line for a 2-phase bundled data protocol. Other protocols show similar trends. The exception is the SAS protocol which has a throughput that is independent of wire delays, and thus remains the same. Therefore, SAS protocol provides the same high throughput irrespective of the physical medium of signal propagation. The latency comparison in Figure 5.10 shows the difference between the two channels. Figure 5.11 shows how communication energy per bit is reduced by using transmission lines. There is a higher variance in latency in the case of transmission lines which serves to display the difference between the various protocols. This effect gets masked by the higher contribution of wire delay to total latency in diffusive RC wires. Transmission lines make energy efficient interconnects.

5.4.3 Wirelength

So far, we have investigated the case of a single wire length (5mm). We now demonstrate the effect of changing the wire length from 3mm to 7mm on both transmission lines and RC wires for different protocols. In Figure 5.12, we map the percentage change in the channel bandwidth when the length is increased from 3mm to 7mm for a single pipeline stage link. The blue squares reflect the TL data, while the orange kites show the corresponding RC wire data. It can be seen that if the line length is increased without adding intermediate pipeline stages, the throughput reduces drastically (up to 2$\times$) for RC channels. The sole exception is the SAS protocol, which as mentioned earlier, is agnostic to wire delay. On the other hand the transmission line system is a lot more impervious to this change (maximum 4.2%). This is due to the higher contribution of wire delay

to cycle time in case of diffusive wire links. As seen from Figures 5.13, 5.14, and 5.15, the transmission line system is almost insensitive to the change in wire delay. Similar arguments can be made for the total link energy, as seen in Figure 5.15. This means that scaling long links is much easier with transmission lines than RC wires as the overall effect on the communication channel performance is modest at best.

5.5 Summary

In this chapter, we have presented parameterized first order models for communication protocols over a transmission line. Comparisons between different protocols have been made and optimal methodologies for each metric has been identified. Across the board, SAS performs better than both synchronous and asynchronous protocols. Also, we observe that transmission lines have significantly better communication characteristics as compared to RC wires. For high performance, low energy, scalable data transfer, we recommend using SAS along with transmission lines.

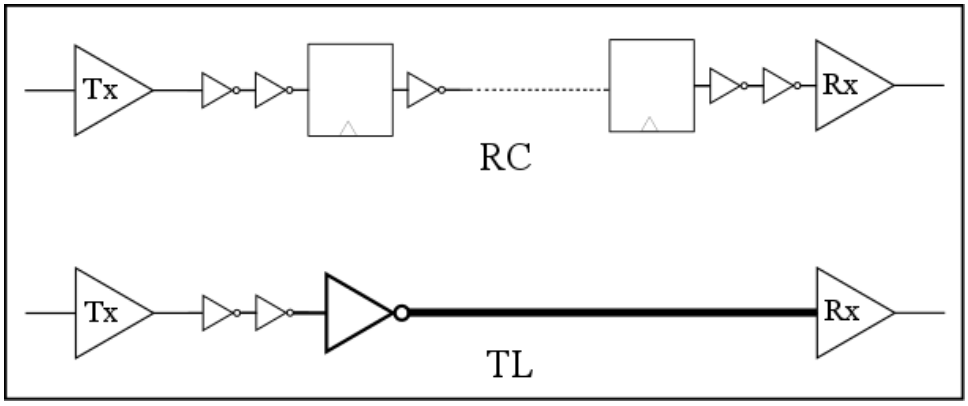
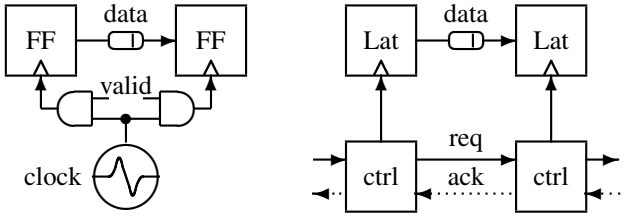


Figure 5.1: No repeaters in TL channels



(a) Synchronous (b) Asynchronous

Figure 5.2: Synchronous and asynchronous topologies

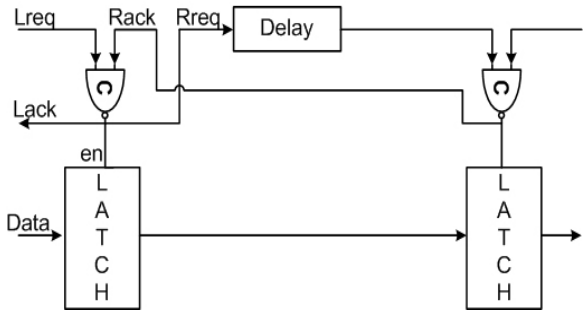


Figure 5.3: Simulated implementation for bundled data 4-phase protocol

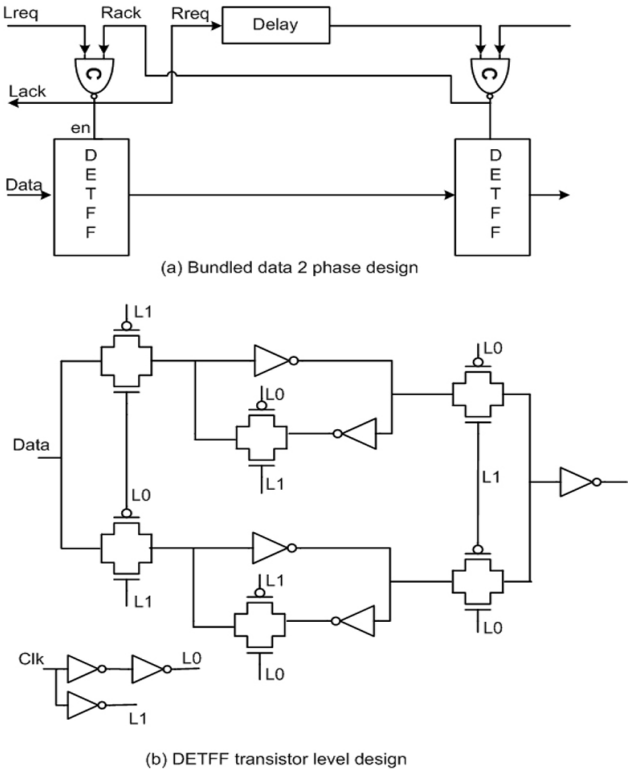


Figure 5.4: Simulated implementation for bundled data 2-phase protocol

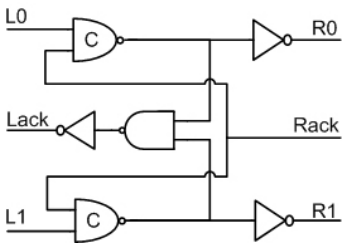


Figure 5.5: Simulated DI 1-of-2 protocol implementation

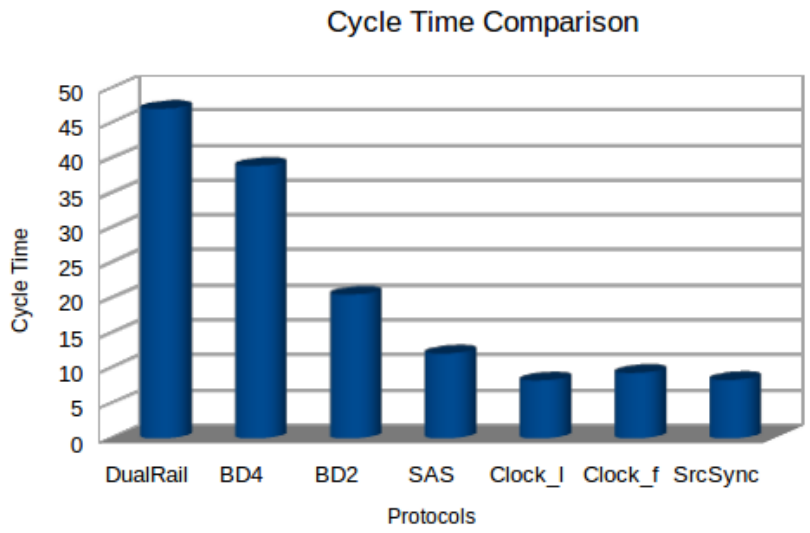


Figure 5.6: Cycle time for various protocols

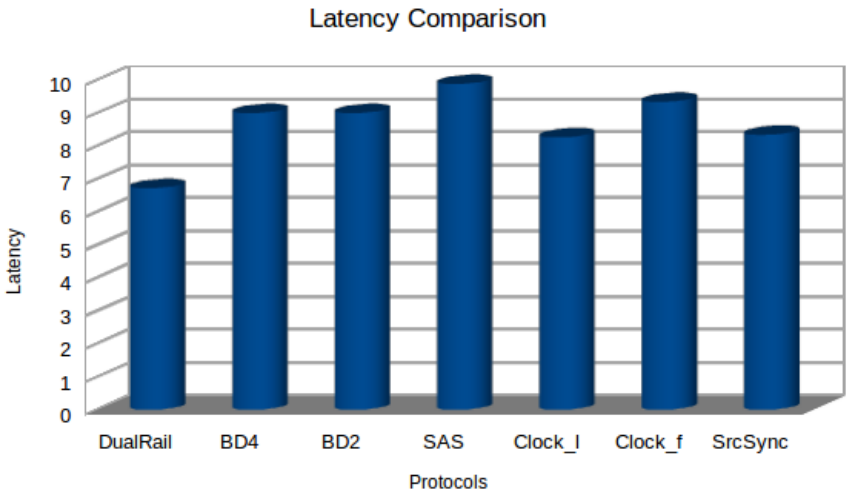


Figure 5.7: Latency for various protocols

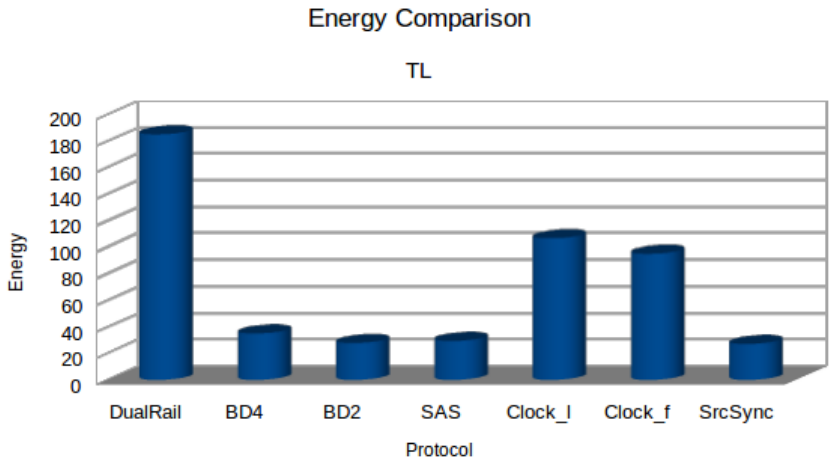


Figure 5.8: Energy for various protocols

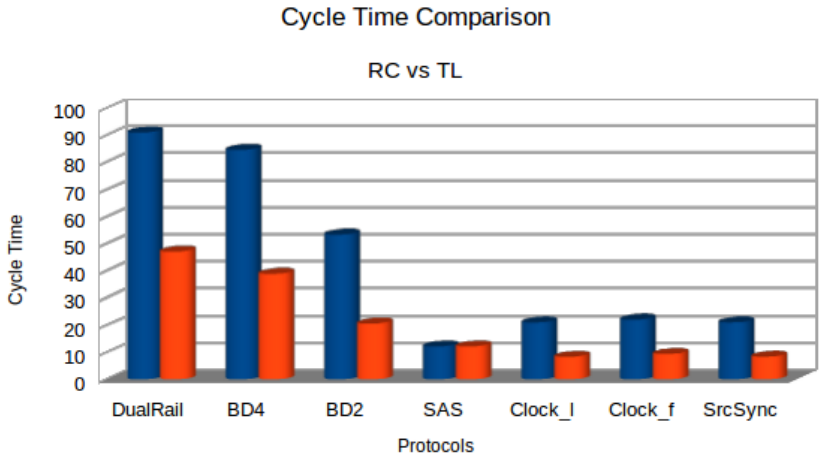


Figure 5.9: Cycle time RC vs TL

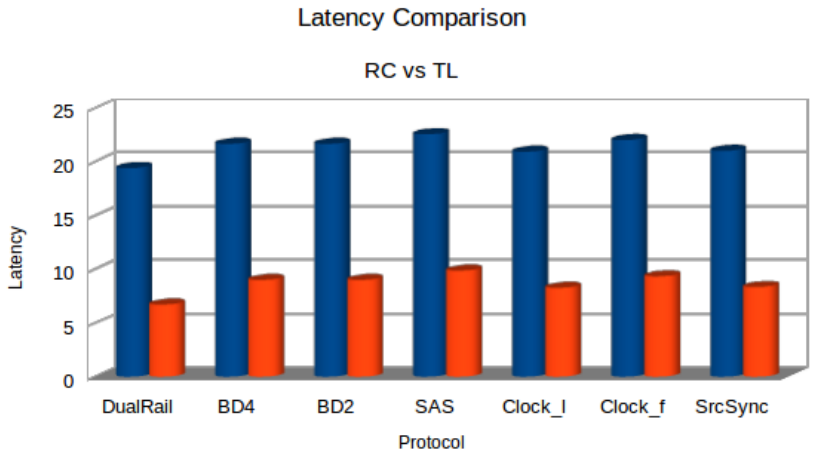


Figure 5.10: Latency RC vs TL

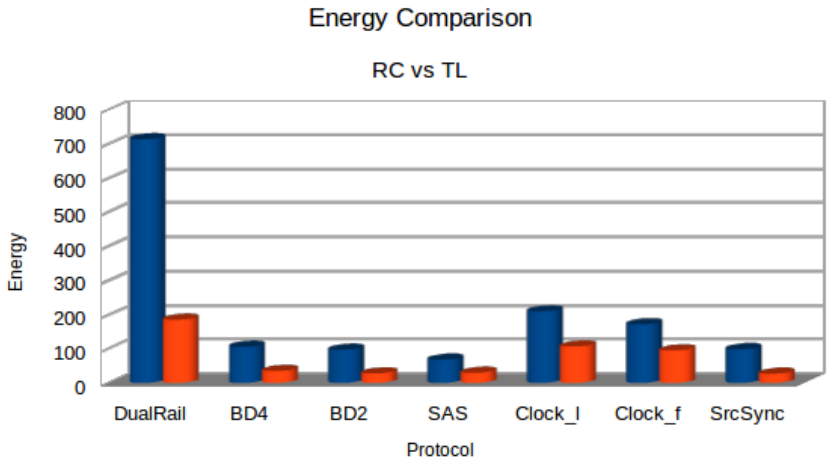


Figure 5.11: Energy RC vs TL

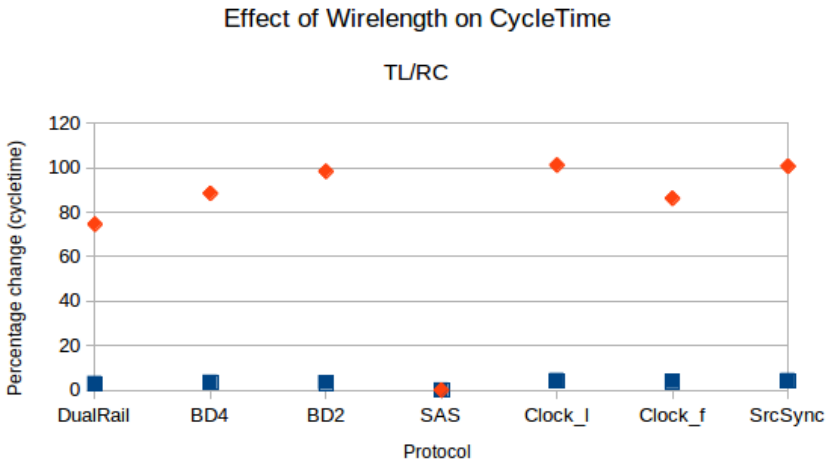


Figure 5.12: Wirelength effect on cycle time

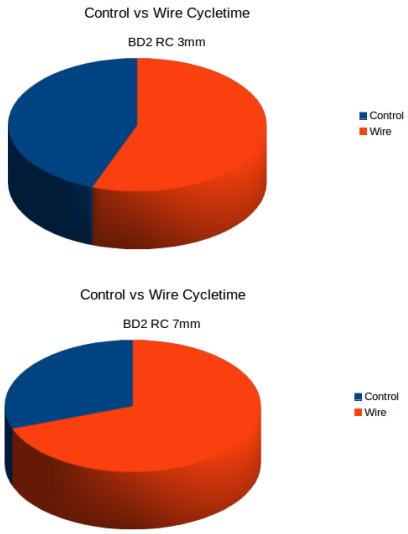


Figure 5.13: Contribution of wire delay to cycle time (RC)

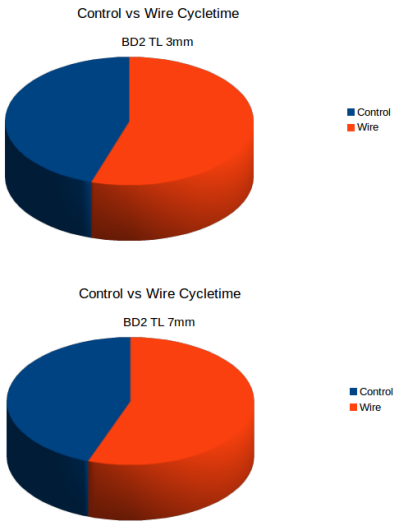


Figure 5.14: Contribution of wire delay to cycle time (TL)

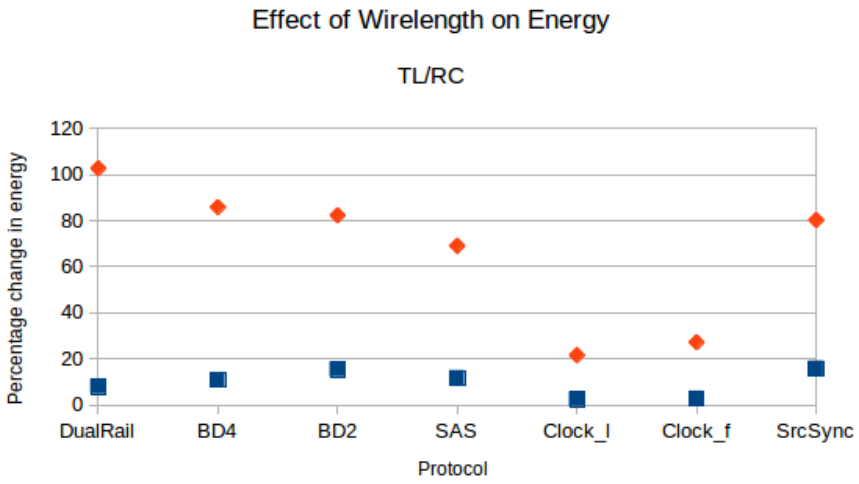


Figure 5.15: Wirelength effect on energy

Table 5.1: Parameter variables and derivatives.

Name	Description	FO4 Value
V_c	Cross coupling variation of wire	0.4
V_p	Process variation	0.16
V_{vt}	Voltage & temperature variation	0.12
T_{cq}	Delay from clock to output	1.8
T_{sul}	Latch setup time	0.48
T_{suf}	Flop setup time	0.96
T_{skj}	Clock skew and jitter	1
T_h	Hold time	0.25
T_{dqf}	Flop data-to-output delay	2.8
T_{dql}	Latch data-to-output delay	1.4
T_{pw}	Minimum pulse width that can be propagated through a critical distance	2.2
T_c	TL Channel Delay	5.0 – 6.3
T_{c+}	$T_c(1+\frac{V_c}{4})$	5.5 – 7.0
T_{c-}	$T_c(1-\frac{V_c}{4})$	4.5 – 5.7
T_{fsm+}	Controller delay per phase	2.55 – 6.64
T_{fsm-}	$T_{fsm+}(1-\frac{V_p}{2})$	2.34 – 6.10
T_{lat+}	Forward latency of async controller	1.18 – 3.95
T_{cfifo+}	Cycle time of 0-bit tree FIFO	12.12
$T_{latfifo+}$	Forward latency of 32-bit tree FIFO	4.33
E_{ctl}	Controller energy per phase per bit	0.072 – 0.90
E_{tl}	Energy of a TL link	1.32 – 2.12
E_{serdes}	Energy of a SERDES apparatus	2.15
E_{datsas}	Energy of a SAS data link	1.42
E_{ctlsas}	Energy of a SAS control logic	5.0
E_{rep}	Energy of a single repeater	1
E_{datf}	Data energy through flop	2.02
E_{datl}	Data energy through latch	1.78
E_{clkf}	Clock energy for flop	0.22
E_{clkl}	Clock energy for latch	0.2
E_{clkdef}	Clock energy for DETFF	0.42

Table 5.2: Operating parameters

Name	Equation	Value
A_w	Activity factor of bus wires	0.18
O_{wd}	Control wire delay optimization	0.85
O_{wa}	Control wire area optimization	1.85
W_b	Width of bus	32
$E_{clktree}$	Clock tree energy per transition	16.84
P_{dc}	% of clock tree energy for communication	10%

Table 5.3: Asynchronous control parameters from 65nm SPICE simulations of the designs

Name	$\mathbf{T}_{fsm}$	$\mathbf{T}_{lat}$	$\mathbf{E}_{ctl}$
4-phase bundled data	4.28	3.45	0.075
2-phase bundled data	4.77	3.45	0.072
DI 1-of-2	6.64	1.18	0.90

Table 5.4: Cycle time equations measured in FO4 delays.

Name		equation	description
T_{ps}	$\leq$	$(T_{fsm+} + T_{cq+} + T_{c+} + T_{su}) - (T_{fsm+} + T_{c-} + T_{fsm-})$	Handshake to data margin
T_{cdel}	$\leq$	$(1 + \frac{V_p}{2})\max(0, T_{ps}) - (1 - \frac{V_p}{2})\max(0, T_{ps})$	Robust delay element size
C_{ff+}	$\geq$	$\max(2T_{pw}, (T_{c+} + T_{su} + T_{skj} + T_{cq+}))$	Clk_flop throughput
C_{l+}	$\geq$	$\max(2T_{pw}, T_{c+} + 2T_{dql})$	Clk_latch throughput
C_{di+}	$\leq$	$4T_{fsm+} + 2T_{c+} + 2O_{wd}T_{c+}$	2-rail and 1-of-4 DI
C_{bd2+}	$\leq$	$2T_{fsm+} + T_{c+} + T_{cdel} + O_{wd}T_{c+}$	2-phase bundled data cycle time
C_{bd4+}	$\leq$	$4T_{fsm+} + 2T_{c+} + T_{cdel} + 2O_{wd}T_{c+}$	4-phase bundled data cycle time
C_{ss+}	$\leq$	$\max(2T_{pw}, (T_{cq+} + T_{c+} + T_{su}), (T_{c+} + T_{fsm+}))$	src-sync cycle time
C_{sas+}	$\leq$	T_{cfifo+}	SAS

Table 5.5: Latency per stage measured in FO4 delays.

Name		equation	description
L_{ff+}	$\geq$	$\max(2T_{pw}, (T_{c+} + T_{su} + T_{skj} + T_{cq+}))$	Clk_flop latency
L_{l+}	$\geq$	$\max(2T_{pw}, T_{cc+} + 2T_{dql})$	Clk_latch latency
L_{di+}	$\leq$	$T_{lat+} + T_{c+}$	2-rail and 1-of-4 DI latency
L_{bd2+}	$\leq$	$T_{lat+} + T_{c+} + T_{cdel}$	2-phase bundled data latency
L_{bd4+}	$\leq$	$T_{lat+} + T_{c+} + T_{cdel}$	4-phase bundled data latency
L_{ss+}	$\leq$	$\max(2T_{pw}, (T_{cq+} + T_{c+} + T_{su}), T_{c+} + T_{fsm+})$	src-sync latency
L_{sas+}	$\leq$	$T_{latfifo+} + T_{c+}$	SAS latency

Table 5.6: Models for average energy per transaction per pipe stage.

Name	equation	description
E_{gf}	$\frac{1}{A_b}\frac{E_{clkf}}{12}+\frac{E_{clkf}}{4}$	Flop clock gate energy
E_{gl}	$\frac{1}{A_b}\frac{E_{clkl}}{12}+\frac{E_{clkl}}{4}$	Latch clock gate energy
E_{fe}	$\frac{1}{A_b}2P_{dc}E_{clktree}+\frac{W_b}{4}(2(E_{gf}+E_{clkf})+A_w(E_{datf}+E_{tl}+E_{serdes}))$	Clk_flop energy
E_{le}	$\frac{1}{A_b}2P_{dc}E_{clktree}+\frac{W_b}{4}(4(E_{gl}+E_{clkl})+A_w(2E_{datl}+E_{tl}+E_{serdes}))$	Clk_latch energy
E_{4e}	$4(E_{ctl}+E_{tl})$ $+\frac{W_b}{4}(2E_{clkl}+A_w(E_{datl}+E_{tl}+E_{serdes}))$	4-phase bundled data energy
E_{2e}	$2(E_{ctl}+E_{tl})$ $+\frac{W_b}{4}(E_{clkdef}+A_w(E_{datl}+E_{tl}+E_{serdes}))$	2-phase bundled data energy
E_{ss}	$2(E_{ctl}+E_{tl})$ $+\frac{W_b}{4}(2E_{clkf}+A_w(E_{datl}+E_{tl}+E_{serdes}))$	src_sync energy
E_{di2}	$\frac{W_b}{4}(4E_{ctl}+2E_{tl})+2E_{tl}+2E_{serdes}$	DI 1-of-2 energy
E_{sas}	$(E_{ctlsas}+E_{tl}+\frac{W_b}{4}(E_{ctlsas}+A_w(E_{tl}+E_{serdes})))$	SAS energy

CHAPTER 6

A NETWORK-ON-CHIP EXAMPLE

This chapter provides an implementation of the techniques and models presented so far in a Network-on-Chip design.

6.1 Introduction

Implementation of high performance and low power on chip transmission lines requires careful design, but they have significant potential uses. In this work, we report on the application of transmission lines to improve the power and performance of Network-on-Chip (NoC) applications.

Process scaling supports the implementation of many processing cores on a single planar die. Distributed memory multiprocessors are an important class of such multiprocessor designs [60]. These systems consist of arrays of a local microprocessor and associated memory, which communicate via a mesh-based NoC communication fabric. The size of the array is defined by n, the number of processors on each edge of the array. The planar nature of the die maps well to regularly spaced two dimensional processing and NoC arrays such as a square 8×8 array containing 64 processor and memory banks. Each node will communicate with its four adjacent nodes in the array.

The hop count is the number of routers a message must traverse between communicating source and destination nodes. The worst case hop counts for a planar mesh unconnected at the periphery is $2(n-1)$, an example shown in Figure 6.1(a). The edges of an array can be connected by adding *wrap* wires, as shown in Figure 6.1(b). This creates a three dimensional interconnect where the wrap wires create a toroidal NoC structure. The wrap wires can be twisted to reduce the worst case hop count, as shown in Figure 6.1(d). Adding wrap lines can reduce the worst case hop count between any two nodes on a mesh by a factor of two to $n-1$

While 3D torus routers reduce the worst case hop count, they are difficult structures to map onto a two dimensional plane. Simply connecting the edges of a planar array of nodes as shown in Figure 6.1(b) results in short local interconnect between nodes on the plane, but long wrap wires that are $n-1$ times longer than the in-plane connections. This results in two very different delay values depending on which type of interconnect is part of the path, be it local or wrap lines. One

solution is to create a "folded" torus that is mapped to the planar chip, shown in Figure 6.1(c). This creates communication links that are all similar in length, delay, and energy by penalizing the short links.

In this work, we introduce a new approach to designing mesh based routers which can create energy efficient networks based on two types of interconnect: diffusive wires and transmission lines. We employ the traditional mesh based routing core with short interconnect between each core implemented with traditional diffusive wires. The long wrap lines are implemented with transmission lines, which gain their power and performance advantage only after approximately 1-2mm of wire length. We call our design a Transmission line Enabled Clockless Network On chip (TECNO), which consists of 64 routers connected as part of an 8×8 tiled system.

6.2 Related Work

6.2.1 On-chip Networks

Single chip complexity is on the rise as we transition from the multi-core to the many-core era. The network fabric must be able to effectively handle multiple processing elements (PEs) communicating among each other with high fidelity while consuming little energy and offering low latency. The heterogeneous nature of modern chips with multiple clock domains and voltage islands further compounds the problem. Networks-on-chip have been studied in great detail in the past and there is adequate understanding about NoC implementation choices such as topology, routing techniques, switching methods, flow control, etc. Most silicon implemented NoC designs prefer mesh topologies due to its simplicity, ease of physical mapping, and use of short wires [61], [62], [63]. However, they suffer from having a larger hop count, latency and lower path diversity. Designers have tried to solve this problem by using high radix routers to build complicated topologies such as flattened butterfly, folded clos, and dragonfly [23]. While these hierarchical topologies work very well as the underlying networks in supercomputers, they are expensive to implement in a power constrained system due to the complexity associated with the routers. Also, the long links used in low diameter topologies result in additional latency and energy cost. A high level study of replacing long diffusive wires with alternative equalized interconnects on low diameter networks has been studied [64]. Networks for nanophotonic technologies have also been studied [65], [66]. In contrast, our design utilizes the regularity of a torus using 5-port routers, but replaces the global channels with low latency transmission lines.

6.2.2 Asynchronous NoCs

There are a number of asynchronous NoC designs available in literature. As is the case with asynchronous design, there are various choices associated with the implementation of these NoC

architectures. Chain [67] makes use of delay insensitive channel protocol, while Gebhardt et. al. [68] and Ghiribaldi et.al. [69] prefer a bundled data approach. The MANGO router architecture provides connection-oriented guaranteed services, as well as connection-less best-effort routing [70]. The QNoC architecture provides multiple service levels and dynamically allocated virtual channels per level [71]. Recently Kasapaki et.al. demonstrated a Time-Division-Multiplexed NoC using asynchronous routers that foregoes the use of global synchronization [72]. Our NoC design is completely asynchronous with self-timed routers designed using relative timing (RT) methodology [22]. All signals follow the bundled data protocol, thereby eliminating the need for synchronizers.

6.3 NoC Architecture and Circuits

6.3.1 Overview

Network simplicity is the key to energy efficient NoC performance. Simple network functions performed by simple circuits result in power savings. Keeping this in mind, we design TECNO to be minimal in structure as well as function. We do not include advanced NoC features such as virtual channels, adaptive routing, quality-of-service guarantees etc. Our network uses basic dimension ordered source routing, single-flit packets, and simple router circuits. Instead of large crossbars that are energy and area inefficient, we use simple 4:1 MUXs for switching. The two most significant bits determine the switch direction and control the Data MUXs. Therefore, no additional address decoding circuitry is required. Bit rotation of the routing bits is the only operation performed between input and output. Each packet works at flit granularity and contains 16 routing bits, and 44 data bits in parallel. Each router has 5 ports, therefore enabling a kn-cube type topology.

Since we use source routing, the number of routing bits depend on the network diameter. For regular topologies like mesh, tori, there are formulae to determine this value. The data width of the link depends on the required throughput, and power and area budgets. The NoC does not cater to any particular transaction layer protocol, such as OCP.

6.3.2 Router

The circuits employed by TECNO are completely asynchronous. While quasi delay insensitive (QDI) asynchronous circuits are more robust to timing variation, we prefer the bundled data (BD) style of design in our designs. BD allows simpler circuits and fewer wires for data transmission which allows far better area and energy efficiency as compared to QDI designs. Another design choice is the use of a 2 phase protocol versus a 4 phase protocol. The throughput across long links is limited by wire delay; therefore 4 phase protocols present twice the link delay as compared to 2 phase protocols and almost half the bandwidth. In contrast, level sensitive 4 phase circuits are smaller, less complex, and more energy efficient as compared to edge triggered 2 phase logic

circuits. To get the best of both worlds, we design our routers to operate using a 4 phase BD protocol, while using a 2 phase BD protocol for communication between routers. Circuits for the translation between the protocols are included as part of the router design.

Our routers consist of two modules – a switch and a merge. They include circuit blocks such as arbiters, 2-to-4 phase converters, and linear controllers. The base modules are adapted from [73]; with the added caveat of using 5-port routers for a torus arrangement. The overall architecture is shown in Figure 6.2.

The switch module picks the output port to which the incoming packet will be routed based on the packet route bits. Figure 6.3 shows the design of the switch module. The 2-to-4 phase converter adapts the link protocol to the router protocol. It generates 4-phase signals that handshake with a BD burst mode asynchronous controller to pipeline the packet from the input to the output latches. A latch is used to pipeline the data. The two most significant bits are used to decode one of four possible output destinations (rr1 – rr4) for the packet. The output request *rr_m* is forwarded to the output destination concurrently with the handshake. Once the output port is chosen, the routing bits are rotated and sent to the merge module.

The merge module arbitrates between the input channels and grants the first incoming request to the output channel. The arbitration circuit consists of a mutual exclusion element, or MUTEX that orders the incoming requests to a shared output channel. The MUTEX drives a multiplexer (MUX) which is a dual circuit of the decoder in the Switch module. Based on the MUTEX output, the MUX selects one of the data streams at its input to store in the output latch. The same MUTEX output generates the lr signal to the 4-to-2 phase converter, which acts the control circuit for the output latch. This converter generates the latch enable signal as well as translates the internal 4-phase protocol to the 2-phase BD protocol used for inter-link communication. The circuit implementation of the merge block is illustrated in Figure 6.4.

All circuits were designed with static low V_t Artisan cell library on IBM10sf process. Custom cells or domino logic were purposefully avoided so as to enable faster CAD flows.

6.3.3 Packet Routing

TECNO uses source routing for its routing mechanism because of its inherent advantages of speed, simplicity, and scalability [23]. Source routing requires all routing decisions to be made a priori at the source. Lookup tables are used to store a set of precomputed routes for each sender-destination pair. A single route is calculated for each combination. The routing bits decide the direction of steering the packet at the switch module; no other computations are performed at the router. At each hop, the two most significant address bits are rotated, and the next two routing

bits appear at the most significant positions. The number of routing bits depends on the maximum hop count or network diameter. Note that every packet requires routing bits, thus contributing a significant overhead for large dimension networks. Also, this means that a network that employs lower hops such as a torus immediately has an advantage over a regular mesh.

The last two bits of the address are an inverse of the final two route bits. The packet knows when it has arrived at its destination if it is directed to go back to the same direction it arrived from. XY routing is used in our network. When multiple possible paths are available, source routing avails the use of a cost function to determine the best possible route for a message. This value can simply be a hop count, or a more complex metric involving wire and router delays. Energy expense can also be a factor in the cost function. For the sake of simplicity, TECNO uses packet hop count as the cost function metric.

6.4 NoC Simulator

Network performance can be estimated using analytical techniques such as queuing theory and probabilistic analysis. However, these methods require making a number of assumptions that affect the accuracy of the results. Also, they are simplistic and do a poor job of modeling the complex aspects of the network. Therefore, simulation is required to understand the network beyond *first-order* models.

6.4.1 Background

Network workload refers to the traffic pattern applied to the network nodes. This includes choosing the source-destination pair and calculating the rate at which packets enter the NoC (injection rate). Workloads can either be *application-driven*, or *synthetic*. Application-driven workloads require a thorough understanding of the system-application interaction beforehand. With fine tuning, this traffic pattern can achieve simulations that closely mirror the real-world behavior of the system. Clearly, this method has limited flexibility. On the other hand, synthetic workloads can be easily modified to suit one's simulation needs. Random, shuffle, tornado, etc. are some examples of traffic patterns that have found use in simulators.

At initialization, a packet entering the network encounters empty buffers and no contention. This is not reflective of actual network behavior and introduces systematic error into the calculations. To avoid this, NoC simulators include a warmup period where the network is allowed to reach a steady state value. This period is decided based on heuristics, and all network measurements up until this point are discarded. This is followed by the sampling period during which various statistics related to the NoC are collected. After the simulations are completed, all packets are flushed out of the NoC in order to remove any bias for future measurements.

The primary metrics that are evaluated during network simulations are throughput, packet and message latencies, and power consumption. Throughput is the rate at which packets are delivered by the network. It is plain that this depends on the rate at which packets are introduced into the network, which is to say, the network workload. Throughput is usually measured as a ratio of either the available bandwidth to the network capacity, or the available bandwidth to the offered bandwidth. Latency measures the delay introduced by the network for a packet to travel from the source to the destination. Apart from the physical delays, this metric indicates the delay introduced by the packets waiting in buffers due to contention. For any on-chip implementations, power is always a prime concern. Simulators calculate the total energy spent in the system during its entire operation. This depends on the switching activity, circuit size, link lengths, etc.

6.4.2 Related Work

There are a number of open source NoC simulators available for use. Noxim [74] uses SystemC to simulate interconnection networks. It provides many simulation parameters including network size, packet size, routing algorithm, traffic pattern, injection rate among others. However, it is limited to a mesh topology. Booksim [26] is a cycle-accurate network simulator. It provides support for various NoC topologies as well as custom options for the router microarchitecture. The entire software is coded in C++; as a result, delay and energy models for the network components lack accuracy. Gebhardt et.al. incorporate synthesized router circuit models in their analysis; however their choices are limited to 3-port T routers and not really suitable for our tori simulations [68]. For our purposes, we only need a limited number of parametric options, while aiming for high accuracy latency and energy calculations. Therefore, our simulator is simply tailored to 5-port routers as described above, and lookup table based source routing. On the other hand, we provide a precise hybrid C++/Verilog operating environment that closely mimics on chip network behavior, the downside being a higher simulation run time. Since TECNO uses unconventional channels for long distance communication, an accurate representation with low level Verilog and SPICE models becomes necessary, while traffic generation, analysis, and data collection are performed more productively with a high level language implementation.

6.4.3 Our Simulator

The simulator used in this work operates at two levels of abstraction. The traffic generation, management, analysis etc. are performed using a high level language (C++). Two synthetic workloads are available in the current version – random and digit permutation. Based on the size of the network, a routing table is generated, and the number of bits required to cover the network is determined. At this point, the packet is created as a collection of routing bits and data bits. As part of

the terminal instrumentation, we implement a source queue in order to provide isolation between the packet generation and network behavior. Timestamps are created to record the time at which packets enter and leave the queue. The traffic manager module houses all the basic functions to interact with the network. It calls the socket function which introduces and removes packets from the network, checks the empty/full status of the source queue, calls the Direct Programming Interface (DPI) function to SystemVerilog, etc. The SystemVerilog module calls the actual grid function in Verilog, which in this case is the device-under-test. The grid comprises of synthesized routers connected in a mesh/torus topology with links that mirror the actual delay/power characteristics of the diffusive wires and transmission lines. The Verilog modules have an independent timekeeping system. The traffic manager translates between the system time used by C++ and the absolute time used by the Verilog. Efficient timekeeping is a key requirement in a NoC simulator since it is the primary recording mechanism for all the important metrics. All routing, switching, flow control mechanisms are hard-coded into the Verilog grid. The high level software plays no part in the actual running of the system. Based on the time of packet shipment/arrival, the traffic analyzer module evaluates packet/message latency and throughput along with hop count and run-time. The *value charge dump* (VCD) file created by the Verilog simulation using ModelSim is parsed to record the number of signal transitions occurring during the sampling phase of the experiment. Separate variables record the transitions on long and short wires. These values are used to calculate the link energy. Network energy data comes from the simulation of the router blocks. These numbers are then collected and analyzed. For the wire delays and energies, simulated values for RC and TL links are used as shown in Tables 6.1 and 6.2.

6.5 Evaluation

We perform two kinds of experiments with the NoCs. In the previous chapter, we observed the difference between using transmission lines and diffusive wires on the performance of a communication channel. Now, we scrutinize the system level implications of choosing one type of link over the other.

6.5.1 Sparse Mesh

Consider a 4×4 sparse mesh network with no local wires and the minimum hop length corresponding to 5mm long link. Such configurations can be used to connect macro blocks at far ends of the chip. We evaluate two cases – each link is an RC wire or they are all transmission lines. Simulations are carried out for three different injection rates.

As seen in Figure 6.5, the latency values display large differences when the links are implemented using diffusive wires and transmission lines. The average latency can drop by up to 59%

between the two implementations. Since all other parameters remain the same, this difference in latencies is a direct consequence of introducing transmission lines. As packets quickly enter and exit the NoC due to the low latencies offered by the TL links, the throughput of the TL enabled NoC design is higher as compared to the regular diffusive wire network. Not only do the transmission lines operate faster, they do so at lower energy cost. This is reflected in Figure 6.6.

6.5.2 Topology Comparison

In the second experiment, we inspect the role of design choices on network behavior. The baseline design is an 8×8 mesh network with each side of length 1mm. The first modification is to connect the extreme nodes to each other, using RC wires of length 7mm. This represents a regular torus. Of course, there are several efforts involved while actually implementing this change, including, but not limited to modifying the routing table. Next, a folded torus is realized by changing the layout such that all wires are of equal length. Finally, our design, TECNO modifies the regular torus design by augmenting it with transmission lines for the long links.

As seen in Figure 6.7, the design choice results in substantial performance variation. Minimum message latency remains more or less the same since every topology has plenty of short paths.However, the maximum message latency metric demonstrates the effect of using a topology with a low network diameter. The mesh performs extremely poorly when compared to the tori networks, at times suffering from a $7\times$ degradation in latency. Similar trend can be observed in the average latency chart as well. Clearly, the right choices for this metric are the folded torus and TECNO that perform similarly across different injection rates. This is because they do not contain any electrically long wires. Both folded torus and TECNO also outperform in the throughput category. The relatively smaller delays introduced by the wiring resources helps the network mostly avoid congestion. The low energy property of transmission line finally allows TECNO to emerge as the clear winner in the energy race as evident from Figure 6.8. Meshes avoid paying the long wire energy penalty, but are still slightly outperformed by TECNO. It must be noted that even though the energy difference does not appear to be significant, it is multiplied by the total number of packets in the NoC. The absolute value of the energy difference in such cases is quite substantial.

6.6 Conclusion

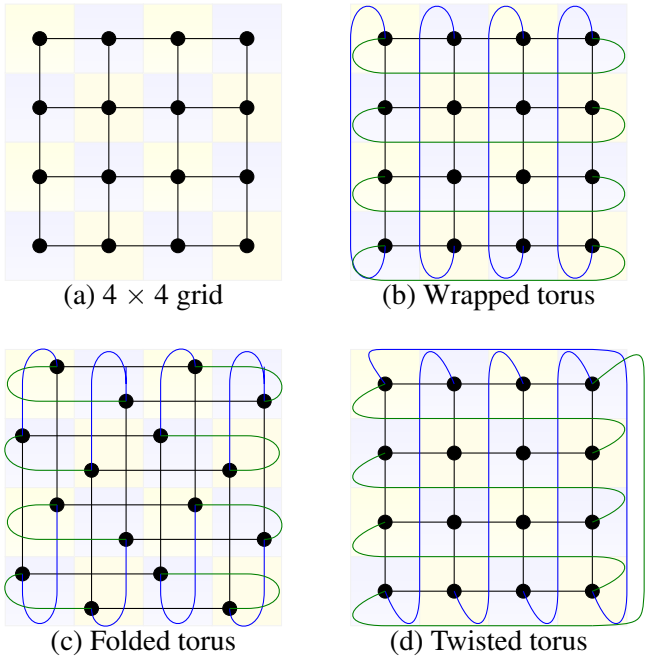
This chapter presents a case study where we apply our high performance, low power techniques to a Network-on-Chip case study in order to see how the system performance is impacted. The results are encouraging and further studies must be performed to validate these results and explore other applications of these methods.

Table 6.1: TL evaluation

Transmission Line (opt)	Latency (ps)	Energy (pJ/bit)	Pitch (μm)	Eye Height (V)	Eye Width (ps)
TL1 (energy)	338.3	0.38	2.8	0.88	339.6
TL2 (latency)	292.9	0.404	7.6	0.85	337.9

Table 6.2: Repeated RC evaluation

Diffusive Wire (metal)	Latency (ps)	Energy (pJ/bit)	Repeater Size (μm)
RC1 (M1)	963.8	1.08	3.6
RC2 (M5)	775.5	0.78	3.6



(a) 4 $\times$ 4 grid (b) Wrapped torus

(c) Folded torus (d) Twisted torus

Figure 6.1: Mesh based routing examples

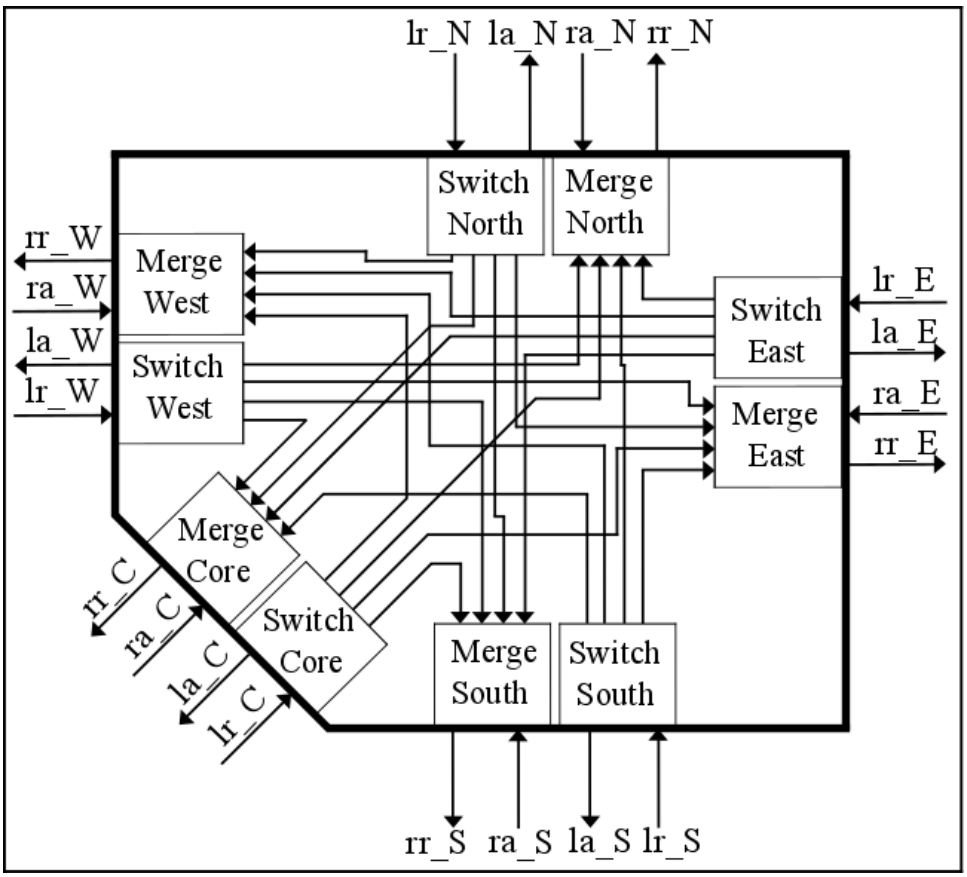


Figure 6.2: Router architecture

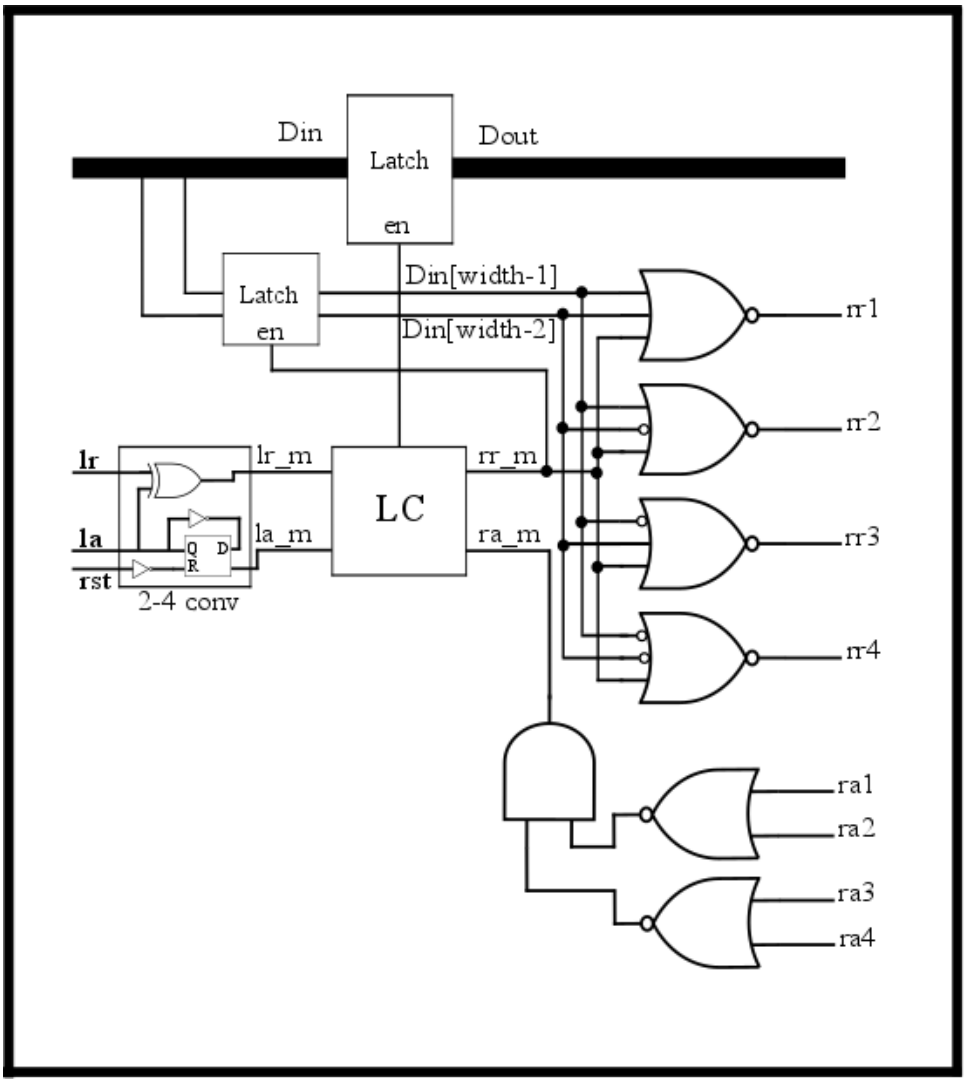


Figure 6.3: Switch circuit

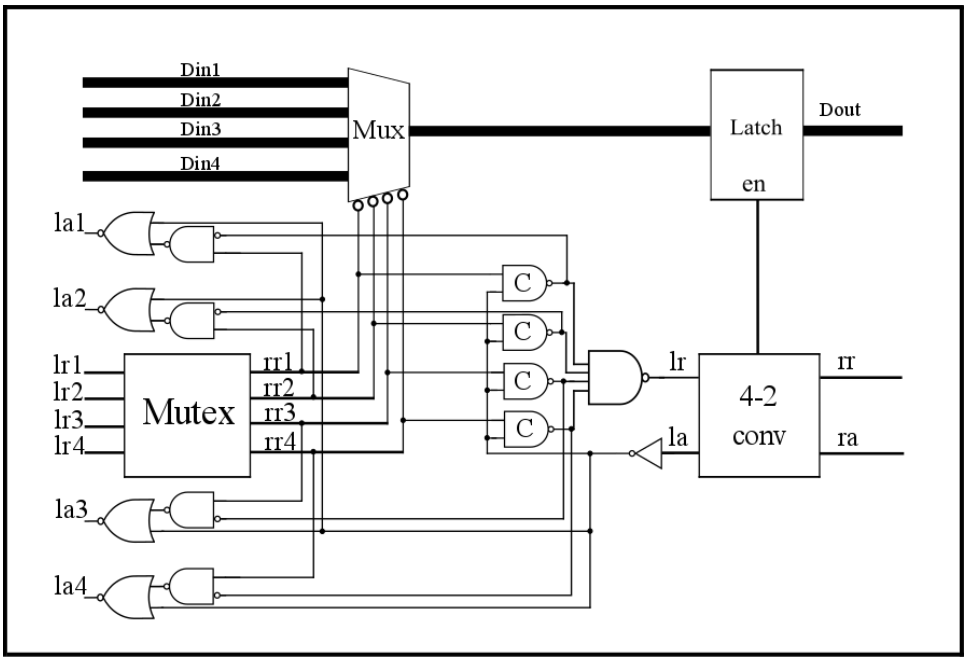


Figure 6.4: Merge circuit

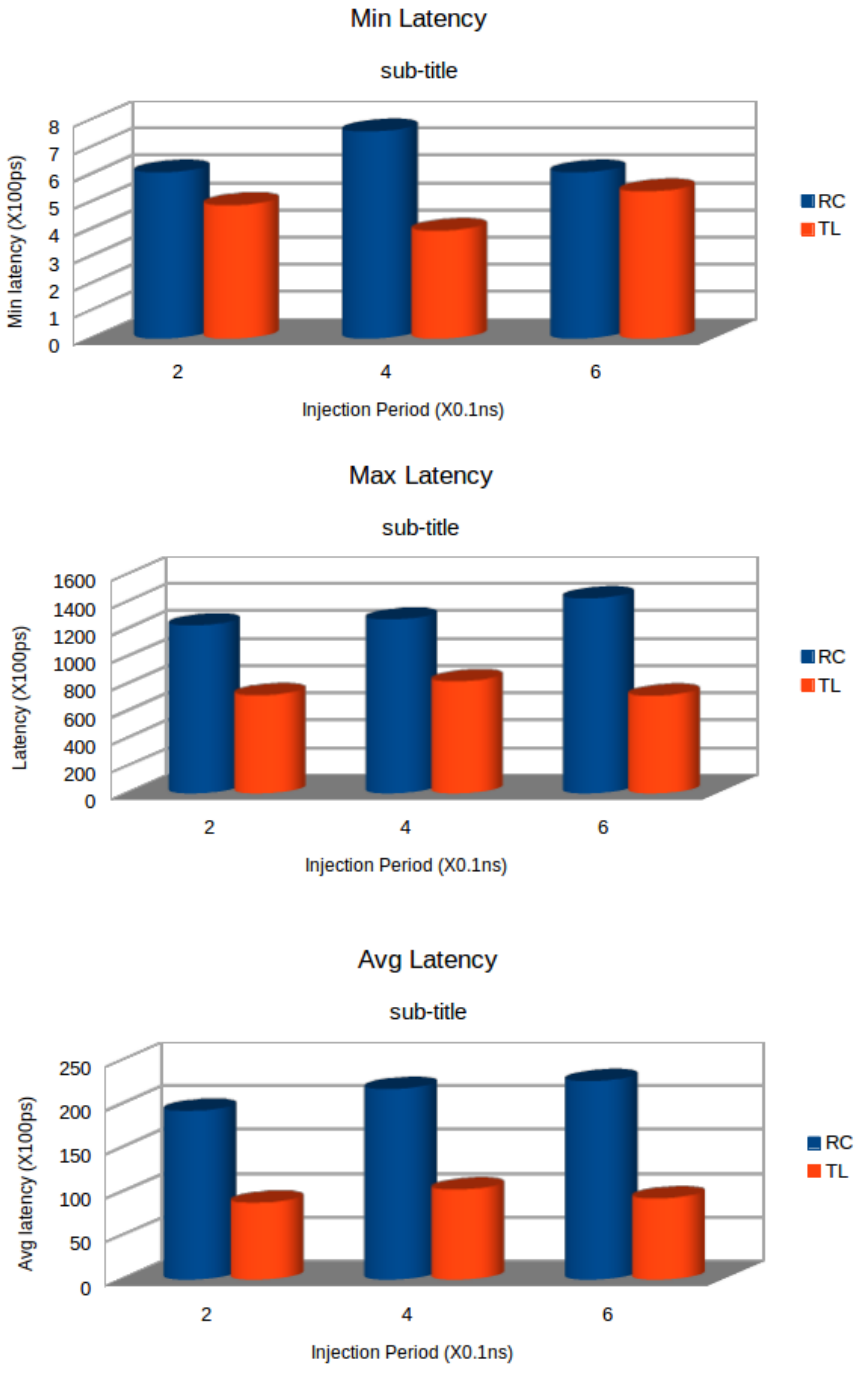


Figure 6.5: Latency of RC and TL mesh networks

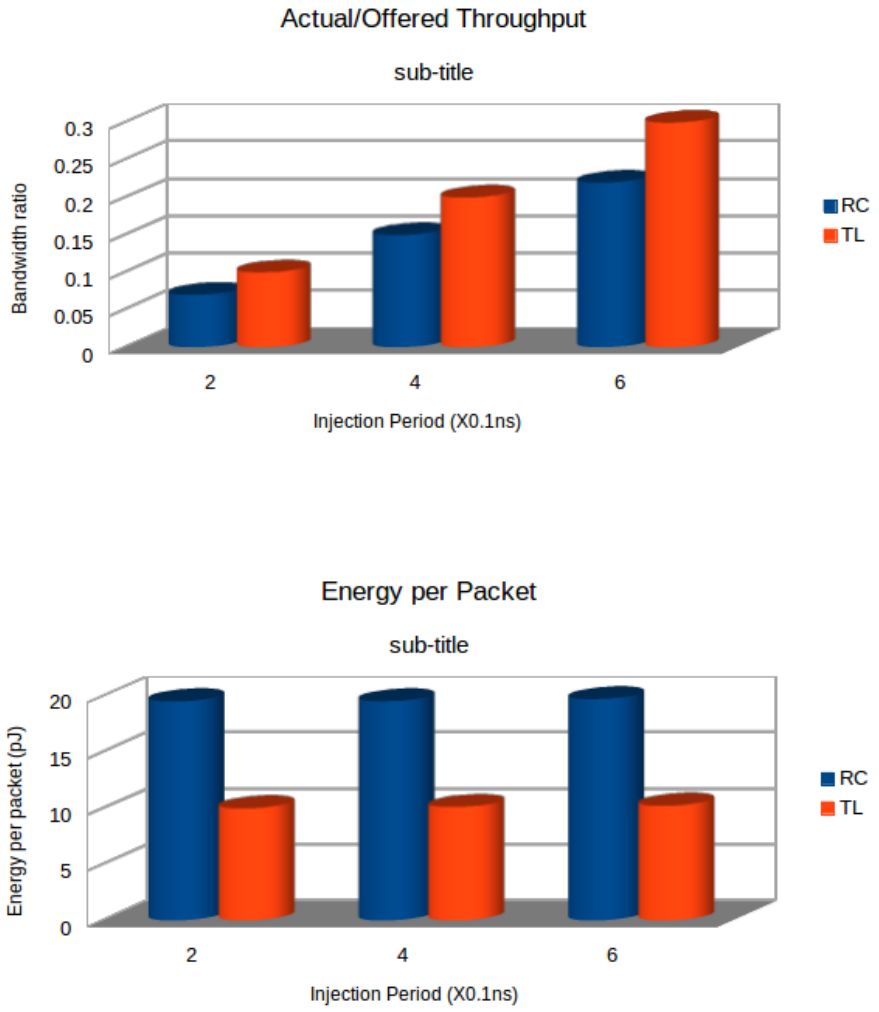


Figure 6.6: Throughput and energy of RC and TL mesh networks

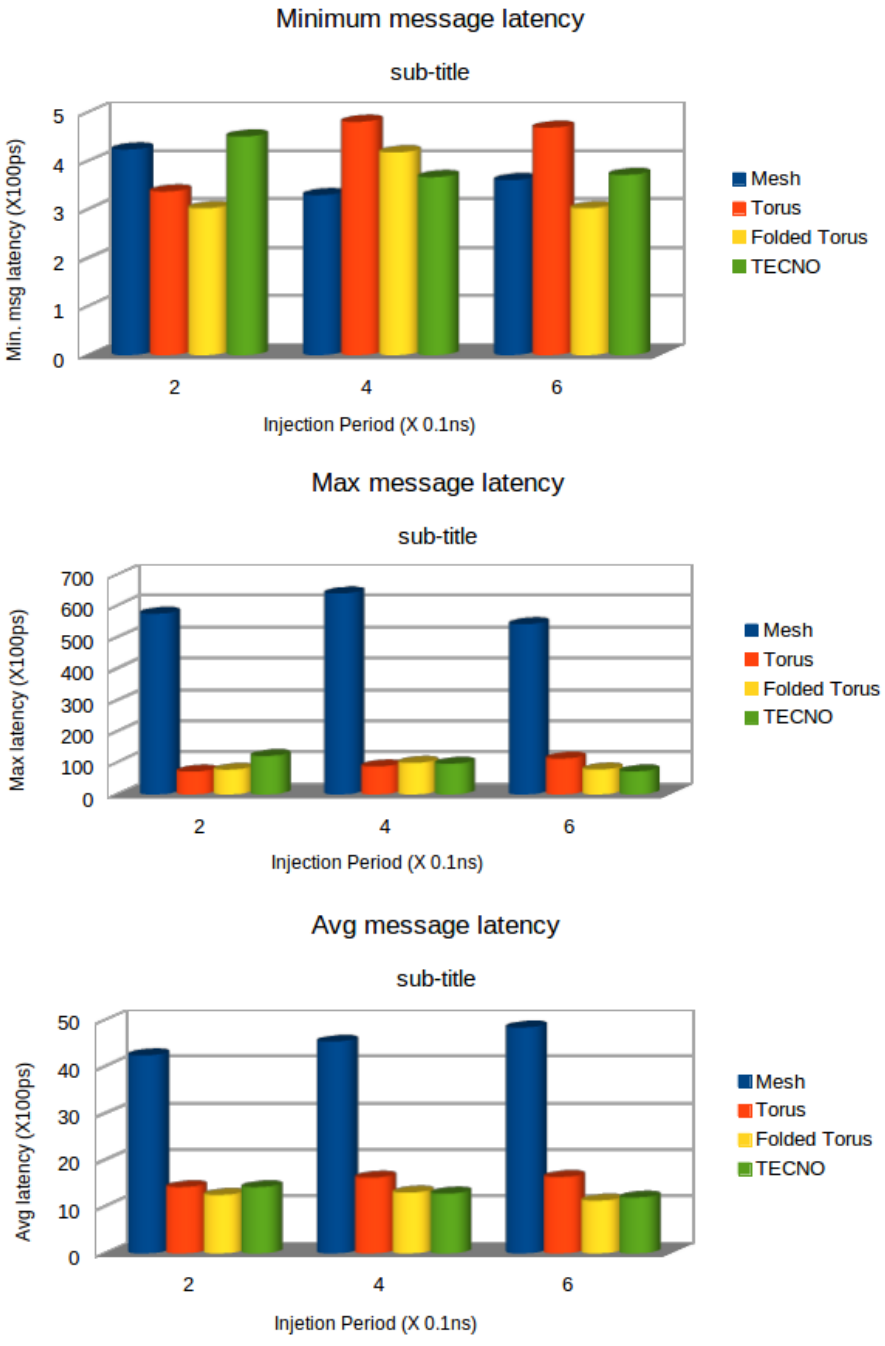


Figure 6.7: Latency of RC and TL mesh networks

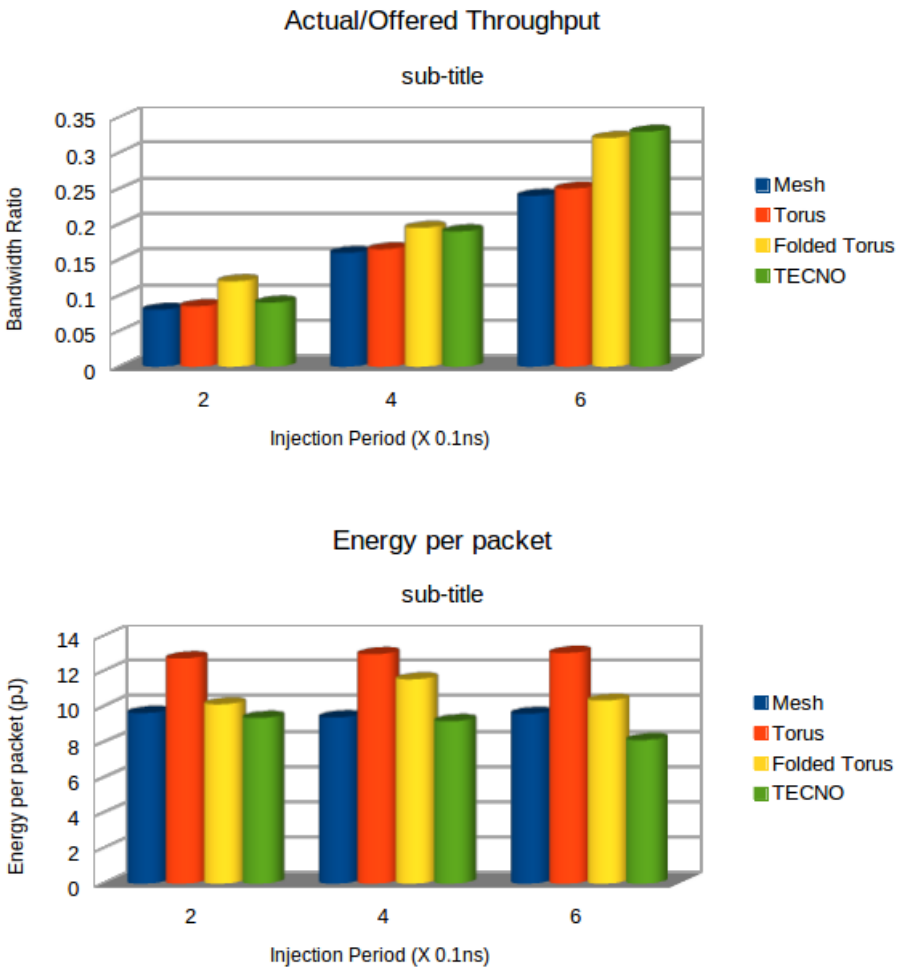


Figure 6.8: Throughput and energy of RC and TL mesh networks

CHAPTER 7

CONCLUSION

This work has explored strategies to improve communication performance for interconnects that carry signals over long distances on a semiconductor chip. There are several challenges to coerce high performance from on-chip wires due to their electrical behavior at nanoscale geometries and high frequencies. Clearly, there needs to be a shift from the traditional approaches into seemingly disruptive ones. Silicon photonics display highly desirable traits, yet the fabrication of such interconnects has not yet reached the point at which the costs justify the gains. Transmission lines are the bridge between current diffusive RC wires and future optical interconnects.

The semiconductor world is already moving from a fully synchronous world to one where different parts of the chip are allowed to run at different rates. We expect this trend to continue as more engineers adopt asynchronous design methods to enable the advantages of multi-frequency operation without the anxiety of employing metastability-prone synchronizers. This work makes an important contribution to the area of asynchronous signaling by facilitating high bandwidth handshake-based communication for long wires.

The paradigm shift from computation-first to communication-first design practices means that network fabrics are becoming more sophisticated. On-chip networks have significantly higher capabilities than their bus/ring counterparts. The adverse effect of complex networks is a higher energy footprint. In this work, we investigate techniques to lower network energy costs while simultaneously improving its performance.

7.1 Inferences

Source Asynchronous Signaling (SAS) protocol enables high throughput communication using bundled-data handshaking. The deleterious effect of wire delay on link performance is negated by this protocol by decoupling the *req* and *ack* signals while still maintaining reliable channel behavior. The elimination of intermediate pipeline stages makes this protocol especially attractive for waveguide-based transport media like transmission lines and optical interconnects. We derive

conditions for correct, free-running behavior and stall avoidance. This technique has the added benefits of low energy operation and reduced link latency.

Using intentional inductance effects in a chip environment requires careful design. Inductance-assisted transmission lines are characterized in this work. High frequency effects on the properties of wire cross-sections are demonstrated. The effect of geometric changes on line behavior are analyzed. Circuits are designed for driving and sensing signals over transmission lines. This work bridges the gap between NoC designer requirements and microwave engineering concepts.

The impact of the choice of communication protocol used in long range data transmission is presented in this work. Transmission lines have different characteristics as compared to diffusive wires. Therefore, the influence of the pipelining method is also markedly different. We evaluate different clocked and unclocked signaling schemes and observe the optimal protocol for each metric.

Finally, we apply our design techniques to implement a network-on-chip and compare it against existing techniques. We observe that our design performs well across the spectrum of environments and produces the best results for a variety of metrics. A NoC simulator that accurately reports RTL level characteristics of the network is also provided.

7.2 Future Work

A lot of the work presented in this thesis is exploratory. There are several exciting avenues to extend this work. Some of them are:

- There is currently little to no EDA CAD tool support for transmission line interconnects. The performance of these interconnects over RC lines serves as a motivating factor to extend CAD tool support for these interconnects. Extending the scope of synthesis, static timing analysis, placement and routing, and other physical design tools to include TL links will help designers adopt this technology faster and more seamlessly. However, substantial research needs to happen in the domain before this is realized.
- There is a lot of published research on high speed circuits. Implementing them for transmission lines will boost their performance. Using equalization techniques will allow higher bandwidth of operation. Using RF design techniques enhances performance at the cost of higher design effort. For special cases, the extra effort may be justified.
- Asynchronous circuit design and synthesis is considered cumbersome and difficult. This is not completely untrue. To make this technique appear more appetizing to engineers, a *push-button* flow needs to materialize to rival the synchronous commercial CAD tool counterparts. Once hazard-free, efficient asynchronous circuits can be built effortlessly, it will unleash a

plethora of applications that adopt the self-timed design style. Consequently, asynchronous NoCs will gain in popularity.

- There is a growing demand for custom NoCs that can be synthesized based on a specified communication pattern. Such synthesis algorithms use concepts from physical design such as simulated annealing and force directed approaches to determine the placement of routers and network interfaces to optimize for a particular metric such as power consumption, average latency, congestion avoidance among others. If transmission lines can be introduced as network links, there is potential for considerable improvement in NoC performance. It will be interesting to investigate the effect of using select low latency, low energy, long lines on the synthesized output.
- The network example used in this work is very simplistic. Additional features such as quality-of-service guarantees, fault tolerance, deadlock avoidance/recovery, adaptive routing, etc. will make for a more compelling design.

REFERENCES

[1] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *High Performance Computing for Computational Science, Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011.

[2] Top500. (2014) List of the top supercomputers. [Online]. Available: http://www.top500.org/lists/2014/11/

[3] A. Chang and W. J. Dally, "Explaining the gap between ASIC and custom power: a custom perspective," in *Proc. IEEE DAC*, Jun. 2005, pp. 281–284.

[4] N. Hemsoth. (2013) No exascale for you! an interview with berkeley lab's horst simon. [Online]. Available: https://www.hpcwire.com/2013/05/15/no_exascale_for_you_an_interview_with_nersc_s_horst_simon/

[5] L. Wilson, "International technology roadmap for semiconductors (itrs)," *Semiconductor Industry Association*, 2013.

[6] A. Naeemi and J. D. Meindl, "Carbon nanotube interconnects," in *Proc. ACM ISPD*, Mar. 2007, pp. 77–84.

[7] C. Xu, H. Li, and K. Banerjee, "Modeling, analysis, and design of graphene nano-ribbon interconnects," *IEEE Trans. Electron Devices*, vol. 56, no. 8, pp. 1567–1578, Aug. 2009.

[8] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3d ics: the pros and cons of going vertical," *IEEE Design Test Computers*, vol. 22, no. 6, pp. 498–510, Nov. 2005.

[9] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless noc as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journ. Emerg. Circuits Systems*, vol. 2, no. 2, pp. 228–239, Jun. 2012.

[10] R. Soref, "The past, present, and future of silicon photonics," *IEEE Journ. Quantum Electronics*, vol. 12, no. 6, pp. 1678–1687, Nov. 2006.

[11] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *Proc. ACM/IEEE ISCA*, vol. 36, no. 3, Jun. 2008, pp. 153–164.

[12] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-tile 1.28tflops network-on-chip in 65nm cmos," in *Proc. IEEE ISSCC*, Feb. 2007, pp. 98–589.

[13] E. Bogatin, *Signal and Power Integrity - Simplified*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2009.

[14] B. M. Beckmann, D. Wood *et al.*, "Tlc: Transmission line caches," in *Proc. ACM/IEEE MICRO*, Dec. 2003, pp. 43–54.

[15] R. Escovar and R. Suaya, "Transmission line design of clock trees," in *Proc ACM/IEEE ICCAD*, Nov. 2002, pp. 334–340.

[16] R. T. Chang, N. Talwalkar, C. P. Yue, and S. S. Wong, "Near speed-of-light signaling over on-chip electrical interconnects," *IEEE Journ. Solid-State Circuits*, vol. 38, no. 5, pp. 834–838, Apr. 2003.

[17] H. Ito, M. Kimura, K. Miyashita, T. Ishii, K. Okada, and K. Masu, "A bidirectional-and multi-drop-transmission-line interconnect for multipoint-to-multipoint on-chip communications," *IEEE Journ. Solid-State Circuits*, vol. 43, no. 4, pp. 1020–1029, Apr. 2008.

[18] H. G. Rhew, M. P. Flynn, and J. Park, "A 22gb/s, 10mm on-chip serial link over lossy transmission line with resistive termination," in *Proc. IEEE ESSCIRC*, Sept. 2012, pp. 233–236.

[19] A. Carpenter, J. Hu, J. Xu, M. Huang, H. Wu, and P. Liu, "Using transmission lines for global on-chip communication," *IEEE Journ. Emerg. Circuits Systems*, vol. 2, no. 2, pp. 183–193, May 2012.

[20] P. Teehan, M. Greenstreet, and G. Lemieux, "A survey and taxonomy of gals design styles," *IEEE Design Test Computers*, vol. 24, no. 5, pp. 418–428, Oct. 2007.

[21] R. Ginosar, "Metastability and synchronizers: A tutorial," *IEEE Design Test Computers*, no. 5, pp. 23–35, Sept. 2011.

[22] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative timing," *IEEE Trans. VLSI Systems*, vol. 1, no. 11, pp. 129–140, Feb. 2003.

[23] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. San Francisco, CA, USA: Elsevier, 2004.

[24] M. Eggenberger and M. Radetzki, "Scalable parallel simulation of networks on chip," in *Proc. ACM/IEEE NOCS*, Apr. 2013, pp. 1–8.

[25] W. Dai and N. E. Jerger, "Accelerating network-on-chip simulation via sampling," in *Proc. IEEE-ISPASS*, Apr. 2014, pp. 135–136.

[26] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE-ISPASS*, Apr. 2013, pp. 86–96.

[27] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[28] K. S. Stevens, "Energy and performance models for clocked and asynchronous communication," in *Proc. IEEE-ASYNC*, May 2003, pp. 56–66.

[29] R. Ho, J. Gainsley, and R. Drost, "Long wires and asynchronous control," in *Proc. IEEE-ASYNC*, May 2004, pp. 240–249.

[30] K. S. Stevens, P. Golani, and P. A. Beerel, "Energy and Performance Models for Synchronous and Asynchronous Communication," *IEEE Trans. VLSI Systems*, vol. 19, no. 3, pp. 369–392, Mar. 2011.

[31] R. E. Nikel, "Low cost 400 MHz source synchronous data links," in *Proc. IEEE EPEPS*, Oct. 1995, pp. 146–148.

[32] A. T. Tran, D. N. Truong, and B. M. Baas, "A reconfigurable source-synchronous on-chip network for gals many-core platforms," *IEEE Trans. CAD*, vol. 29, no. 6, pp. 897–910, Jun. 2010.

[33] T. N. K. Jain, M. Ramakrishna, P. V. Gratz, A. Sprintson, and G. Choi, "Asynchronous bypass channels for multi-synchronous nocs: A router microarchitecture, topology, and routing algorithm," *IEEE Trans. CAD*, vol. 30, no. 11, pp. 1663–1676, Oct. 2011.

[34] A. Mandal, S. P. Khatri, and R. N. Mahapatra, "A fast, source-synchronous ring-based network-on-chip design," in *Proc. IEEE-DATE*, Mar. 2012, pp. 1489–1494.

[35] A. J. Joshi, G. G. Lopez, and J. A. Davis, "Design and optimization of on-chip interconnects using wave-pipelined multiplexed routing," *IEEE Trans. VLSI Systems*, vol. 15, no. 9, pp. 990–1002, Sept. 2007.

[36] P. Teehan, G. G. F. Lemieux, and M. R. Greenstreet, "Estimating reliability and throughput of source-synchronous wave-pipelined interconnect," in *Proc. ACM/IEEE NOCS*, May 2009, pp. 234–243.

[37] R. Dobkin, A. Morgenshtein, A. Kolodny, and R. Ginosar, "Parallel vs. serial on-chip communication," in *Proc. ACM SLIP*, Jun. 2008, pp. 43–50.

[38] D. Gebhardt, J. You, and K. S. Stevens, "Link pipelining strategies for an application-specific asynchronous noc," in *Proc. ACM/IEEE NOCS*, May 2011, pp. 185–192.

[39] L. P. Carloni, A. B. Kang, S. V. Muddu, A. Pinto, K. Samadi, and P. Sharma, "Accurate predictive interconnect modeling for system-level design," *IEEE Trans. VLSI Systems*, vol. 18, no. 4, pp. 679–684, Apr. 2010.

[40] J. You, "Asynchronous Network-on-Chip Design and Evaluation," Ph.D. dissertation, University of Utah, Salt Lake City, UT, USA, Dec. 2011.

[41] J. C. Ebergen, I. E. Sutherland, and R. J. Drost, "Apparatus and method for high-throughput asynchronous communication," US Patent 7 417 993, Aug. 26, 2008.

[42] A. Peeters and K. van Berkel, "Single-rail handshake circuits," in *Proc. IEEE-ASYNC*, May 1995, pp. 53–62.

[43] E. Brunvand, "Low latency self-timed flow through fifos," in *Proc. AR-VLSI*, Mar. 1995, pp. 76–90.

[44] H. Han and K. S. Stevens, "Clocked and asynchronous fifo characterization and comparision," in *Proc. IFIP/IEEE VLSI-SOC*, Oct. 2009, pp. 101–108.

[45] HSPICE. (2001) Extracting transmission line parameters. [Online]. Available: http://www.ece.uci.edu/docs/hspice/hspice_2001_2-128.html

[46] J. A. Davis and J. D. Meindl, "Compact distributed rlc interconnect models-part i: Single line transient, time delay, and overshoot expressions," *IEEE Trans.Electron Devices*, vol. 47, no. 11, pp. 2068–2077, 2000.

[47] S. Palermo. (2015) Transmitter circuits. [Online]. Available: http://www.ece.tamu.edu/~spalermo/ecen689/lecture11_ee689_tx_circuits.pdf

[48] A. J. Martin, M. Nyström, and P. I. Pénzes, "Et2: A metric for time and energy efficiency of computation," in *Power aware computing*. Springer, 2002, pp. 293–315.

[49] I. E. Sutherland, "Micropipelines," *ACM Communications*, vol. 32, no. 6, pp. 720–738, Jun. 1989, turing Award Paper.

[50] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber, "Delay-insensitive, point-to-point interconnect using m-of-n codes," in *Proc. IEEE-ASYNC*, May 2003, pp. 132–140.

[51] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*. Addison Wesley, 1980, ch. 7.

[52] K. S. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. J. Myers, K. Y. Yun, R. Kol, C. Dike, and M. Roncken, "An asynchronous instruction length decoder," *IEEE Journ. Solid State Circuits*, vol. 36, no. 2, pp. 217–228, Feb. 2001.

[53] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, "Exploring sub-20nm finfet design with predictive technology models," in *Proc. ACM/IEEE DAC Design Automation Conference 2012*, Jun. 2012, pp. 283–288.

[54] Y. Cao, T. Sato, M. Orshansky, D. Silvester, and C. Hu, "New paradigm of predictive mosfet and interconnect modeling for early circuit simulation," in *Proc. IEEE CICS*, May 2000, pp. 201–204.

[55] M. Pedram, Q. Wu, and X. Wu, "A new design of double edge triggered flip-flops," in *Proc. ACM/IEEE DAC*, Feb. 1998, pp. 417–421.

[56] A. M. Lines, "Pipelined asynchronous circuits," Master's thesis, California Institute of Technology, Pasadena, CA, 1998.

[57] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. VLSI Systems*, vol. 6, no. 3, pp. 464–474, Sept. 1998.

[58] B. D. Winters and M. R. Greenstreet, "A negative-overhead, self-timed pipeline," in *Proc. IEEE-ASYNC*, Apr. 2002, pp. 37–46.

[59] ——, "Surfing: A robust form of wave pipelining using self-timed circuit techniques," *Elsevier Microprocessors and Microsystems*, vol. 27, no. 9, pp. 409–419, Oct. 2003.

[60] A. Lastovetsky, *Distributed Memory Multiprocessors*. John Wiley and Sons, Inc., 2004, pp. 95–139.

[61] G. Chen, M. Anders, H. Kaul, S. Satpathy, S. Mathew, S. Hsu, A. Agarwal, R. Krishnamurthy, V. De, and S. Borkar, "A 340 mv-to-0.9 v 20.2 tb/s source-synchronous hybrid packet/circuit-switched 16 x 16 network-on-chip in 22 nm tri-gate cmos," *IEEE Journ. Solid-State Circuits*, vol. 50, no. 1, pp. 59–67, Jan. 2015.

[62] B. Daya, C.-H. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. Chandrakasan, and L.-S. Peh, "Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *Proc. ACM/IEEE ISCA*, Jun. 2014, pp. 25–36.

[63] P. Ou, J. Zhang, H. Quan, Y. Li, M. He, Z. Yu, X. Yu, S. Cui, J. Feng, S. Zhu, J. Lin, M. Jing, X. Zeng, and Z. Yu, "A 65nm 39gops/w 24-core processor with 11tb/s/w packet-controlled circuit-switched double-layer network-on-chip and heterogeneous execution array," in *Proc. IEEE-ISSCC*, Feb. 2013, pp. 56–57.

[64] A. Joshi, B. Kim, and V. Stojanovic, "Designing energy-efficient low-diameter on-chip networks with equalized interconnects," in *Proc. IEEE-HOTI*, Aug. 2009, pp. 3–12.

[65] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *Proc. ACM/IEEE ISCA*, vol. 36, no. 3, Jun. 2008, pp. 153–164.

[66] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *Proc. ACM/IEEE ISCA*, vol. 37, no. 3, Jun. 2009, pp. 429–440.

[67] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE MICRO.*, vol. 22, no. 5, pp. 16–23, Sept. 2002.

[68] D. Gebhardt, J. You, and K. Stevens, "Design of an energy-efficient asynchronous noc and its optimization tools for heterogeneous socs," *IEEE Trans. CAD*, vol. 30, no. 9, pp. 1387–1399, Sept. 2011.

[69] A. Ghiribaldi, D. Bertozzi, and S. M. Nowick, "A transition-signaling bundled data noc switch architecture for cost-effective gals multicore systems," in *IEEE-DATE*, Mar. 2013, pp. 332–337.

[70] T. Bjerregaard and J. Sparso, "Implementation of guaranteed services in the mango clockless network-on-chip," *IEE Comp. Digital Tech.*, vol. 153, no. 4, pp. 217–229, Jul. 2006.

[71] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *Proc. IEEE-ASYNC*, Mar. 2005, pp. 44–53.

[72] E. Kasapaki and J. Sparso, "Argo: A time-elastic time-division-multiplexed noc using asynchronous routers," in *Proc. IEEE-ASYNC*, May 2014, pp. 45–52.

[73] D. Gebhardt, J. You, and K. Stevens, "Comparing energy and latency of asynchronous and synchronous nocs for embedded socs," in *Proc. ACM/IEEE NOCS)*, May 2010, pp. 115–122.

[74] F. Fazzino, M. Palesi, and D. Patti. (2008) Noxim: Network-on-chip simulator. [Online]. Available: http://sourceforge.net/projects/noxim