A SIMPLE METHODOLOGY FOR DESIGN TRADEOFF ANALYSIS IN ASYNCHRONOUS CIRCUITS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jonathan Davin Tse

February 2016

A SIMPLE METHODOLOGY FOR DESIGN TRADEOFF ANALYSIS IN ASYNCHRONOUS CIRCUITS

Jonathan Davin Tse, Ph.D.

Cornell University 2016

Engineering design will continue to grow in complexity. The traditional tool of abstraction, while still very functional, has become more nuanced as design considerations have begun to frequently cross levels of abstraction. As practicing engineers, we must develop tools and methodology to combat the increased complexity of the design process. I propose a general methodology leveraging heuristic optimization to automate design space exploration with a focus on exposing the tradeoffs between metrics of interest. This allows designers to make informed decisions when choosing between competing designs as well as build intuition regarding the design and application space they are working in.

While the methodology is useful for all engineering disciplines, I will focus primarily on asynchronous VLSI design in this thesis. To that end, I implemented a research-grade tool called *hopTK* to aid in the analysis of asynchronous circuits. Given a parameterized circuit, hopTK will sample the design space of that circuit to obtain the Pareto front across metrics of interest—typically energy, area, and throughput. We can then use these Pareto fronts to compare designs and build intuition about said designs.

I apply hopTK to two major classes of circuits: communication links and arithmetic logic. I evaluate the major circuit families typical of asynchronous design to find the best-in-class families as well as examining any tradeoffs between circuit families.

BIOGRAPHICAL SKETCH

Jonathan Tse was fortunate enough to attend Franklin W. Olin College of Engineering for his undergraduate education. While there, he focused primarily on robotics and power electronics. In 2008, he somehow graduated with a Bachelor of Science in Electrical and Computer Engineering. While attending Olin College, Jonathan was lucky enough to intern at DEKA Research and Development and assisted in the iterative refinement of what would become the 2009 FIRST Robotics Competition (FRC) control system. This was a nice loop closure, as Jonathan had participated in the FRC throughout high school and credits much of his engineering philosophy and success to his experiences in that program.

In 2008, Jonathan was miraculously accepted to the M.S./Ph.D. program at the Cornell School of Electrical and Computer Engineering, where he switched from robotics to Asynchronous VLSI, a field which differs in physical scale by over six orders of magnitude. Any success Jonathan has had during or after the changeover can be attributed to his personal approach to learning of "learn to learn quickly."

Jonathan believes strongly in the importance of education, not just in the school setting but as a lifelong pursuit of knowledge and understanding. To that end, he intends to make as much of a contribution towards the education of the generations to come in whatever way he can.

For all those on whose shoulders I stand.

ACKNOWLEDGEMENTS

My meager successes, if I may be so bold, would not have been possible without the generosity and patience of others in all things. All failings and shortcomings are my own. I must first and foremost thank my PI, Professor Rajit Manohar, for giving me the opportunity to work with him and essentially paying me to go to school. Managing so many graduate students and a successful career in academia is not an easy job, but Rajit makes it look easy. I must thank my other committee members, Professor David Albonesi and Professor Dawn E. Schrader, as well. Both Dave and Dawn have taken a genuine interest in me and their support and encouragement have been invaluable.

Where would we be without Scott Coldren and Daniel Richter? They have pulled me out of the fire so many times and helped me navigate the labyrinthine bureaucracy that sometimes can crop up in an institution as large as Cornell. Logistics wins wars, and a PhD is no exception. There is a treasure trove of experience and good advice accumulated between the two of them, and they're wonderful people to boot. The Cornell ECE department is truly lucky to have them and their talents. Thank you, Scott and Dan!

I must of course thank my family and friends. Your quiet, unconditional support has meant the world to me over these past years. I only hope that I have not failed you all. I will opt for brevity here, as it would require a thesis-length acknowledgments section otherwise. Let me be clear, the best thing I have taken from my PhD are the personal relationships I have earned and maintained. A few folks deserve special mention, in no particular order: Mom, Dad, my favorite sister, λ, TeamDoge, TeamPig, the Shumby and Casano family, Ginneh, C&E, Annanaut, Brown Bear, KK, TRza, Badger Rasputin, KC2ANT, KC2WAB, KD2BGR, KD2EAT, KD2FOU, WB2EMS.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

OVERVIEW

> The average Ph.D. thesis is nothing but a transference of bones from one graveyard to another.
>
> J. Frank Dobie, *A Texan in England*, 1945

1.1 Why Read This Thesis?

If you're looking for technical innovation, *don't*. This thesis is, as Dobie says, an exercise in grave digging. The PhD process itself is essentially digging yourself into a hole, but I will refrain from belaboring that point. I will now summarize the work of seven years in this short paragraph:

As with every field, there are multiple approaches to solving any particular problem. This thesis espouses the notion that we should frame design decisions using the tool of Pareto-optimal fronts within multiobjective design spaces. In short, I have proposed that we adopt this methodology to provide an answer more satisfactory than "It Depends" to the question "Which solution is better, A or B?" The particular incarnation of this methodology presented here makes use of a multiobjective optimization framework in the context of asynchronous circuits, but the methodology is generalizable across all disciplines. My particular implementation allows the designing engineer to graphically examine the quantified engineering tradeoffs between competing design options and select the best option for their particular usage case.

To my knowledge, this methodology is only hinted at in undergraduate education and only sometimes taught or self-discovered in academia. My implementation

of the methodology itself is pedestrian, if workmanlike—but certainly not a significant technical contribution. However, the shiny bauble that I have shamefully stolen from the grave of corpses, to continue the admittedly generous metaphor for my thesis, is to propose that we, as a community of engineers across disciplines and careers, use this methodology when practicing design and teach this methodology to the next generation of engineers. If, in spite of my warning, my corpus of work continues to interest you, dear reader, then forge ahead. What follows is my meager attempt to describe, develop, and defend the described methodology.

1.2 Introduction

For many years, Moore's Law has dominated hardware design. The transistor count available to designers has doubled every 12 to 18 months, but recent trends have complicated the effective use of these transistors. Furthermore, as we cross the 28 nm node, basic physical principles have begun to limit the growth in transistor count. The degree to which these issues have affected design is still a subject of debate at the time of writing, but one thing remains clear: the level of complexity a designer is faced with has risen over time.

This increase in complexity comes from two major factors. The first is that transistor count generally increases the size and therefore the complexity of a design. The second factor is a direct result of the continual drive to decrease the planar dimensions of transistors and wires, which has a profound negative effect on their electrical characteristics. As an example, we can no longer abstract a long wire as a simple connection on a block diagram, but perhaps we can abstract a long wire with spaced electrical buffering as a connection. This can create additional

levels of design hierarchy as well as introducing more cross-hierarchy considerations for engineers.

In fact, a practicing engineer should be aware of considerations at levels of hierarchy above and below the level they are working at. To reuse the long wire example, the electrical buffering added by the interconnect designer might also be a logical buffer, as in the case with most asynchronous buffers. This additional slack may have profound implications for the microarchitecture, not just in throughput and energy but also correctness by potentially causing deadlock. Of course, with the increase in complexity it is difficult for a single individual to be intimately familiar with all aspects across all levels of hierarchy in a sufficiently large—interesting—design.

As a designer, there are several ways to manage this complexity and also maintain a reasonable understanding of design decisions outside your immediate area of responsibility. The first is to simply throw people and money at the problem and have a large and—hopefully—well-managed design team that stays in regular contact. A second is to forfeit intimate understanding of the lower levels of hierarchy and obtain intellectual property (IP) from a specialized company or group. This is typically done for IP blocks with well-defined interfaces such as SRAMs, SERDES links, off-chip I/O, etc. The third major option is to leverage automation tools to either simplify the design process or aid in building designer intuition.

In short, we have limited time and resources to manage a design task potentially spanning multiple levels of hierarchy and several design teams. Furthermore, there may be conflicting specifications such as a design calling for a functional block with a small die area, an extremely low power envelope, and a high throughput target. This begs the question as to which level of hierarchy a design team should focus

their effort to achieve maximum impact when trying to meet design specifications. I believe that the largest effects are obtained at the microarchitectural level.

Architecturally, the design is more or less fixed. The customer, be it the end user in an industrial setting or the organization funding a grant in the academic setting, will drive the feature list as well as the coarse-grain design specifications. Without renegotiating the desired features, it is up to the design team to meet the requirements. Designer freedom opens up at the microarchitecture level. Here, engineers can organize the design into interconnected functional blocks. While the internals of each functional block can be abstracted away, the basic functionality requirements of each block define the inter-block interfaces as well as a lower-bound on the complexity of the internals of each block. For example, an adder unit must be able to add two numbers together, and we thus know that there will be adder hardware within the block along with connectivity to source the operands and sink the result crossing the block periphery.

Once basic functionality is defined, it is easy to begin the in-depth design process and begin implementing the adder block, for example. However, we might be leaving optimizations on the table at the microarchitectural level. To continue the adder example, if we are certain that we will always be adding or subtracting a 4-bit number from a 16-bit number, we would not implement a adder capable of accepting two 16-bit numbers. Instead, we could implement a 4-bit adder with a counter to represent the most significant twelve bits, which is more efficient. We also have the choice of number representation: twos complement, carry-save, etc. While this choice begins to blur the line between microarchitecture and the implementation details of lower hierarchy levels, we are still well above details such as transistor sizing and placement.

The bottom line is that the design process is quite complex and involves many cycles of iterative refinement and optimization. Design decisions at the microarchitectural level directly affect transistor count, wiring pressure, power consumption, and system performance. Decisions lower in the hierarchy can also affect these metrics, but are not as impactful. Furthermore, many of these lower-level design decisions can be automated, such as transistor sizing. Some decisions, such as choice of circuit family, details of a transistor-level implementation, or data encoding can be troublesome. This is especially true in asynchronous design where tool support is limited and there as yet no standardized design flow.

At the time of writing, asynchronous design has not seen widespread adoption by industry, leading to many competing solutions to the same problem, especially with regards to asynchronous circuit families or design methodologies. While some older solutions have been eclipsed by more modern techniques, it is often unclear as to which proposed design style or circuit family is the best fit for a particular application. Many academic labs will use in-house solutions even though competing solutions may be a better fit for their application, often because they are the most familiar with that particular design methodology or circuit family.

In this thesis, I propose a design process to address the specific problem of choosing an asynchronous circuit family for a given application. If we rephrase design specifications as multi-objective optimization problems, we can leverage a rich body of optimization techniques. For example, rather than attempt to meet a throughput target within an energy envelope, we can ask the question: "In the energy-throughput design space, what are the design points which offer the best energy-throughput tradeoff?" We are not limited to a particular circuit family when asking this question. In fact, we can account for choice of circuit family as

a changeable design parameter when determining the best tradeoff points.

In the process of cobbling together the various parts of this thesis, it became apparent that neither I nor my peers had been officially taught a design process like that I am proposing, at least in our formal schooling. To further investigate this observation, I engaged in a brief, exploratory study of how the design process is learned and practiced in both industry and academia. Interested readers, if there are any, can find more details later in this thesis.

1.3 Organization

Obviously I don't expect anyone to read this thesis from start to finish—or at all. Thus, for your convenience, dear reader, I have closed each chapter with a 3-4 paragraph executive summary. Where appropriate, I also include a criticisms section in each chapter to enumerate as well as address common criticisms of my work. For those of you seeking the most terse version of my thesis, you can find it below in section 1.4.

As for the overall structure of the thesis, it is loosely divided into two major sections. The first covers the methodology I have proposed, along with a description of the hopTK tool that implements the proposed methodology. I also include the results of a short survey, which seeks to understand how practicing engineers in academia and industry do major design tasks. The second section is covers the application of hopTK to asynchronous circuits of two varieties: links and logic. Thank you for reading thus far.

1.4 Contributions

- I propose a methodology using heuristic optimization techniques to obtain a well-populated, evenly distributed Pareto front of points in a multi-objective design space for a design of interest. See chapter 2.
- I motivate why Pareto front analysis is valuable. See section 2.2.
- I report the results of a small survey of practicing engineers in academia and industry. The survey covers their current design practice as well as how they learned the practice.
- I describe *hopTK*, a heuristic optimization toolkit for use with the asynchronous toolflow developed and maintained by the Asynchronous VLSI and Architecture group of the Computer Systems Laboratory at Cornell University.
- I propose an independently developed single-track asynchronous ternary signaling (STATS) protocol for use in thru-silicon via (TSV) applications.
- I apply hopTK to various single-bit and multi-bit on-chip self-timed communication links to determine the best-in-class as well as any tradeoffs that may exist between competing link designs.
- I apply hopTK to various asynchronous logic design styles including QDI, NCL, and Bundled Data to determine the best-in-class as well as any tradeoffs that may exist.

1.5 Criticisms

Your methodology is not novel.

Short answer you are correct. Pareto optimality and heuristic optimization have both been around for longer than my lifetime. Furthermore, the algorithm that I use NSGA-II, which is a heuristic optimization algorithm designed explicitly to generate a well-populated, evenly distributed Pareto front. Even though the methodology does not seem novel, it seems most people do not use it, or something like it. Either this means that people are unaware of the methodology, or something is wrong with the methodology. Novel or not, I do believe the methodology has value and provides realistic, useful results. In other words, the purpose of this thesis is to build an artifact promoting this methodology, regardless of whether or not people actually read it.

So why should I read this at all?

Well, I have already encouraged you not to! More seriously though, it seems that my methodology, novel or not, is not explicitly taught in school nor widely used in industry or academia. Furthermore, in my experience it seems that many people are only willing to try a few design points, if at all. It is also not clear that there is a generally-accepted systematic way of testing different design points to ensure good coverage of the design space. There ostensibly is some worth in reading about a careful application of my methodology in a case study, if for no other reasons than idle curiosity or abject boredom. If you read only this paragraph, the takeaway I have for you is this: When evaluating a design, especially where inter-metric tradeoffs are important, perform a systematic, automated design space exploration to ensure you

have a good understanding of the space to inform your design decisions.

Did you do *anything* innovative?

Well, if you must know, I developed a single-track asynchronous ternary signaling (STATS) link. Aside from building a methodology off of the excellent work of others, STATS is a contribution I can call my own. There are a number of major issues with STATS, which I will detail in section 7.7, but it does work and at least on paper has some interesting benefits.

1.6 Executive Summary

Engineering design will continue to grow in complexity. The traditional tool of abstraction, while still very functional, has become more nuanced as design considerations have begun to frequently cross levels of abstraction. As practicing engineers, we must develop tools and methodology to combat the increased complexity of the design process. I propose a general methodology leveraging heuristic optimization to automate design space exploration with a focus on exposing the tradeoffs between metrics of interest. This allows designers to make informed decisions when choosing between competing designs as well as build intuition regarding the design and application space they are working in.

While the methodology is useful for all engineering disciplines, I will focus primarily on asynchronous VLSI design in this thesis. To that end, I implemented a research-grade tool called *hopTK* to aid in the analysis of asynchronous circuits. Given a parameterized circuit, hopTK will sample the design space of that circuit to obtain the Pareto front across metrics of interest—typically energy, area, and throughput. We can then use these Pareto fronts to compare designs and build

intuition about said designs.

I apply hopTK to two major classes of circuits: communication links and arithmetic logic. I evaluate the major circuit families typical of asynchronous design to find the best-in-class families as well as examining any tradeoffs between circuit families.

The major contribution of this thesis is the argument for and description of the methodology. There are many failings of the methodology, which will be discussed in detail later, but the general idea is sound and hopefully useful to designers.

CHAPTER 2

METHODOLOGY

> If you call failures experiments, you can put them in your resume and claim them as achievements.
>
> Mason Cooley

2.1 Overview

Ostensibly, this chapter describes the main contribution of my thesis—my proposed methodology. In simple terms, my methodology provides a systematic way of evaluating a given design using user-defined metrics. If the design is parameterized, i.e. key features such as transistor sizes are mutable, my methodology will provide a sampling of the design space defined by the metrics provided by the user. What this capability allows is for a user to evaluate a design in a multi-dimensional space either in isolation or in comparison to a competing design.

This is especially powerful in cases where competing designs cover different parts of the design space, e.g. design A offers high performance for a high energy cost and design B offers low performance but correspondingly low energy cost. The tradeoffs between designs A and B may seem trivial, but it is important to note that each design is not represented by a single point in the design space but rather a cloud of points representing different parameter choices. The interesting region is the intersection of the two point clouds. In other words, does the lowest energy design point of A offer more performance than instances of design B that consume the same amount of energy? If the desired operating point of the overall design falls within this performance/energy region, is design A or design B better?

2.2 Pareto Fronts

Comparing two overlapping point clouds is messy. Fortunately, we can reduce each point cloud to a collection of points on the surface of the cloud. In two dimensions this is equivalent to reducing a 2D scatter plot to a curve. As an illustration, we can look at Figure 2.1, which shows a collection of points in an energy-throughput space. Each point represents a configuration of a design evaluated using the common metrics of energy consumption and workload throughput. Ideally we are looking for the highest possible throughput for a given energy expenditure, akin to looking for the best deal when shopping in a store. The Red line in Figure 2.1 shows the *Pareto front* for this point cloud. In other words, the set of points in the Pareto front represent the best energy-throughput tradeoff design points for the evaluated design. For example, if we look at the highest throughput point in the Pareto front, we get 10 throughput for 8 energy—there are no SI units but this is just for illustration. There are very few reasons to implement either of the design points that offer 10 throughput for 9 or 10 energy, especially if your only metrics of interest are energy and throughput.

The algorithm sketch for finding the Pareto front from a given design space point cloud is relatively straightforward. Using our throughput-energy example, we simply sort the points by throughput and examine the highest throughput points. Of those, we choose the point with the *lowest* energy. We then examine the set of points with the next highest throughput and from among them choose the lowest energy points. This algorithm generalizes to cases such as an energy-area space, where we are looking for the smallest area and the lowest energy. It also generalizes to point clouds of more than two dimensions in which case the resulting Pareto front is a surface as opposed to a curve.

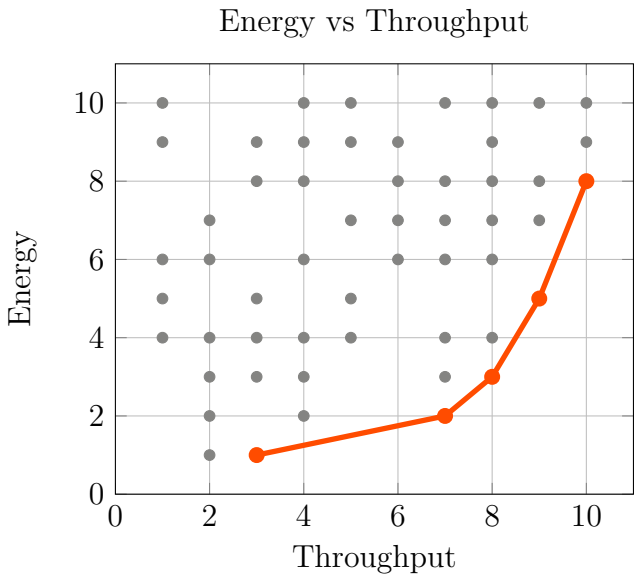


Figure 2.1: Pareto Front Illustration

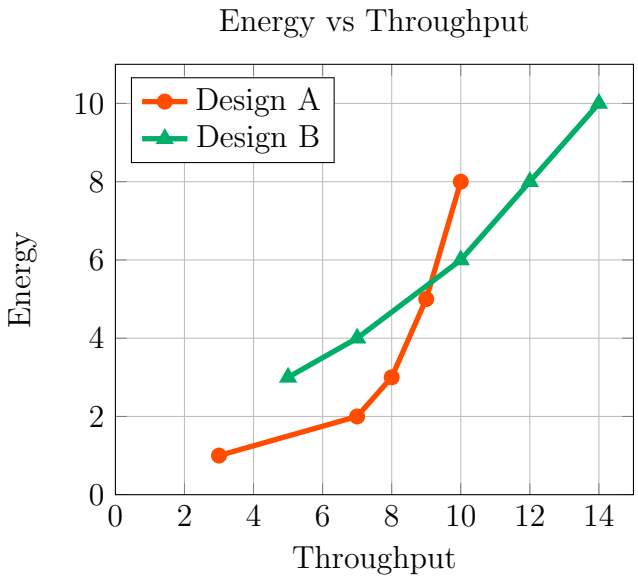


Figure 2.2: Pareto Front Crossing

The power of the Pareto front becomes apparent once you have multiple fronts shown on the same plot, as in Figure 2.2. The Red curve is the same as that from Figure 2.1, but we've added a new Green curve representing the competing design B. This plot allows designers to choose the best design for their given needs. If their system calls for a design with a lower throughput, they should choose design

A, as it offers lower-energy configurations for lower-performance. However, if the system requires throughput above 10, design B is the clear winner as it offers more energy-efficient throughput in that region. In fact, design A does not offer throughput above 10.

One additional benefit of this Pareto front analysis is a great reduction in the number of design points a designer must manually examine. In practice, an automated tool will sample the design space to generate the point cloud, as in Figure 2.1, and present only the resulting Pareto front to the designer. The designer can now examine these points individually to correlate the design parameters to the metrics of interest. For example, we can see the effects of transistor sizing on throughput and energy, for example, by examining only the *best* points.

In summary, Pareto front analysis offers two major advantages. The first is the capability to directly compare two or more competing designs using several metrics across a range of operating points. This was illustrated in Figure 2.2, which allowed us to effectively choose between designs A and B based on our desired throughput target. The second major advantage is that the Pareto front represents the best points, allowing a designer to make the most efficient use of their time. We can examine the effects of tuning particular parameters within a particular design and build intuition not just about our design but about our design space.

2.3 Optimization

Thus far, we have provided the first order definition of a Pareto front, and we have discussed the benefits they provide to designers. This section is primarily concerned with the details of how we can obtain a Pareto front for a given design.

I will intentionally omit the technical implementation details, as they distract from the core idea. I provide a full description of the implementation in chapter 5 for any interested parties.

Let us start with a buzzword laden run-on sentence: My proposed methodology is one which uses heuristic optimization techniques to evaluate a given design in a multi-objective design space with the intent of providing a designer a well-populated, evenly-distributed, multi-dimensional Pareto front of design points for further evaluation. If that made sense, great. If not, I will break it down:

Suppose you are given a design to optimize. What does that mean? Well, first you have to ask what metrics am I optimizing for? Is the design supposed to be fast? Energy-efficient? Small? Once you have determined the metrics you care about—typical ones include energy, area, and throughput—you start calling them *objectives*. For example, it might be the objective of your optimization to minimize energy and area while maximizing throughput. A problem with multiple metrics of interest is typically referred to as a multi-objective optimization problem.

Single-objective problems are relatively rare in practice, but abound in textbooks and problems sets. You might be asked to find the minimum of the function $f(x) = 2x^2 + x - 3$, for example. Readers who still remember basic calculus—I barely do—will quickly determine that the minimum occurs at $x = 0.25$. Real-world problems are never this cut-and-dry. Even if a technical problem is relatively straightforward, you must still account for resource expenditure such as design time, the cost of fabrication, etc.

Furthermore, scoring well on all metrics in a multi-objective problem is often an exercise in mutual exclusion. As a toy example, fast circuits usually have large

transistors to increase current capacity, thus making transistor size proportional to operating speed. Transistor area translates directly to die size, which translates directly to manufacturing cost. By this line of reasoning, it is in a designer's best interest to make transistors just big enough to meet a throughput target and no bigger.

There are many classes of optimization technique, and the field of optimization itself is still an active research area. I will refrain from enumerating and describing all possible sub-fields of optimization as that is a Sisyphean task. Still, to provide context for you, dear reader, we can loosely describe the classes of optimization techniques. We have already encountered one, the *convex* optimization problem, with our $f(x)$ example earlier. Problems of this class assume the multi-objective—multi-dimensional—design space is convex, i.e. there is a global maximum or minimum. We can thus iterate through some algorithm and arrive at the best solution. Other classes make other assumptions, such as all parameters take on integer values, solutions are discrete-valued, inputs can be modeled as random variables, etc. My methodology focuses primarily on heuristic optimization techniques, because they do *not* assume there is a global optimum. In fact, in multi-objective problems we often care about the tradeoff space, i.e. the Pareto front, as there is often no best solution or global optimum. We can make use of the other optimization techniques in the technical implementation, which I will discuss in more detail in chapter 5.

Heuristic optimization techniques generally make little to no assumptions about the optimization problem. Parameters can be arbitrary and we place no constraints on type or number of objectives we optimize over, so long as we can assign a numerical value to each objective of interest. We also If we look at every possible con-

figuration of a design in the multi-objective space, we obtain a multi-dimensional point cloud, where each dimension represents a metric of interest. Even with automation, evaluating every possible point is prohibitively expensive, especially if the number of design parameters is large.

I will leave the complete discussion of heuristic optimization techniques to those more competent than I, but I will try to provide a working intuition. Heuristic optimization effectively samples a design space in a structured way. The key assumption is that points that are located "close" to one another in the design space are often closely related in design parameters. For example, a slightly faster design may have slightly larger transistors. While this is not necessarily always the case, it is a useful working assumption. Thus we can explore the design space quickly through sampling, i.e. "throwing darts." We can throw darts randomly at first. Some of the design points we sample will clearly be inferior to others, so we can discard them. We can then explore near the "darts" that have better scores on the relevant metrics. Even with this pruning mechanic, many heuristic optimization algorithms continue to make random moves on occasion to prevent getting "stuck" in a local optimum.

So far we've discussed heuristic optimization techniques, multi-objective design spaces, and Pareto fronts. What we are missing is a description of what I mean by well-populated and evenly distributed. As we generate the point cloud describing the design space using heuristic optimization, depending on the implementation details we may cluster our sampling in certain parts of the overall point cloud. This is especially true if we aggressively prune suboptimal points that may lead to good tradeoff points that would lie on the Pareto front. Figure 2.3 illustrates the difference between a Pareto front that is both well-populated and evenly distributed

and one that is not.

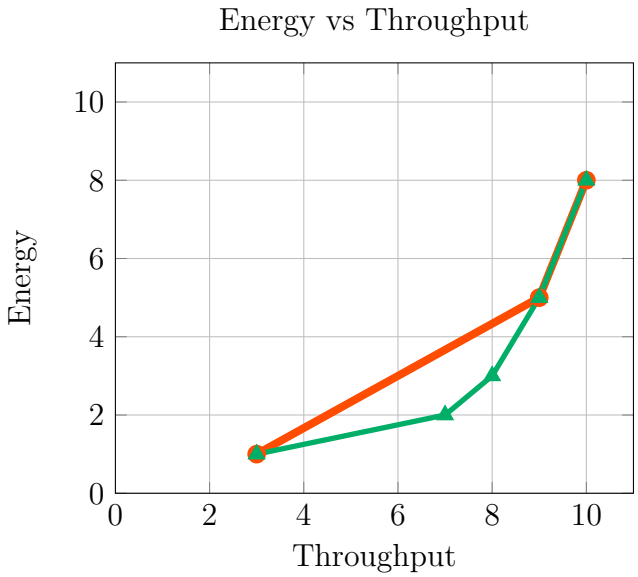


Figure 2.3: Well-Populated, Evenly Distributed Pareto Front

The Green curve represents a well-populated and evenly distributed Pareto front. It has many points along its length and represents a good sampling of the tradeoff space. A designer has a more complete picture of the design space, can choose from a more varied menu of options, and can build a better intuition for the way the design is situated in the multiobjective space. In contrast, the Red is poorly populated and poorly distributed. We do not have a complete picture of the design space, especially at the mid-range throughputs, and are thus limited as to our design choices. Do note that this difference is not due to a difference in the way the Pareto front is obtained but rather is indicating that the point cloud is less populated and full of clusters in the Red case.

2.4 Criticisms

What guarantee do you have of even approaching the global Pareto front?

Heuristic optimization methods do not guarantee this at all. Depending on your specific choice of algorithm and your specific optimization problem, you can place bounds on overall quality of the solution. Usually these bounds are tied to iteration count, i.e. a longer runtime gets you closer to the global Pareto front. If you do not use a structured methodology and use your best design effort, you are also not guaranteed to find a global optimum. If you think you can do better than the methodology, you can always seed the heuristic algorithm with your best effort designer solution.

Why use heuristics at all? Transistor sizing is a convex problem!

This is certainly true. Transistor sizing is a convex problem assuming you set up your assumptions correctly. My answer to this criticism is two-pronged. The first is that transistor sizing is just one of many parameters in a design. Other parameters may have a non-convex relationship with metrics of interest. The second is that transistor sizing, when formulated as a convex problem typically is concerned with minimizing delay. In asynchronous circuits, there is often a timing cycle between adjacent stages and this may not always be the best optimization method. Finally, it is possible to incorporate a convex transistor sizing step after the other non-sizing parameters have been selected by a heuristic.

2.5 Executive Summary

Engineers would like to meaningfully evaluate two or more competing designs using several metrics of interest such as energy, area, and throughput. Any given design can be implemented in different ways by changing various design parameters such as transistor sizing, in the case of circuits. Trying every possible parameter variation is intractable, so we turn to heuristic optimization to sample the design space. Scoring well on all metrics is typically not possible. Different design points often offer tradeoffs between the various metrics of interest.

If we think of the design space as a multidimensional point cloud, each point represents a design instance with unique parameter configuration. The coordinates of each point are its scores on the various designer-defined metrics of interest. We can extract a *Pareto front*, or set of points, from the point cloud which represents the best tradeoff or value of all points. In other words, we choose the fastest design that has energy cost 1, the fastest that has energy cost 2, and so on. Plotting the Pareto fronts of multiple designs on the same axes allows us to directly compare the designs against each other across various regions of operation, which is valuable to a designer.

We are interested in using heuristic optimization techniques that provide a number of options to the designer. For example, we do not want all fast or all slow points, in our Pareto front. We would prefer to have many points distributed along the throughput/speed axis. This is known as having a well-populated, evenly distributed Pareto front.

My proposed methodology is to leverage heuristic optimization techniques to generate point clouds in multi-objective design spaces and extract the Pareto fronts

from each point cloud to present to a designer. This allows the designer to compare and contrast two designs, as well as to build intuition about where each design sits in the overall space. The designer can also examine the design parameters of each point in the Pareto front to understand how varying the parameters affects how each design instance scores on each metric of interest.

CHAPTER 3

RELATED WORK

> Your most unhappy customers are your greatest source of learning.
>
> Bill Gates

3.1 Overview

As I alluded to in chapter 1, my methodology is not new. There is a great deal of research, especially in the Computer Automated Design (CAD) community and the optimization communities, that I have drawn upon. Automated design space exploration using Pareto fronts has been used and proposed before, although perhaps applied directly to the context of asynchronous links and logic.

The application of Pareto fronts to understand the inherent tradeoffs between design points of a particular design is well-understood by many in the CAD and optimization communities, although as we will see in chapter 4 it was not discussed in a survey of practicing engineers with significant frequency.

The remainder of this chapter will be organized into several sections. The first will be an overview of how Pareto fronts are treated in the existing literature. I will also touch on the methods of optimization or Pareto front generation in the existing literature. The last point is a sketch of how transistor sizing is typically handled. VLSI systems are used in a fairly wide array of contexts, so I do not make a distinction based on application space. I address the related work for each case study in the relevant chapters, chapter 6 and chapter 9. Most authors, like myself, emphasize the generality of their approach in an effort to convince their

reader of the worth of their methodology. In practice, I think at least in for the cases of optimization and design space exploration they and perhaps even I are right to make this claim.

3.2 Pareto Fronts

As I discussed in chapter 2, Pareto fronts are a very powerful tool for evaluating the tradeoffs inherent to a design in a multi-objective space. The most frequently discussed space is the energy-delay space, which is often conflated with the energy-performance space. This is a direct result of VLSI designers being concerned with meeting some performance target within a particular power budget. Unfortunately, increased performance, i.e. decreased delay, often comes with increased energy cost. There are a wide array of quite robust single-objective optimization techniques, the simplest being the high-school set the first derivative of the function to zero to find the maxima. Multi-objective optimization, while certainly not new, is not as widespread of a topic as single-objective optimization. As a result, tools to do multi-objective optimization are less well-known.

In some cases, you can refactor a multi-objective optimization problem into a single-factor problem. For a long time, many circuit and architecture researchers made use of the energy-delay product, first proposed by Horowitz [36]. This product is ideal for opposed metrics where you are seeking the minimum for both metrics, e.g. we want things to happen immediately and cost no energy. The minimum of the energy-delay (ED) product is thus the best operating point for the design at hand and, ignoring all other factors, is a good candidate for implementation. The ED metric has been extended to an ED^2 metric, and even an

ED^n metric [60]. While this does lead to a simple, single-number metric and in some cases provides voltage-invariant design valuation [30], a synthetic metric by its nature obfuscates the actual performance numbers of a circuit.

I believe that Pareto fronts are more informative than the single-number synthetic metrics, and literally provide a more complete picture of the design tradeoff to the designer. They also more closely mimic the design process, wherein a designer is tasks to meet a performance target within a power envelope. Rarely is a design so flexible that any performance or energy value is acceptable, so long as the ED^n product is below a certain value.

One of the most oft-discussed themes associated with Pareto fronts in the VLSI literature is the concept of yield, i.e. the percentage of fabricated circuits we can expect to function. In general, yield comes at the cost of increased energy and decreased performance. Pareto fronts an ideal way to capture this tradeoff. For a given design, one can plot multiple fronts, where each front is represents a different yield percentage [74]. Similar to my methodology, this allows direct visual inspection of the design space, but with the additional metric of yield represented as well.

Yield issues are most commonly attributed to Process, Voltage, and Temperature (PVT) variations and are often evaluated using Monte Carlo analysis. In general, the consensus seems to be to use the Pareto front to capture information about design metrics, and to generate a new front for each yield point of interest [38,70,74]. Some treat yield as a design metric, generating a Pareto front where yield is a dimension in the design space [62]. Either approach is valid, and the choice a designer makes will largely be influenced by design context. The general methodology described in the papers cited above has two-steps. The first is

to somehow sample or otherwise explore the design space using non-yield metrics, and the second is to perform some form of Monte Carlo analysis on the design points chosen to determine yield.

The resulting yield points can be either separated into bins, or incorporated into the Pareto front calculation, depending on which sub-methodology of the two described you espouse. My methodology implementation, as described in chapter 5, does not explicitly examine yield in the traditional PVT sense. It does calculate overall failure percentages across all trials, which is informative but not as much as a full Monte Carlo simulation infrastructure would be. This is not to say that the methodology described in chapter 2 does not support Monte Carlo, far from it. It would be a reasonable extension to the existing hopTK tool to include Monte Carlo simulation of each evaluated design, but like most things was dropped due to time constraints. The methodology itself and the evaluation of the links and logic presented later in this thesis are not intended as a yield analysis but rather a demonstration of the main benefits of the methodology as a tool for comparison and understanding of designs.

3.3 Generating Pareto Fronts

While I talk about and espouse heuristic optimization in chapter 2 and use the NSGA-II algorithm [13] in chapter 5, heuristic optimization is not critical to the methodology. The basic requirement of the methodology a technique that in some way samples the design space of interest, providing a collection of configuration points that is well-distributed in the multi-dimensional space of desired design metrics. The simplest algorithm is simply a brute force iterative search through

the design space, trying the entire combinatoric space of every possible design configuration. For simple designs with a relatively small number of design parameters this may be tractable, but this can quickly explode out of the realm of feasibility.

The question then becomes how do we most efficiently explore the design space, and heuristic methods are an excellent answer. In effect, heuristics are a targeted sampling method, which allows us to achieve coverage of the design space without exhaustively trying every configuration. Smedt and Gielen proposed a methodology very similar to and in some ways better than my own more than a decade ago [11]. Their tool, WATSON, is intended to automate design space exploration for RF and other analog circuits. In their paper they offer an argument for why single-objective algorithms are insufficient for design space exploration. I will offer a brief summary of their argument here for your convenience, all credit goes to them.

Single-objective design problems are very rare in practice. Most designs must be evaluated on many different, often contradictory metrics. In order to reduce the multi-objective space to a single-objective space, we employ some sort of aggregate or synthetic metric, e.g. the ED and the ED^n products. As I described earlier, these synthetic metrics obfuscate the actual values of the metrics used to derive them and thus hide critical tradeoff information. Smedt and Gielen suggest that one could weight metrics differently in the product, for example, to obtain a different tradeoff picture, but different portions of the Pareto front would require different weightings. This effectively results in a piecewise combination of cost functions, dramatically reducing the likelihood of convergence, not to mention the additional complexity, runtime, etc.

Smedt and Gielen therefore adopt multi-objective optimization methods. They

also have proposed that we make use of approximated, i.e. curve-fit or multivariate regressed, Pareto fronts, in the event that we want to make mathematical comparisons between two different Pareto fronts. They claim that interpolation is unnecessary as the relationship between design parameters and performance is "relatively flat" [11]. The primary benefit of the approximation is to reduce calls to the cost function, typically some form of circuit simulator, like SPICE [46].

Smedt and Gielen make use of evolutionary algorithms, but not NSGA-II. This is a common theme amongst related works, with other design space exploration efforts using some form of evolutionary algorithm [16,19,62], although some use them in conjunction with other heuristic algorithms such as Simulated Annealing [1] The choice of NSGA-II was a careful one on my part, as it is intended to generate a well-distributed Pareto front close to the global optimum. Others have also made use of NSGA-II, particularly in the context of yield-aware fronts [62,70]. While Smedt and Gielen offer similar benefits using their multivariate regression to approximate the front in regions they do not directly evaluate, NSGA-II might provide more coverage of such regions, although I do not have any quantitative evidence to support that conjecture.

It would be a reasonable next step to combine the results from NSGA-II with the multivariate regression of Smedt and Gielen to further characterize the front. Examination of my particular case studies would suggest that a multivariate regression would "hide" some of the circuit behavior. In particular, I would reference the area/throughput fronts in chapter 6 as prime examples. The stair-step like configurations in the planar link studies clearly indicate the insertion of additional buffers, a feature that the regression would smooth out. I would argue, then, that my methodology as is is more valuable from an educational perspective, as

a designer can see the actual data points and correlate them against the design parameters. While some of these features can be captured by their regression techniques, the key value of the regression lies in the automatic comparison of designs numerically using the resulting curves.

3.4 Transistor Sizing

Transistor sizing is a thesis topic in and of itself, with entire books written on the subject [31]. I will provide a very basic overview here. Transistor sizing of a linear chain of gates is a well-understood topic, but the problem becomes more complicated—NP hard—when branches or fanout are taken into account [54]. There are a number of more formalized mathematical approaches that can help [10,65,69], but heuristics have also seen success [9,26,39,43].

Due to the handshaking inherent to most asynchronous circuits, there can be many electrical loops. Most sizing algorithms can handle electrical loops, typically by breaking them using a heuristic. This is certainly a valid and effective approach, but as I comment in section 6.4, may miss some optimization opportunities arising from loop-related timings. hopTK, the implementation of my methodology, treats sizing using the evolutionary algorithm toolset, which can be quite inefficient computationally, requiring many simulation runs.

I have addressed this inefficiency by quantizing the available sizing choices and collecting gates into sizing groups by gate context, but there is room for improvement here. Future work would definitely address the implementation and inclusion of a dedicated sizing algorithm into hopTK. Non-sizing parameters would still receive the evolutionary algorithm treatment, but sizing would be done for

each of these new population members pre-simulation, cutting down on overall runtime. For the ambitious, this runtime could be spent by implementing Monte Carlo testing to determine yield.

3.5 Criticisms

Someone did this more than a decade ago? Definitely not novel.

Yes, I agree. My work is an amalgam of techniques refined and developed by my betters. There are some minor differentiating points between my methodology and that of Smedt and Gielen, however, and I apply the methodology to a new context.

So this thesis is a classic example of wasted effort?

There are certainly times where I am in agreement with that sentiment. However, based on the results in chapter 4, we have discovered that the work of those individuals a decade ago has not reached the mainstream educational system yet, at least in my small sample size. What this suggests to me, at least, is twofold. First, this type of methodology is not unique to me and thus might have some actual useful bearing on engineering, i.e. other people think it is a good idea. Second, the methodology has not been widely adopted, meaning that a thesis promoting and describing the methodology could be a useful contribution.

3.6 Executive Summary

The application of Pareto fronts to circuit optimization is not a new technique. It has been used in the past, most often to evaluate a single design under different yield criteria. Typically, some form of evolutionary algorithm (EA) is used to generate the Pareto front. In some cases, even a sampling technique like EA is too computationally expensive, so the Pareto front is estimated using multivariate regression on a smaller set of points. This may lose some of the features of the Pareto front, depending on the regression used, but does allow for mathematical description of the front for comparisons, etc.

Other Pareto front and EA-based methodologies and tools exist. They are typically directed towards a specific design application, most often some form of analog circuit. Key differentiating factors when compared against my methodology are this design application focus, an emphasis on yield evaluation, and Pareto front approximation. In contrast, while I target my methodology towards asynchronous circuits, from an educational standpoint I believe it is important to use it or similar methodologies as teaching tools across all engineering disciplines and perhaps beyond. My methodology does not explicitly target the issue of silicon yield, but again, could be extended with effort. I do not estimate Pareto fronts, preferring to retain all features and the well-distributed nature afforded by NSGA-II, but for those interested the approximation is but a multivariate regression away.

Transistor sizing is still a subject of discussion in the research literature. It can be very application-dependent, although a wide variety of general techniques exist. For simple circuits the problem can be considered solved, but for more complicated high-fanout or cyclic configurations we must turn to mathematical models or some form of heuristic to reduce compute time of our sizing algorithms. In the case of

hopTK, I solve this issue by using heuristics and pay the cost of compute time. hopTK could be extended to incorporate a cheaper sizing algorithm to reduce to the overall number of evaluations, which would make a Monte Carlo-based yield estimation feature tractable and appealing.

CHAPTER 4

EDUCATION SURVEY

> We are students of words: we are shut up in schools, and colleges, and recitation-rooms, for ten or fifteen years, and come out at last with a bag of wind, a memory of words, and do not know a thing.
>
> Ralph Waldo Emerson

4.1 Overview

This section aims to make this thesis more than just a bag of wind. As I mentioned in section 1.5, my methodology itself is not particularly novel. As I will discuss in this section, my methodology is not really taught or in widespread use. This of course could be indicative that the methodology is worthless and I missed the memo, so to speak. However, novel or not, I do believe that the methodology is sound and valuable to the engineering design process.

In an effort to draw attention to the fact that my proposed methodology is underrepresented in engineering curricula, I surveyed practicing engineers in both academia and industry. The main questions of the survey cover the design process currently used by the respondents, as well as where they learned that particular design process. My initial hypothesis in building the survey was that most of the respondents would have learned the design process they currently use after completing their undergraduate education. The data shows that the repondents' education did contribute to the design process, but it was definitely supported by project experience and by mentors.

I posit that one of the main reasons for this is that undergraduates need to learn

how to build systems that are *correct* before they can learn to optimize design to meet specifications. So as to not turn this thesis into a treatise on the problems with engineering education, I will make one point and move on. Teaching correct design is a must, but being exposed to optimization is not only good for spiral learning, it also can help build intuition for students. By seeing how designs respond to parameter variation, students can get a feel for the underlying mechanics of the system. They also will have experimental data to help understand the theory, which often is taught with limited context if at all.

The remainder of this chapter is devoted to a detailed description of my survey and how I collected the data. I, of course, will provide the results of the survey and briefly discuss them. Do note that I am not making any claims regarding completeness, bias, etc. This survey is simply meant to provide context for my work and to pique the interest of those more qualified than me to investigate this subject further. For those only interested in the technical details of my methodology, the takeaway of this chapter is that my methodology is not explicitly taught in school and that most practicing engineers do not make regular use of my methodology.

4.2 Survey Details

Engineering in academia and engineering in industry are sufficiently different, at least enough to warrant separate surveys. Both surveys ask very similar questions but in slightly different language. Each survey was preceded by the following header, corrected to reflect the respondents:

> Hello! My name is Jon Tse (jontse.com), and I am asking for your help in providing data regarding the design process in academia/industry

for my PhD thesis. I am trying to evaluate any disconnect between the design process in industry/at the graduate level in academia and how it is taught in courses.

Participation is entirely voluntary and anonymous. Participation does not confer any benefits, compensation, or consideration.

Thank you very much for your time!

I have reproduced both surveys in Figure 4.1 and Figure 4.2. Each question, shown in **bold**, was accompanied by some clarification text when appropriate, which I included below each question. Mandatory questions are marked by an asterisk. All questions were text-response, i.e. there were no multiple choice or yes/no questions.

1. **Affiliation**
   What graduate institution do you currently attend?
2. **How long have you been in academia?**
   Potentially painful question, I know. Sorry!
3. **What are 5 key metrics you can use to evaluate a design?***
   Examples: Energy/Power, Market Demand, Performance, etc.
4. **How do you evaluate these metrics?***
   Examples: Manually try different design points, use an automated tool, etc.
5. **Which level of design abstraction do you typically spend the most time evaluating?***
   Example: Block diagram, code/RTL, physical implementation
6. **When and how were you first introduced to this type of design process?***
   Examples: Class in school, project, by my PI, etc.
7. **How long have you been practicing this design process?***
8. **Why did you move to this design process? Where there any major challenges in switching?***

Figure 4.1: Academic Survey

1. **Affiliation**
   What company or organization do you work for?
2. **How long have you been in industry?**
3. **What are 5 key metrics you can use to evaluate a design?***
   Examples: Energy/Power, Market Demand, Performance, etc.
4. **How do you evaluate these metrics?***
   Examples: Manually try different design points, use an automated tool, etc.
5. **Which level of design abstraction do you typically spend the most time evaluating?***
   Example: Block diagram, code/RTL, physical implementation
6. **When and how were you first introduced to this type of design process?***
   Examples: Class in school, project in industry, etc.
7. **How long have you been practicing this design process?***
8. **Why did you move to this design process? Where there any major challenges in switching?***

Figure 4.2: Industry Survey

The surveys were built and administered using the Google Docs forms feature, which collects results in a spreadsheet. As this was a informal, exploratory survey, I did not make use of a commercial survey service that would have provided a well-distributed sample of respondents. Instead, I sent an email to my personal contacts asking for them to respond and to pass on the survey to as many of their contacts as they could. As the survey does not ask for personal information and is primarily concerned with factual information about a process each respondent has participated in, the survey is not required to undergo Institutional Review Board evaluation.

4.3 Survey Results

I received a total of 37 responses, 15 from engineers in industry and 12 from engineers in academia. The survey ran for about a month: the earliest response was on April 22, 2015 and the latest response was on May 28, 2015. I did not sample the results, i.e. I have included all responses. I have separated the question responses, combining when appropriate, into sections below along with a summary of the overall findings.

4.3.1 Affiliation

Affiliations are more or less straightforward and are reported primarily as demographic data. As I am a Cornell University student, it only makes sense that many of the respondents would be from Cornell, at least in the academic set of respondents. Many of my colleagues and classmates ended up at Carnegie Mellon, and asked their colleagues to respond to my survey. I reached out to a number of contacts at various industry positions, mostly electrical engineering. In some cases, they asked their colleagues to respond as well. While some of the respondents have shared some of my educational trajectory, there are many who I have met in the course of internships or are one degree removed whom I have never met. For those interested, Table 4.1 enumerates all off the affiliations for the respondents. Some of the industry respondents declined to give their affiliation. It should not be a huge surprise that many of the respondents are computer architects, computer engineers, or otherwise associated with some part of the VLSI design process, as that is my field.

Table 4.1: Respondent Affiliations

Industry Affiliation	Count
Advanced Micro Devices	1
Analog Devices	1
Cypress Semiconductor	1
Electric Boat	1
IBM	2
Intel	1
NVIDIA	1
Palantir	1
St. Jude Medical	1
UPMC Enterprises	2
University of Chicago	1
WGT Media	1
Declined to Respond	2
Total	15

Academic Affiliation	Count
Carnegie Mellon University	5
Cornell University	5
Stony Brook University	2
Total	12

4.3.2 Key Metrics

Table 4.2 lists all of the metrics I was able to extract from all the responses. Since I allowed free-form responses and nomenclature differs person by person and field by field, there were no standardized metrics to choose from. In an effort to collect the responses into a usable format, I applied a content analysis approach where I created sets in which to bin responses based on the content of the responses. A more involved study would involve better, more focused questions and a grounded theory approach.

As I am doing this alone, I have provided my codebook of sorts as the Description column in Table 4.2. The count columns provide the cardinality of each set of metrics, e.g. how many respondents chose Area as a key metric of interest.

Table 4.2: Key Metrics

Metric	Academia Count	Industry Count	Description
Application Specific	4	2	A metric very specific to a particular application, such as *fun factor* in interface design.

Area	5	5	Any metric related to physical dimensions, be it planar silicon area or otherwise.
Complexity	2	5	Any measure of engineering feasibility, how many moving parts a design has, or if the respondent actually used some form of the word complex.
Cost	5	6	Cost represents cost to build, cost of parts, cost of maintenance, cost to the customer, etc. Essentially it is any representation of monetary value.
Demand	2	3	Market demand, user interest, etc.
Documentation	0	3	Any metric assessing the documentation portion of the design/maintenance process.
Efficiency	3	3	Refers to efficient use of resources either in the manufacturing process or during operation of a completed and deployed design.
Elegance	0	1	There was only one respondent who used this metric, and they did not elaborate or give additional explanation.
Energy	7	5	Energy expenditure of a design during operation.
Features	0	5	As a metric represents total feature count from a consumer marketing standpoint, or how well a design covers a given application space.
Flexibility	0	2	Any metric dealing with the adaptability of reconfigurability of a design.
Latency	1	0	A measure of elapsed time in a design.
Maintenance	1	4	Any metric evaluating the ease of maintenance or upkeep of a deployed design.
Manufacturability	1	0	Used in isolation, presumably to refer to ease of manufacture.
Performance	6	7	Any metric evaluating frequency, throughput, bandwidth, or some sort of operations per unit time.
Power	3	6	Joules per second. It was explicitly mentioned alongside energy often, so it is its own metric.
Quality	0	2	Any measure of adherence to best practices, use of good materials/techniques, or where the word *quality* was explicitly mentioned.
Robustness	1	1	Like elegance, given as a one-word answer to this question.
Safety	0	2	Presumably a measure of user safety, but not explicitly defined as such.
Specifications	1	4	Any measure of how well a design meets user specifications or commonly used standards.
Sustainability	1	0	Any metric measuring the environmental impact of a design.
Time	4	3	Any mention of time. Typically means design time, but sometimes was used as time spent for maintenance tasks.
Usability	1	3	Any measure of user experience.

All in all, Table 4.2 lists 23 total metrics. A total of 7 were represented only in Industry, and a total of 3 were represented only in Academia. Documentation, Elegance, Features, Flexibility, Quality, and Safety were mentioned only by engineers in industry, and Latency, Manufacturability, and Sustainability were mentioned only by academics. Ostensibly academics should be more free from the constraints of project management and product development, which would explain the absence of metrics like Documentation and Features from the list of academic metrics. In short, in academia your project only has to work a handful of times in a very controlled environment. In industry, unless you're part of a research team, you are developing a product that must meet and maintain some quality standard. For similar reasons, industry products much provide additional features, such as a JTAG interface, a robust and standardized software stack, meet safety and usability standards, etc. Monetary cost is relevant to both academics and industrial engineers for grant and budgetary reasons, no doubt.

Of course, I am *not* advocating for bad project management practices in academia. In fact, I think that the assitance provided by my methodology can potentially provide more time for an academic to do more project management tasks. I will briefly stand on this soapbox in front of a non-existant audience: We should not expect academic projects and products to provide the same feature richess as an industrial design—this is often too much work for little gain. We should, however, expect academics to engage in good project practices, especially documentation. If you are an academic reading this work—sorry, no doubt you have experienced some form of wheel re-invention. That simply is not a good use of time. Do make sure to keep good records of your work, how to use your homegrown tools, and the reasoning behind your choices. If not for others, for yourself. After all, being unable to explain your work renders it completely irrelevant and useless—not that

being able to explain makes it useful.

Moving on, as most of the respondents are VLSI designers and architects or closely associated with that particular field, many metrics are very relevant to circuit designs. The usual culprits include Area, Energy, Performance, and Power. While Energy and Power are related, enough respondents explicitly included both as metrics, so I do as well. For completeness, the application specific metrics are as follows: application interference, correlation, effective impedance, estimated gate count, fun factor, SNR, stability, user adoption, user experience, and user stickiness. While estimated gate count could ostensibly be lumped into area, the respondent also included area in their responses. The gate count likely is in reference to FPGA gate count.

4.3.3 Respondent Methodologies

Table 4.3 lists all of the different methodologies used by respondents. In general, many respondents make use of automated tools, simplified models, or even back of the envelope calculations to evaluate designs based on the metrics previous described. However, only a single respondent described a systematic, automated design space exploration tool. While there is no identifying information attached to the surveys, I am aware of who that respondent is, and of the fact that they have used my tool and methodology before.

Table 4.3: Evaluation Methodologies

Method	Academia Count	Industry Count	Description
Automated Exploration	1	0	Using tools to automatically explore the design space.
Automated Testing	4	0	Using tools to automatically evaluate a design. Does not imply automated space exploration, just the evaluation of a collection of design points.

Back of the Envelope	5	1	Anything described as getting a rough estimate of a value.
Benchmarks	0	2	Using well-defined test benches to evaluate a design on particular metrics.
Cost Analysis	0	2	The process of determining the financial drain or return of a particular product, process, or design.
Feature Comparison	0	2	A side-by-side comparison of the features of two or more competing designs.
Hand Analysis	6	8	By-hand evaluation of a design, usually by way of some sort of analytical mathematical model or similar.
Heuristics	0	1	Using a set of well-defined criteria to filter or otherwise evaluate designs.
Manual Exploration	4	8	Human-driven effort to examine the design space. Typically involves selection of a small number of design points that warrant further investigation.
Market Analysis	0	2	Looking at the market in which the product or design would be situated to compare it to other options from competing organizations or companies. Also, examining an unfamiliar or new market for how best to situate a design.
Lab/Field Testing	0	3	Evaluating a prototype or completed design in the laboratory or in the field. Differs from other techniques in that it requires a mostly-finished or at least working design.
Process Optimization	0	3	Any methodology which introspectively examines the design process, typically looking to increase the efficiency of the design process. Particularly associated with the time cost of the design process.
Safety Testing	0	1	Similar to lab/field testing, but with a focus on safety. Specifically addresses the metrics of safety, usually measured against a defined standard.
Simplified Model	6	1	Using a less-complex model of the design to evaluate particular metrics. Can be used to do hand analysis, manual exploration, or simulation.
Simulation	6	2	Any use of a computer to systematically evaluate a model of a design.
Tool Assistance	3	6	Using CAD tools or otherwise to aid in evaluating a design. Differs from simulation because the tools do no necessarily do simulation. Could be an automated sizing tool for transistors, for example.
Top Down	2	0	Starting from the highest level of abstraction, increasing design resolution at each lower level of abstraction. In other words, the simplest model at the highest level.
User Testing	0	3	Any process that involves users or representatives of users of the design.

The differences between academia and industry are more pronounced here. Of a total of 18 different methodologies, 10 were industry-only: Benchmarks, Cost Analysis, Feature Comparison, Heuristics, Market Analysis, Lab/Field Testing, Process Optimization, Safety Testing, and User Testing. In contrast, only Automated Exploration, Automated Testing, and Top Down were academia-only. This is no doubt a direct result of increased resources, both in money and in labor, but also because industry cares about different metrics. For example, safety as a metric is not of interest to an academic, as their experiments and designs are all evaluated in a controlled, closed environment, so there is no need to perform detailed safety testing.

Again, the main point I would like to make here is that essentially all respondents did not indicate that they use a systematic, automated design space exploration methodology. That said, there are some interesting observations. The first is that Hand Analysis was widely used by both sets of respondents. However, Back of the Envelope techniques were more popular in academia, likely because a more in-depth study is prohibitively expensive in time or money. Estimating a number to within an acceptable error range is likely good enough for non-critical parameters, especially for simulation, which is already a source of imprecision and error. Note that this is not tacit approval to estimate all numbers. Fundamentally, back of the envelope calculations are intended to filter the design space, and are acceptable when used as such.

Other interesting observations include the mention of a Top-Down approach in academia but not industry. This could be the result of the additional manpower in industry, where different people are responsible for different levels of hierarchy, and/or the fact that graduate students often work alone on their thesis project

and must understand it in totality. There also is not a strong focus on simplified models in industry, likely because you must actually build the system in industry as opposed to just simulating it in academia.

4.3.4 Introduction and Practice

To get a better sense of the demographics and background of the respondents, I asked them about their relative experience level in engineering, as well as how they learned the design methodology they currently practice. There are two separate questions regarding their experience level, the first being how long they have been in their current engineering role and the second being how long they have been practicing the methodology they described. The average time spent in their current role in both academia and industry was 9 years, although some respondents have been at their job for several decades. It looks interesting that the average time practicing a particular design methodology is lower in academia, but I think some respondents assumed I was asking how long they have been in school and included their entire academic careers.

Table 4.4 shows the histogram bin counts as well as the average years spent, rounded to the nearest integer. Most of the academics were either just starting or near the end of their graduate school careers. Those in industry were more well-distributed in terms of level of experience.

The far more interesting question was regarding how the respondents first learned the particular methodology they currently practice. To be honest, I had originally hoped that overwhelmingly it would not be from coursework, but Table 4.5 refutes that. As a reminder, there were 15 industrial respondents and 12

Table 4.4: Experience Distribution

	Academia		Industry	
Years	Role	Design	Role	Design
1- 2	2	4	2	1
3- 5	0	1	7	7
6-10	7	7	3	4
>10	3	0	3	3
Average	9	5	8	9

academics, meaning that roughly half of the respondents at least in part learned their methodology from coursework. In academia, the same could be said for learning from mentors, mostly PIs, or projects. However, in industry, projects were overwhelmingly responsible for the switch to the respondents' current methodology.

Table 4.5: Method of Introduction

	Coursework	Mentor	Project
Academia	7	5	6
Industry	7	1	13

I extracted the numbers for Table 4.5 based on the text of the response. If a respondent attributed their learning to courses or some form of structured schooling, I associated that response with Coursework. I applied similar criteria to the Mentor and Project categories. Do note that the categories are not mutually exclusive. The takeaway for this table is that, at least regarding the metrics and methodologies discussed above, the engineering education apparatus has not failed. Either coursework alone is enough preparation, or the coursework has prepared the student to learn on the go in projects or from mentors. This somewhat invalidates my assumption that coursework does not adequately prepare students to do design work, but we still do not have a gauge for the quality of their work. All we can say with reasonable certainty is that around half of respondents felt that coursework was helpful in preparing them for engineering design.

That said, perhaps we can learn something from the *why* of the move to a particular methodology. Table 4.6 lists the various reasons given by respondents for their change to the described methodology. Like the other categorization tables, I extracted the list of reasons from the text of their response and combined similar responses. Some respondents self-discovered their methodology or are constantly adapting to changing project requirements, particularly in user experience design cases. However, in general it seems that most respondents were forced by some external force to adopt their methodology. Either there was a need, because "[t]he full description of [their] systems is complex and often unknown," for example, or the change was dictated, particularly in industry. A typical response in the latter vein was:

> Design process virtually unchanged since decades before I joined company. Virtually impossible to change because both customer and company see doing anything differently as introducing risks to cost, schedule, and quality... Often long-tenured employees, too, prefer keeping processes as-is for job security...

Table 4.6: Reason for Use

Reason	Academia Count	Industry Count	Description
Ambivalent	3	4	The catchall field where the respondent did not give much of a response, where they did not seem to have a strong opinion, or similar.
Constant Adaptation	1	2	The respondent continually changes and adapts their design methodology.
Need	4	2	The word "need" was explicitly mentioned, the structure of the response was "I did/did not have X, so I moved to Y," the response was similar to "Because of XYZ reasons," or similar.
No Choice	1	7	If the respondent externalizes the choice to a superior in their organization, either their manager or their PI.

Pre-Defined	4	4	Any response where the techniques were taught to the respondent with little to no improvement or discovery on the part of the respondent. Also includes responses where the words "no choice" were explicitly used.
Self-Discovered	2	0	The respondent developed their methodology largely on their own.

I also asked respondents if there were any major issues in learning/adopting their design methodology. Most declined to elaborate in detail, so it was difficult to extract meaningful trends from the responses. There were several responses, however, which I reproduce here:

- the questions 6, 7, and 8 seem to assume some sudden transformation in the process, while usually the changes are gradual. The risk of switching to something new is usually high and mature industries try to avoid that. For that reason, it is better to start at start-up companies :-)
- Since I worked on a "new" design paradigm, there is a lack of tools to support manipulation of the design. There is none "advanced" stuff like high-level APIs or object oriented manipulation.
- Major challenges: the tools suck! I spent my whole thesis writing new tools...
- The primary difficulty is in selecting appropriate models to base our design decisions on.
- Challenges include that it's difficult to see how a lot of the higher level tools map to actual implementation. There are many layers of complexity that are necessarily abstracted away, and it is important to see them all, but it takes a lot of effort and time and understanding.

Three out of the above five responses explicitly mention tool-related difficulties, which certainly is not a surprise. With modern silicon designs easily eclipsing a

million transistors, adequate complexity management with tools is very important. This is particularly true in an academic setting, because teams are smaller, often a single individual.

The point about "selecting appropriate models" is a good one, as it indirectly gets at one of the key difficulties in the computer engineering academic research fields. Due to the huge complexity of modern processors, even smaller ones, even cycle-accurate simulators have grown to be relatively heavyweight. It is very difficult to estimate many factors, such as power consumption, at a high level of abstraction. While there have been a number of efforts to adequately model the power consumption of a particular architecture, these models are not always easily transferable to other, even related, architectures. In my experience, a great deal of research is based upon the assumption that these power models or their ilk are transferable, an assumption which may be flawed.

I would also like to address the first response above, the one describing how the change process is gradual. That is certainly true. I betray my own bias here in assuming that the respondents are current graduate students or recent industry hires—with a decade. That particular respondent stated that they had been working in industry for well over a decade, which would explain their response. In the case of a current graduate student or a relatively recent hire, there may have been a significant change in the way they approach engineering design, hence my question.

4.4 Criticisms

This isn't a real study. There's bias and assumptions everywhere!

Well that's certainly true. It's not intended to be a complete study, nor do I make the claim that it is a representative cross-section of practicing engineers. I do think that it gives a reasonable first-order picture of what computer architects and engineers think about the design process, and is at least useful for providing context for my thesis.

Looks like schooling worked. Why do we need to do more?

Well, while it is true that about half of the respondents reported that their formal education either prepared or helped prepare them for the design process they currently practice, that still leaves the other half. Furthermore, formal education was not the only contributor even in the cases were it was helpful. Finally, the design methodology currently taught does not involve automated design space exploration.

You never explicitly asked about design space exploration!

That is also true. I did not explicitly ask to avoid leading the respondent in a particular direction. I was purposely vague and instead asked how they evaluated the metrics they described. Perhaps that is the wrong way to go about asking engineers for information, as—at least in my experience—they are likely to be very literal about their responses. Still, several did talk about the way they compared designs, which is encouraging. Bottom line, were I to spend more resources on answering these questions it would require a rework of the survey and a much larger response pool. I would be remiss if I did not acknowledge the potential for bias, but I believe the conclusions I have drawn are in keeping with my practical experience and that I have been forthcoming

about my assumptions. You are of course welcome, dear reader, to dismiss anything you read in this thesis, as is the norm in academia.

4.5 Executive Summary

I asked 15 engineers from industry and 12 from academia about their design methodology using a form survey. While I collected some demographic data as well, I asked them for 5 key metrics they use to evaluate a design, the methodologies they use to evaluate those metrics, how they learned the methodology they described, and why they use their described methodology. My intent was to be able to compare and contrast responses from industry and academia, with the hypothesis that perhaps formal education did not completely prepare the respondents to practice engineering design.

Most of the respondents were electrical and computer engineers and thus cared about traditional metrics such as energy, power, area, performance, latency, etc. Respondents from industry tended to care about business-related metrics such as market demand, the features of a design, safety, usability, etc. Industry engineers were also more concerned with the lifecycle of the design, in particular how easy the design is to document, how easy it is to maintain, etc, although both industry engineers and academics were concerned with time expenditure.

As for how designs were evaluated, both pools of respondents made use of automated tools as well as manual estimations. More academics talked about simplifying the problem to a manageable level, likely because there are less resources, particularly manpower, in academia. Again, industry engineers were more concerned with the process as well as being more product focused. They discussed

lab testing as well as process optimization, for example, whereas academics did not. The key point I would like to make here is that no one from either pool, save one academic who has used my methodology, made mention of automated design space exploration. They both mentioned automated *evaluation* of a design point, but not of a space. I think that fact at least situates my proposed methodology in what could be a useful context.

Finally, I also examined the manner in which respondents began using this design process. Approximately half of the respondents indicated that formal education, i.e. a class of some sort, either introduced them to the methodology they use or otherwise aided them in preparing them for the eventual introduction. Others were introduced through the do-learn process of actually applying a design methodology in a project, and still others were guided by a mentor of some sort. What this indicates to me is that formal education is at least somewhat effective in preparing students, but there is a need for continued development in the workforce or in graduate school, which should not come as a surprise to anyone. As to why people began using the methodology they described, most cited a need or a dictate from above to change. Industry engineers in particular were forced by tradition or company policy to use a certain methodology.

No doubt you are wondering what to make of this chapter. The takeaway I have for you is as follows: the design process for both academics and industry engineers is similar, with industry engineers placing more of an emphasis on being product and market driven. For most of the respondents, the design methodology they are practicing now was either taught to them or already in place at the institution they joined. Very few respondents talked about systematic, automated design space exploration, leading me to believe that my methodology and tools might be

somewhat useful to the practicing engineer.

CHAPTER 5

HOPTK: HEURISTIC OPTIMIZATION TOOLKIT

I met a traveller from an antique land
Who said: "Two vast and trunkless legs of stone
Stand in the desert. Near them, on the sand,
Half sunk, a shattered visage lies, whose frown,
And wrinkled lip, and sneer of cold command,
Tell that its sculptor well those passions read
Which yet survive, stamped on these lifeless things,
The hand that mocked them and the heart that fed:
And on the pedestal these words appear:
'My name is Ozymandias, king of kings:
Look on my works, ye Mighty, and despair!'
Nothing beside remains. Round the decay
Of that colossal wreck, boundless and bare
The lone and level sands stretch far away."

Percy Bysshe Shelley, 1818

5.1 Overview

From chapter 4, we learned that automated design exploration is not widely adopted, at least amongst the respondents to my survey. To solve this particular problem in the context of asynchronous VLSI design at Cornell University, I have developed a tool to implement design exploration. As described below, my tool can be extended to cover any metric of interest and be incorporated into a wide variety of the methodologies described in chapter 4.

In chapter 2, I covered the basic tenets of my methodology. This chapter concerns the technical details of my implementation of the methodology. As a side effect of being a researcher in asynchronous VLSI, the toolflow I use on a daily basis is bespoke, so to speak. As such, much of what follows will not be directly usable by the casual reader not affiliated with an asynchronous VLSI group with

lineage tracing back to Caltech. That said, the core implementation is portable to any toolflow.

I have termed my toolflow the Heuristic Optimization Toolkit, or hopTK for short. As a plain English summary of its purpose, hopTK automates design space exploration. The input to hopTK is a parameterized transistor-level netlist description of the device under test along with a test harness that will properly exercise the circuit. hopTK will then spawn multiple instances of the device under test, each with different parameter choices, and simulate the circuit using SPICE. The output is a Pareto front of design points situated in the metric space of interest. Effectively, it automates the user manually changing design parameters and then running simulations.

Those readers who have done design parameter sweeps will be quick to point out the following: Depending on the number of parameters and the granularity or step of the sweep, the number of simulations that need to be run quickly explodes into the unreasonable range. In order to combat this issue, hopTK makes use of some eponymous heuristics. Specifically, it uses various classes of Genetic Algorithms (GA), a subclass of evolutionary algorithms, themselves a subclass of heuristic optimization algorithms. As discussed in section 2.3, hopTK attempts to cover the search space without clustering trials of design points, at least initially.

This may all seem relatively mundane, and it is. hopTK is a graduate student answer to the problem of a toolflow that did not have automated, systematic testing. Even so, it does allow a single individual within a relatively short time the power to evaluate competing designs against one another and make an informed decision based on project specifications.

5.2 Base Toolflow

Before I can adequately explain hopTK, I must give at least a—very—brief sketch of the basic toolflow for our particular brand of asynchronous VLSI. Figure 5.1 shows a very basic cartoon of our asynchronous toolflow. The ultimate goal of the toolflow is to turn a high level description of a system into a working chip, or at least a transistor-level simulation of a chip.

The first three major steps in the toolflow are all Hardware Description Languages (HDLs) at various levels of refinement. The first, Communicating Hardware Processes (CHP) [48], is a high-level behavioral description of the system. In general, no explicit notion of bit widths or other implementation details enter at this stage, but experienced designers will often take such considerations into account as they write CHP. In some instances, especially when dealing with circuits with unusual timing considerations, designers will expand CHP into a bit-level representation that outlines the details of communication between processes in greater detail. Since most communication actions are structured as a handshake action between two different hardware processes, this step is often termed the Handshaking Expansion (HSE).

Typically, most designers skip an explicit HSE step and move directly to a Production Rule Set (PRS), which is a transistor-level description of a hardware process. At this point, the system is abstracted as boolean logic expressions—which can be translated to transistors, and do not need to obey good design practice. For example, the circuits designed might not be CMOS implementable or might have too many transistors in series to work reliably. Of course, good design practice should always be followed.

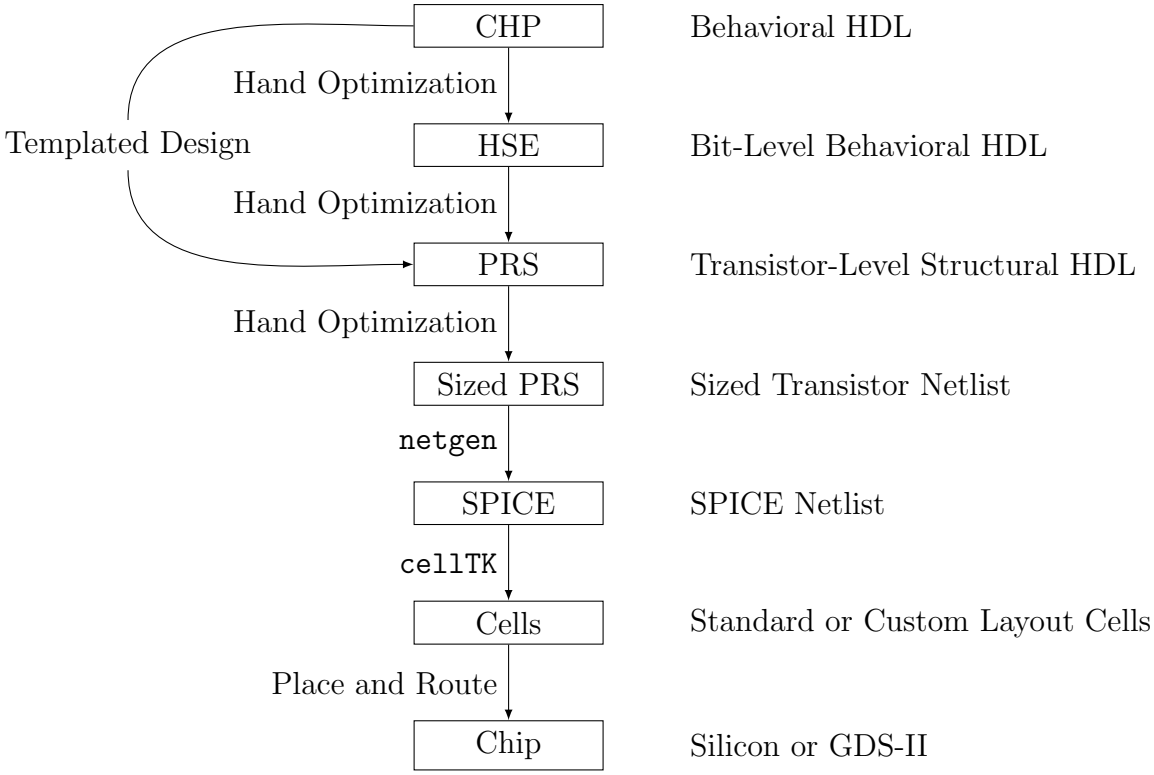


Figure 5.1: Base Asynchronous Toolflow

At its core, chip design can be reduced to data transport and transformation. You are shuttling data from one place to another, performing some sort of computation upon it, and transferring or storing the result. As a result, there are many commonly used structures: buffers and storage elements to hold and transport data, control structures to route data or make decisions based on data, and computation structures performing arithmetic and the like. By maintaining an extensible, parameterized library of commonly used structures, designers can map CHP constructs to PRS *templates*, including computation structures.

While PRS is effectively a transistor-level netlist, basic production rules do not address the analog behavior of actual transistors. Thus, there is a sizing step in which designers specify physical characteristics of transistors as well as other

circuit-implementation details. For example, the ordering of transistors in a stack may be explicitly specified to decrease overall gate latency, reduce charge sharing issues, etc. Once these circuit-level considerations are addressed we can make use of an automated tool, `netgen`, to convert our sized and folded production rules to the industry-standard SPICE circuit specification format.

From SPICE we can make use of a combination of homegrown and industry tools to generate standard cells, i.e. an actual geometric specification for our circuits, and then place and route them to attain a design that can be fabricated. There is a great deal of work that goes into these last two steps, a detailed discussion of which is well beyond the scope of this thesis. I will focus primarily on the toolflow stack at the SPICE level and above, although a concerted effort to integrate hopTK with the standard cell generation provided by `cellTK` is certainly possible.

5.3 Toolflow in Practice

As suggested by Figure 5.1, the toolflow is barely automated, although there are efforts at the time of writing to change that. There have been a number of tools designed to automate various parts of the toolflow, particularly the PRS to Sized PRS step, but in practice we do not make regular use of them. In industry, sizing is generally not a concern as most of the analog-level decisions are abstracted away via blackbox standard cells. Designers have access to a suite of standardized cells in different sizing increments, dramatically decreasing the design effort. Our HDL is also custom and incompatible with HDLs used by other researchers in our field, making sizing tools difficult to repurpose.

To further complicate the issue, at the time of writing, there are no widely used or readily available synthesis flows to convert CHP to PRS. As as result, most of the design effort to date has *not* been automated and involves a significant amount of optimization by hand. As with all types of design, chip design is an iterative cycle. While there are simulators that exist at the CHP and HSE abstraction levels, they are not widely used and my opinion do not encourage the user to follow good design practice.

The first major step in which simulation is widely used is the PRS step. The description the design is close enough to the actual implementation details that low-level bugs are exposed, especially those related to communication timing. While we can estimate metrics such as performance, power, and area at this step, the fidelity of such estimations is poor as some of the analog behavior is not captured. The primary objective of simulation at the PRS level is to evaluate correctness in the face of timing variation. At the SPICE level, however, we can more accurately evaluate the usual engineering metrics.

Evaluating a design at the SPICE level involves several major steps. First we must define and select analog characteristics for transistors such as sizing and folding—transistor ordering. After converting sized PRS to spice using `netgen`, we can then run our simulation and evaluate our design. In most cases, this evaluation loop is where many designers in our research group spend a significant amount of time. One could view hopTK as a tool to automate this loop, not incorrectly, but there are some additional benefits that I hope to expose in this chapter. Hopefully the above toolflow sketch has provided sufficient context for where hopTK sits in the flow.

5.4 hopTK Execution Sketch

hopTK is written in Python and makes use of the DEAP toolkit [27]. In a single sentence, hopseTK automates the setup and execution of SPICE simulations of a sized transistor-level network. hopTK assumes you have a parameterized sized PRS netlist and all supporting simulation files on hand. The design philosophy is file-based, i.e. it assumes design and simulation parameters are stored in files and that the results are written to files. Being intended for use in a UNIX environment, the definition of a file becomes somewhat more flexible but the general statement holds. hopTK, when possible, attempts to encapsulate as much of the simulation as possible into a single simulation directory to reduce any file dependency related problems and to allow for simulation portability and reuse. While the design methodology underlying hopTK is not tied exclusively to circuits, as described in chapter 2, hopTK is designed to work with the toolflow described in section 5.2. As such, it only understands the Asynchronous Circuit Toolkit (ACT) HDL and operates exclusively using the tools mentioned earlier. The general hopTK execution map for a single simulation run is as follows:

1. Parse directory of ACT files to build a database of hardware processes.
2. Extract relevant hardware processes and all dependencies.
3. Copy files to a simulation directory, hard-coding parameter choices in the copied files.
4. Execute the relevant type of simulation.
5. Measure or otherwise extract the relevant metrics for the simulation.

For parameter sweeps or space exploration simulation run sets, steps 1 and 2 are performed once and the results are cached. This of course provides the usual

caching benefits, especially when running several thousand simulations during a space exploration. hopTK supports basic SPICE simulations along with whatever instrumentation the actual SPICE simulator supports. Adding additional basic SPICE simulators is reasonably straightforward, so I will refrain from discussing the vagaries of the simulators I used in this thesis.

hopTK also supports Verilog sources and sinks, which is particularly useful when evaluating computational circuits for correctness. The actual code implementing the compilation of the Verilog harness is not my own, it is the work of Carlos Tadeo Ortega Otero and Filipp Akopyan, two of my graduate school colleagues[1]. They have cobbled together a fragile collection of Perl, Java, and XML to abstract the ugly details of the compilation process from the end user with very consistent success when the right incantations are spoken and Jupiter is in retrograde.

Metric value extraction is accomplished via whatever scripting is necessary. For example, power consumption is calculated via SPICE current measurements, frequency via waveform viewer measurements scripted using Tcl, and area lower-bounded as the sum of the width-length products of all transistors.

The framework itself is pretty bare bones and does not do much hand-holding, if any. For example, the ACT parser is written using `pyparsing` as opposed to a more traditional lexer as implemented by a package like `ply`. If the implementation of ACT as of this writing had an exposed API, I would have used that to save myself from re-inventing the wheel. However, I was interested only in extracting object names and object dependencies from the ACT files, not any fine-grained structures. As a result, the parser is a cobbled-together curly-brace parser with

[1]Entirely not in jest, I definitely owe a lot to these two individuals, not just their implementation of this tool. Thanks to you both!

some Bauer-Naur Form (BNF) expressions for more complicated details.

The end result is a simple but fragile parser that reads ACT files that match a certain stylistic convention. It does not handle all edge cases, and supports a subset of implemented ACT language features—there are of course many unimplemented ACT features that nothing supports at the moment. Parser features and capability grew as necessary, but reached a pretty stable plateau after about a month of continued iteration. As a comparision, `cosim.pl`, the tool written by Carlos and Filipp, has its own Java-based parser that examines only interface specifications for the Verilog harness. Their design philosophy mirrors my own in that respect. I included hooks for the most commonly used SPICE simulators, SPICE output file formats, and measurement tools. As a general rule of thumb, if two or more experimental setups required a particular feature, I rolled it into the base hopTK framework.

5.5 hopTK Details

The basic structure we are evaluating is what I term the Device Under Test (DUT). A DUT is the minimum working example of the design you are interested in evaluating. It should have enough details fleshed out with enough fidelity to give meaningful values when measurements are made. Typically, this means the design has reached the sized PRS stage and is ready for SPICE simulation. hopTK technically supports circuits of arbitrary size, but in practice designs more complex than a hundred gates are not well-suited to detailed SPICE simulation.

Each DUT is represented by a Python class with fields representing metadata or circuit specifications. The DUT class also has hooks into the hopTK parser

to assemble the relevant circuit when it comes time to simulate. Each circuit likely has different needs for its harness, especially with regards to design parameters such as operating voltage. To address this issue, each Python class has an `evaluate()` method that correctly instantiates a circuit, its harness, and measures the metrics of interest. As you are likely already aware, this has the benefit of abstracting the DUT implementation details from the evaluation and design space exploration engine. It also has the benefit of providing the full power of Python as a scripting/glue language to enable the DUT evaluation.

Each DUT can also store the results of its evaluation, thus making each Python class a self contained unit with knowledge of how to set up a simulation, instructions on how to run the simulation, and storage for the results of the simulation. Each DUT also stores the relevant design parameters for the circuit, allowing a designer to immediately correlate design parameters and simulation-obtained metrics.

This makes each DUT a convenient logical container for design instances. We set various design parameters, evaluate the resulting design instance, and receive neatly packaged results. Design space exploration is now a simple matter of instantiating many DUTs with different design parameters, calling the `evaluate()` method, and collecting the results. However, as mentioned earlier in section 2.3, combinatorics makes an exhaustive parameter sweep intractable, so we turn to heuristic optimization methods.

I will leave the theory to the experts who have helpfully written the Distributed Evolutionary Algorithms in Python (DEAP) toolkit [27], but I will endeavor to provide a short description here. Essentially DEAP wraps a genetic algorithm around a collection of DUT instances. The toolkit chooses design parameters

based on its internal heuristics, evaluates each individual DUT, and creates new DUTs with updated design parameters based on the previous evaluations. In this way, DEAP can make targeted evaluations of the design space.

DEAP provides a number of choices of evolutionary algorithms, but we focus primarily on the $(\mu + \lambda)$ Genetic Algorithm (GA) [4] coupled with the NSGA-II selection algorithm [13]. Breaking down that mouthful, we first focus on GAs. Effectively what a GA does is approximate the evolutionary selection process. As described earlier, each DUT has a collection of design parameters that characterize one particular design instance of the circuit of interest. We can consider that collection of design parameters as an ordered list, with each element representing a single parameter. In GA parlance, this list is referred to as a *genome* and each element is referred to as a *gene*. Much like sexual reproduction in the biological sense, we can use the genome of two parents or DUTs and create children based on different parings of their genes. As in the biological sense, we cannot have illegal configurations of duplicate genes, missing genes, etc.

The GA analogy to biology continues with the idea of populations and selection. A group of DUTs is called a population generation. As a concrete example, let's say we have a 16-member population of DUTs. We evaluate all sixteen population members, or *individuals*, using our metrics of interest, which establishes their relative fitness to one another. As a result of their genome—design parameters, some DUTs may have better fitness—higher performance at lower energy cost than others, for example. We then *select* individuals to reproduce and create the next population. The process of reproduction is simply using the genomes of the fittest members of the population to generate new genomes.

For example, we can take the first half of the design parameter list from DUT

A and the second half from DUT B, thus creating DUT C, a child DUT. C is now part of the next generation of DUTs and will eventually reproduce to create the subsequent generation. Of course, the resulting combination of genes that created DUT C might have resulted in a particularly underwhelming child. However, when DUT C is simulated in the next round of evaluations, it is unlikely to get selected to reproduce and pass on parts of its configuration to the subsequent generation.

This population structure is actually quite useful for several reasons, the most practical being the built in parallelization benefits. We can simulate multiple individuals in parallel, stopping to collate and judge the fitness of all the children of that generation before moving onto the next. When most simulations are on the several minute timescale and the typical simulation count hovering around 5000 for the DEAP parameters I have chosen, the parallelization speedup is very necessary.

The other primary benefit of the population organizational model is more subtle. As discussed in chapter 2, we are interested in a well-populated, well-distributed Pareto front across a multi-objective space. In order to achieve that goal, we must cultivate design configurations that are situated at different corners of the design space. Population selection algorithms evaluate a set of individuals, find suitable parents, breed them to generate children, and cull the population. The most simplistic selection algorithms may seek to maximize or minimize a particular metric, randomly choose from the top x% of the population, etc. However, comparatively few selection algorithms seek to keep a set of DUTs that are well-distributed across the design space. NSGA-II, the selection algorithm I referenced earlier, is designed to do just that.

Again, we can examine a concrete example. In the context of the $(\mu + \lambda)$ algorithm I mentioned earlier, we start with a population of size μ. From that

population, we generate λ children, where λ is usually much larger than μ. From the new population of $\mu + \lambda$, NSGA-II will select μ individuals based on their fitness as well as their degree of separation on the current Pareto front to proceed to the next generation. Note that this allows for individuals with good fitness to remain in the breeding pool, so to speak, for many generations. For most of the simulations, μ was 20 and λ was 100.

After some predefined number of populations have been created, judged, and culled, we are left with several thousand points, with some well-distributed along the Pareto front of the design space. Of course, there will be many points with awful fitness values, but some simple scripting will allow us to filter non-Pareto individuals out of the set that undergoes human inspection. While the number of individuals on the Pareto front changes from design to design, five thousand or so evaluations will turn into less than fifty individuals on the final Pareto front. This is a reduction of over a hundredfold and is well within the capability of a single human to analyze in detail. To aid the designer, the genome of each individual on the final Pareto front is readily accessible. As a result, the designer can correlate the genome to the fitness of an individual and thus build intuition for what design parameters are most relevant as well as the general response of the design to parameter variation.

5.6 Criticisms

Why should I use this fragile, highly-specialized, research grade tool?

You probably should *not* use it. It is quite likely that ACT will change at some point, rendering the parser portion of the tool obsolete. Furthermore,

as industry simulation tools evolve, the hooks I have built into the software will not work anymore. However, the tool is proof that a single graduate student of questionable competence can build a framework that implements a new methodology in a short amount of time. The tool works, provides useful results, and has informed design decisions. Graduate school is simply not the place to be producing, much less supporting, robust tools.

How is this not garbage in, garbage out?

Your diction is well-suited to the situation at hand. The tool does implement some error checking based on input/output token stream matching, but correctness is largely enforced by the user. The tool requires relatively intimate knowledge of how it itself works, as well as the simulation environments and support file structure. As a result, there is a fair amount of opportunity to get "under the hood," so to speak, and verify that each part of the tool is functioning as desired. Of course, this means more opportunities for a user to make mistakes. Your mileage may vary, as they say, should you use it at all.

5.7 Executive Summary

hopTK is a python-based framework to aid in the automation of iterative simulation of circuits. It assumes that user is trying to evaluate the design space of a circuit on several predefined metrics of interest. The circuit must be specified in the Asynchronous Circuit Toolkit (ACT) Hardware Description Language (HDL) with enough detail to be able to extract a sized transistor-level netlist from the HDL specification. Paired with the ACT circuit specification is a list of mutable design parameters, which are tunable to elicit different behaviors from the circuit.

hopTK will then parse the ACT specification, generate a harness for simulating the circuit, and iteratively simulate different design configurations of the circuit. This iterative simulation is shaped by the DEAP toolkit and the NSGA-II selection algorithm to extract the best-in-class Pareto front from the set of all simulations. hopTK supports any harness or measurement features supported by Python scripting and has rudimentary multithreading support to decrease overall simulation runtime.

Typical run times for a moderately sized circuit, say a on-chip link or simple arithmetic circuit, hover around 1-3 days on a reasonable compute node at the time of writing. The total number of tried configurations is around five thousand, with the population of the final Pareto front being south of fifty. As compute scales with time, the number of tried configurations could be increased assuming the time expenditure is acceptable.

The hopTK package also includes filtering capabilities for analysis. The most useful of which is a dashboard display of the final Pareto front members and the design parameters associated with each member. This way, a designer has direct access to the correlation between design parameters and the resulting fitness, evaluated of that design instance evaluated on their metrics of interest.

CHAPTER 6

SINGLE BIT LINKS

> In any dispute the intensity of feeling is inversely proportional to the value of the stakes at issue.
>
> Sayre's Law, Wallace S. Sayre
> Charles Issawi, *Issawi's Laws of Social Motion*, 1973

6.1 Overview

As technologies scale and power envelopes tighten, it is time to revisit the design of the humble single-bit on-chip link. Of course, designers will still use the highest bandwidth links within their energy and area budgets, but other considerations have grown in importance. Synchronous designers have long felt the additional constraints associated with clock distribution [57], and asynchronous designers have seen a host of other issues arise as well, such as the need to pipeline long planar links [64] and variability-related problems [50]. Even the link wires themselves present design challenges. Increasing system complexity has begun to put serious pressure on planar wiring resources [34]. At first glance, new process nodes and better back-end-of-line (BEOL) manufacturing have kept the problem mostly at bay. Unfortunately, while designers might have enough wires to meet connectivity requirements in all but the most wire-starved designs, the RC characteristics of the wires have not scaled with transistors. In order to keep shrinking BEOL features without dramatically increasing wire resistance, chip foundries have increased the cross-sectional height of wires. The resistance of long wires can no longer be ignored—the lumped capacitor model is no longer valid in deep-submicron technologies [28]. Furthermore, taller, more closely spaced wires have

resulted in large coupling capacitance values, increased crosstalk, and decreased performance. Some designers of high-frequency systems have resorted to increasing planar wire spacing to decrease wiring capacitance and crosstalk, thereby preserving performance. Over-reliance on this technique can artificially increase pressure on wiring resources, especially for wide buses.

Inductance, especially for very long wires, is also a factor to consider [37]. One design parameter typically not discussed is the *width* of each planar wire. Width and resistance are inversely proportional, but width and capacitance scale together. For wires where a RC-only model is appropriate, on the surface it is unappealing to alter wire width, especially since that increases wiring pressure. This is especially true for protocols with a high wiring pressure, such as some of those of the asynchronous variety. For such a situation, buffer insertion can be more effective [3]. In the case where the wires in question are long enough for inductance to be a significant factor, some designers argue that increasing the wire width may offer attractive energy and delay benefits [18].

Regardless of bus width, wire spacing, or signaling protocol, the energy of intra-chip communication represents a non-trivial portion of total chip energy consumption [45]. Some projections show wide, cross-chip links consuming a hundred-fold more energy in wire transitions alone than in computation [41]. One way to alleviate this problem is to move to 3D integration, for both energy [7] and performance [6], as transmitting data inter-die through a through-silicon-via (TSV) is lower in energy and delay than transmitting data through planar wires across a die. 3D integration has its own problems, such as variability [2] and thermal management [32]. However, for the purposes of this study we focus on the fact that TSV resources are quite limited in comparison with planar wire resources.

TSV pitch is at least 1 µm and is often much larger, on the order of 25 µm [73], well over tenfold the pitch of modern planar wiring.

Self-timed single-bit links are uniquely situated in this complex design space. While they are robust to delay variations, the encodings used incur additional overheads in transition counts and wiring resources—especially important in the TSV case. In comparison, synchronous links make efficient use of wiring resources but suffer from clock distribution and recovery problems. As such, the benefits they provide in comparison with self-timed links are largely dependent on usage case [63].

In light of the pressures on planar and TSV wiring resources by today's asynchronous designers, we present an analysis of self-timed single-bit links. We evaluate representatives from the various classes of self-timed links on the metrics of throughput, energy per bit (token) transmitted, and circuit area. We also present our Single-Track Asynchronous Ternary Signaling (STATS) single-bit link design, which is a single-wire link intended for use in wire/TSV limited environments. However, evaluating each link type at a single point in the throughput/energy/area space is unfair, as factors such as transistor sizing, V_{DD}, and circuit topology can easily change that point. As part of this work we present an optimization framework to obtain throughput/energy/area Pareto efficiency fronts. While this work focuses solely on self-timed single-bit links, multi-bit or even synchronous protocols are on our future-work road map.

6.2 Single-Bit Signaling Protocols

Table 6.1 shows the self-timed single-bit signaling protocols we chose to study, representative of the various classes of competing schemes. Figure 6.1 shows the wire transitions required for each to send the same token pattern. Transitions are aligned in time for readability; in general the different buffer types will *not* have the same latencies. We provide a brief description of each protocol and justification for our choices below.

Other self-timed techniques, such as bundled data [66] and GaSP [67], leverage traditional clock-based datapath elements like flip-flops and latches for pipelining. They generate "clock signals" for each pipeline stage locally, and amortize the cost of this control circuitry over the many bits of a wide datapath. We plan to revisit these techniques in a multi-bit study, which we believe is a more appropriate comparison space. We also omit link protocols which do not include any handshaking flow control, such as [68].

Table 6.1: Self-Timed Single-Bit Signaling Protocols

Name	Handshake	Timing	Voltage	Wires
ATLS	4-Phase	QDI	Ternary	2
RQDI	2-Phase NRTN	RQDI	Full-Swing	3
STATS	2-Phase RTN	Single-Track	Ternary	1
STFB	2-Phase RTN	Single-Track	Full-Swing	2
WCHB	4-Phase	QDI	Full-Swing	3

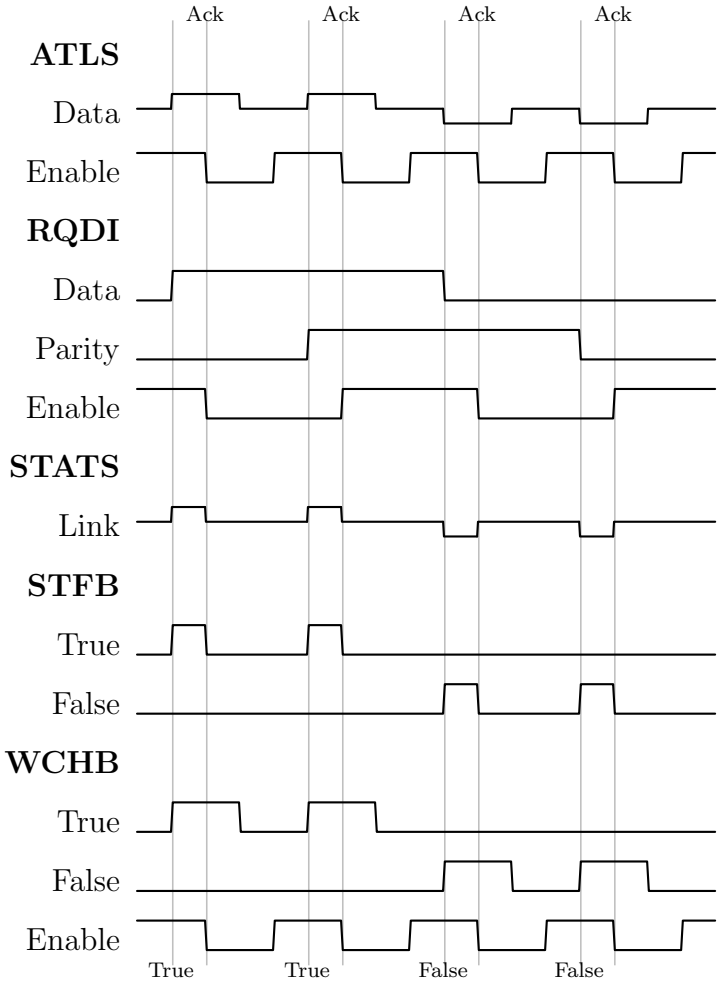


Figure 6.1: Signaling Protocols

6.2.1 WCHB

The Weak-Conditioned Half Buffer (WCHB) [44] is a handshake reshuffling of the 4-phase dual-rail Quasi Delay-Insensitive [49] protocol, which we refer to as *e1of2* (*e* for "*enable*", an inverted-sense *acknowledge* signal). While there are other possible reshufflings such as the PCHB and PCFB [44] used for logic, we chose the WCHB variant because the buffer implementation is small, simple, and fast. Of all the schemes we study in this paper, the e1of2 protocol is the most conservative.

The other link types relax timing assumptions or use more aggressive signaling techniques (e.g. low swing, single track). Evaluating the WCHB allows us to compare the effects of those decisions on throughput, energy, or area.

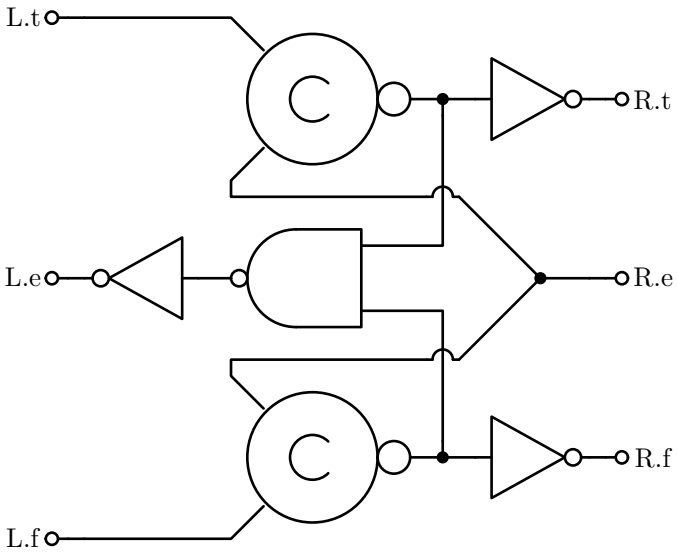


Figure 6.2: WCHB Buffer

6.2.2 RQDI

The Relaxed Quasi Delay-Insensitive (RQDI) buffer design [42] implements a 2-phase, non-return-to-null (NRTN) protocol. It leverages a timing assumption already present in QDI circuits to reduce circuit complexity. We have implemented the LEDR [12] 2-phase encoding, although RQDI supports other 2-phase encodings. Our future multi-bit work will examine LETS [52] as well. We use RQDI to represent the state of the art in 2-phase, single-bit QDI links.

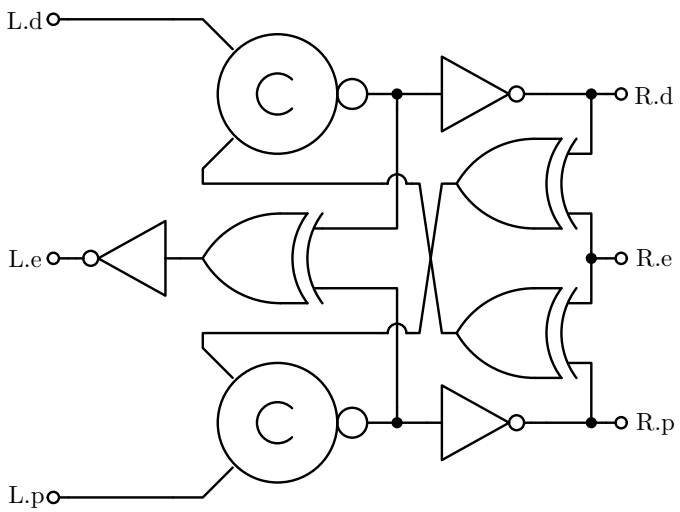


Figure 6.3: RQDI Buffer

6.2.3 ATLS

Asynchronous Ternary Logic Signaling (ATLS) [23,61] is a 4-phase, QDI signaling protocol with a ternary delay-insensitive data encoding. This encoding compacts the dual-rail data wires into a single wire. V_{DD} encodes a *true* token, *GND* a *false* token, and $\frac{1}{2}V_{DD}$ represents the *null* state of the dual-rail encoding. The half-swing encoding reduces the energy cost of data rail transitions, which is attractive as a power saving measure but lowers static noise margins. The enable rail is still full-swing. ATLS simultaneously attacks the problem of limited wiring resources and power consumption, hence its inclusion in our study.

Our implementation of ATLS differs from the proposed circuits in [23] and [61] as the proposed ternary decoding structures have not scaled well into deep submicron technologies. As in the original proposed circuits, we assume a $\frac{1}{2}V_{DD}$ power supply is available and account for it in our power measurements. We use the circuits described in subsection 6.2.5 to encode/decode the ternary data rail. Since

ATLS as proposed does not include any pipelining, we use an additional WCHB buffer when necessary as a pipelining element.

6.2.4 STFB

The Single-Track Full Buffer (STFB) [24] is designed for throughput. It uses a 2-phase, return-to-null (RTN) protocol with no control wires. It is, however, dual-rail, using a total of two wires to transmit a single bit. An upgoing transition on the *true* (*false*) rail encodes a *true* (*false*) token, and a downgoing transition on the rail signals an RTN. The sending process is responsible for raising a rail and the receiving process is responsible for lowering it. The single-track timing assumption requires that the sender and receiver are not simultaneously driving the rail, to avoid shorting the chip power supplies across a link.

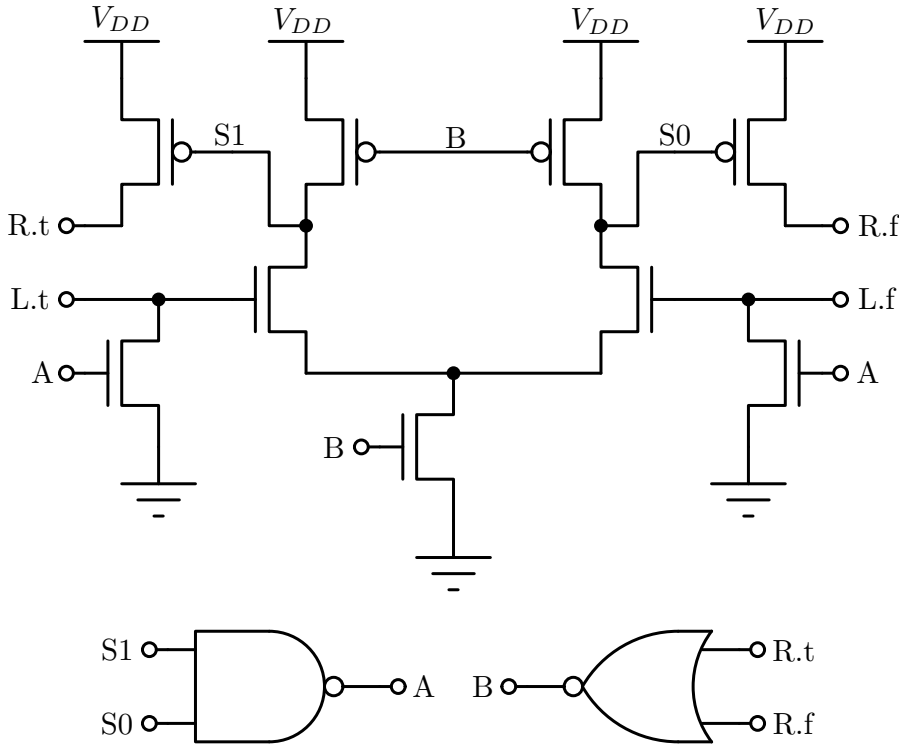


Figure 6.4: STFB Buffer

6.2.5 STATS

Single-Track Asynchronous Ternary Signaling (STATS) is a single-track buffer template of our own design. The design goal was to reduce the total wiring resource requirements to a single wire. It combines the ternary encoding of ATLS with the 2-phase RTN, single-track handshake of STFB. To send a *true* (*false*) token, the sending process sets the wire to V_{DD} (GND). The receiving process returns the state to *null* by driving the wire to $\frac{1}{2}V_{DD}$. As with STFB, the single-track timing assumption requires that the sending and receiving process do not simultaneously drive the link.

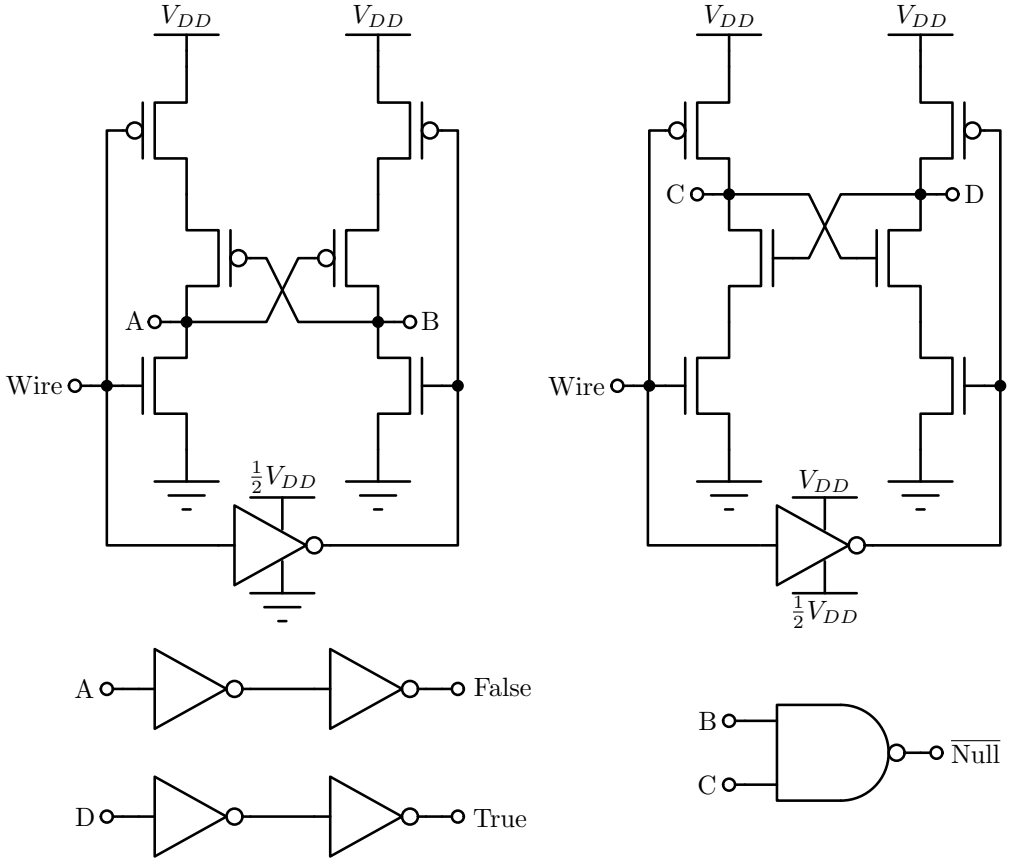


Figure 6.5: Ternary Voltage Decoder

To decode the ternary link, we use the pair of level shifter structures shown

in Figure 6.5. The cross-coupling ensures full rail-to-rail swing, minimizing static power dissipation. While the level shifters are fragile to pathological imbalances in pullup/pulldown network sizings, weakening the pullup/pulldown cross-coupled stacks with respect to their pulldown/pullup counterparts to a ratio of 1:2 is sufficient. Further increasing the drive strength disparity by changing transistor thresholds is recommended. The inverters and NAND gate should be sized to equalize load capacitances on nodes A, B, C, and D.

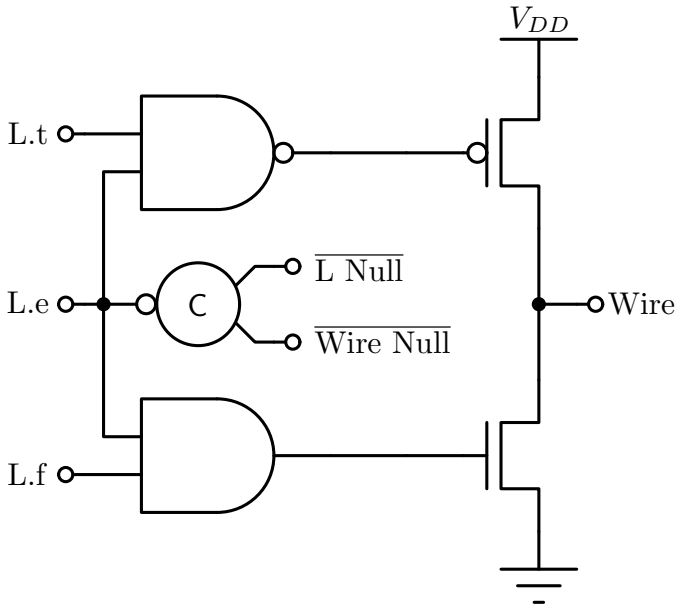


Figure 6.6: STATS Transmit Stage

The link is driven to V_{DD} or *GND* by a single appropriately-sized pMOS or nMOS transistor, respectively, as shown in Figure 6.6. The null calculation for the Wire and L are obtained from the NAND from Figure 6.5 and the traditional NOR dualrail calculation (not pictured), respectively. A parallel combination of one or more of the circuits in Figure 6.7 returns the link to the *null* state at $\frac{1}{2}V_{DD}$. en high starts the RTN process, and True and False are signals from the decoder shown in Figure 6.5. We allow our analysis framework, described in subsection 6.3.2, to permute the combination and sizing of the RTN circuits to fully explore the

tradeoff space.

- **Passgate** (Figure 6.7a): This circuit drives the link to $\frac{1}{2}V_{DD}$ using the least energy, by connecting to the $\frac{1}{2}V_{DD}$ supply. It is the most conservative of the three, but also the slowest.
- **Self-Invalidating Driver** (Figure 6.7b): The self-invalidating driver is the most aggressive of the three designs, as it is essentially a full rail-to-rail transition interrupted halfway. While it offers the best slew-rate (a single RC time constant is more than a $\frac{1}{2}V_{DD}$ swing), it relies on the level-shifter structures in Figure 6.5 to resolve the state of the wire quickly and switch the True/False signals depicted in Figure 6.5 and Figure 6.7b. A slow transition on either of those two signals will result in an overshoot of $\frac{1}{2}V_{DD}$ and potentially a spurious token on the link.
- **Shorted Inverter** (Figure 6.7c): The shorted inverter makes use of the CMOS inverter voltage transfer curve behavior to drive the wire very quickly to $\frac{1}{2}V_{DD}$. It is faster than the Passgate technique, but very energy inefficient as it essentially shorts V_{DD} to GND while enabled.

6.3 Methodology

We constructed a framework to evaluate the various link types described in section 6.2 across a wide range of operating points. Links were studied in two contexts: on-chip planar communication, and 3D signaling through TSVs.

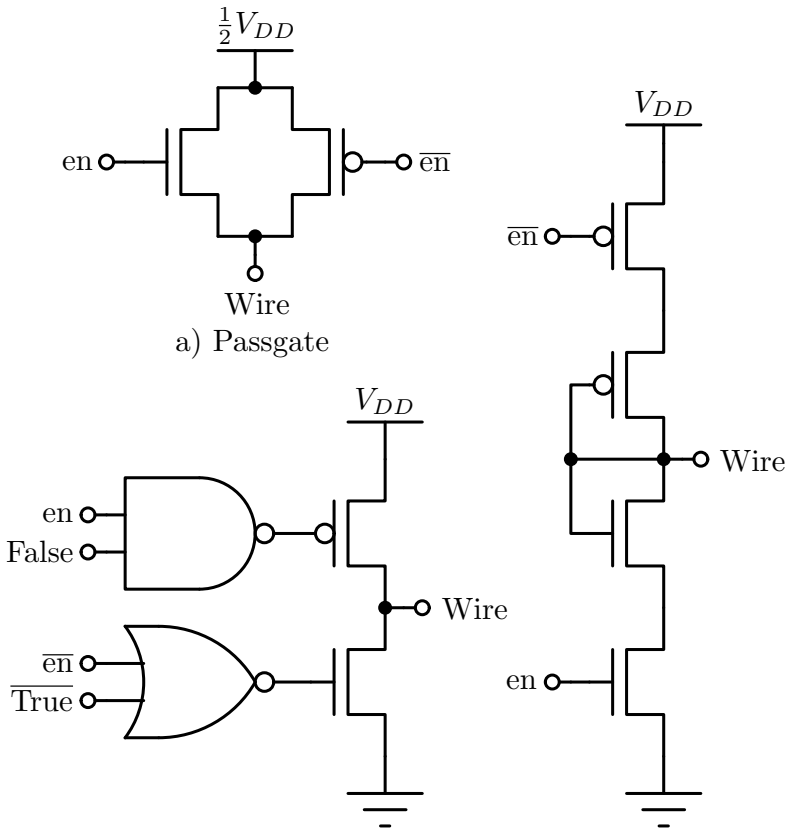


Figure 6.7: Ternary Return to Null Schemes

6.3.1 Link Simulation

We used SPICE simulation to determine the throughput and energy for each link type. Figure 6.8 shows the basic Device Under Test (DUT) for these simulations. Double-headed arrows represent link channels (e.g. STFB); 3-wire channels from the environment are e1of2. "Link TX" and "Link RX" convert to and from the link protocol, respectively, while "Link Buffer" is a native buffer for the protocol. The link DUT is a FIFO pipeline, implemented at the transistor level. It is driven by an environment that generates pseudorandom tokens as fast as the link can accept them.

The link DUT also includes a distributed RC interconnect model (planar wire

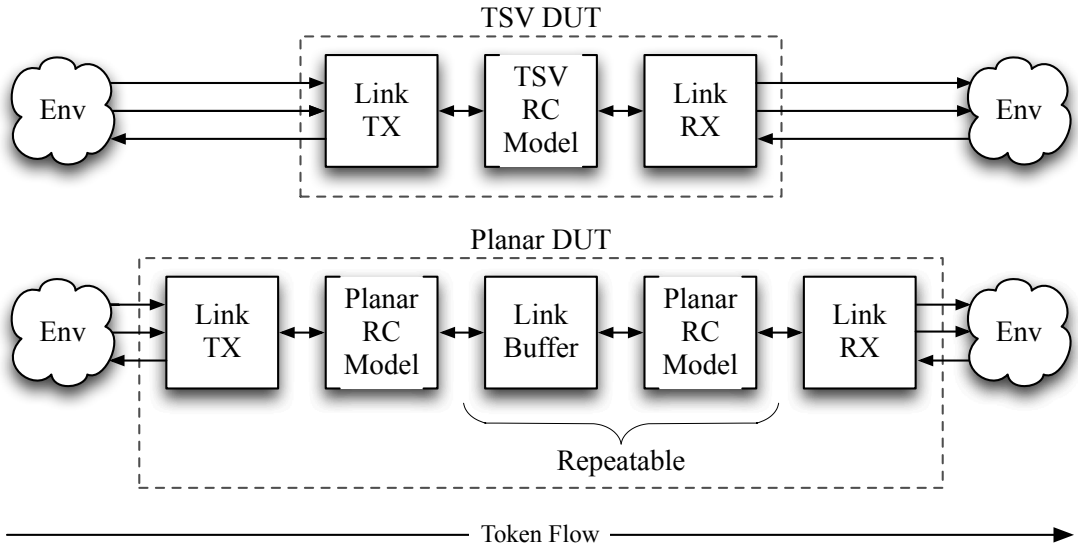


Figure 6.8: Planar and TSV Link DUT

or TSV). Planar wires of a given length may be broken up into several shorter wires by adding extra buffers as pictured. TSV links cannot be so divided, as there is no way to insert a buffer in the middle of a TSV. We discuss interconnect models in more detail in section 6.4.

In our simulations, the environment communicates using e1of2, and the cost of conversion to the protocol used by the DUT is included as part of the energy and area costs of the link. This simulates a fully-asynchronous system where computation is done with islands of 4-phase QDI logic and the links are used to shuttle data across planar links or TSVs [51]. This assumption penalizes 2-phase protocols, but it is generally accepted that 2-phase computation is unwieldy in comparison to 4-phase [53] (with the possible exception of STFB [24]).

Figure 6.9 shows the complete simulation harness. Average frequency is measured using the right-side enable signal, and power dissipation is measured for the DUT alone using a dedicated power supply. Since the environment source and sink are implemented in Verilog, we use two WCHBs to decouple the DUT from

any digital boundary effects. The harness is designed to operate faster than the DUT, so that link throughput is governed primarily by the DUT itself and the RC characteristics of the interconnect.

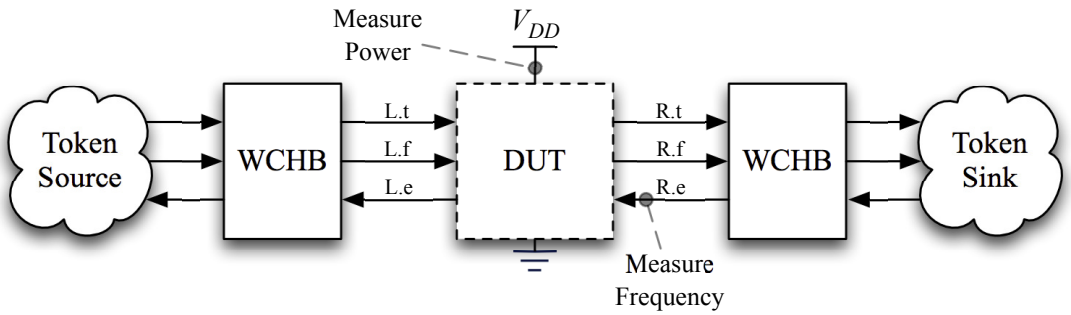


Figure 6.9: SPICE Simulation Harness

ATLS and STATS buffers require an additional $\frac{1}{2}V_{DD}$ supply. In order to be fair, we allow RQDI, STFB, and WCHB links to run at a voltage lower than the harness V_{DD}. To support this, we implemented pipelined level shifters based on the WCHB template, shown in Figure 6.10. These are considered part of the environment, so they are not counted against the link energy and area. The usual protocol converters are still required in addition to these level shifters for non-e1of2 links. The C-elements have one low-swing input and one full-swing input. In the C-elements, the nMOS transistors connected to the low-swing input are LVT and sized double-width, and the pMOS transistors are HVT. All other transistors are standard VT.

6.3.2 Optimization Framework

The goals for our links are to maximize throughput, minimize energy dissipation, and minimize buffer silicon area. Because these measures are not independent, the solution to this multi-objective problem is a Pareto front of different buffer

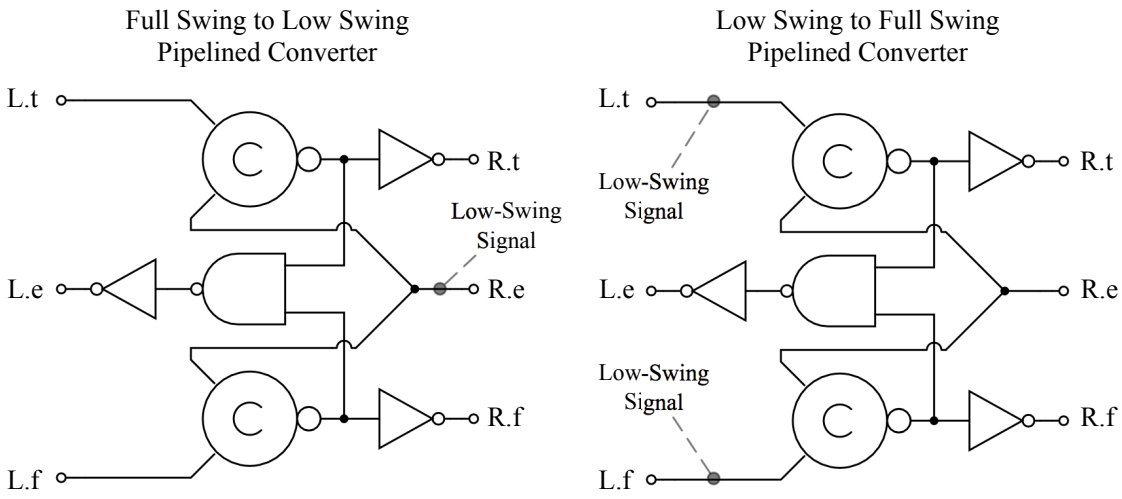


Figure 6.10: WCHB Level Shifters

configurations situated in a three-dimensional tradeoff space between throughput, energy, and area.

To explore this space, we can apply multi-objective heuristic optimization algorithms. While heuristic optimization algorithms are not guaranteed to find the true Pareto front of a given space, i.e. the globally optimal front, in practice a reasonable approximation can be obtained.

We chose the DEAP [27] toolkit and its implementations of the $(\mu + \lambda)$ genetic algorithm[1] (GA) [4] and the widely-used NSGA-II [13] population selection algorithm. NSGA-II-based genetic algorithms are designed to provide a well-distributed family of points along a Pareto front, allowing us to capture the engineering tradeoffs in the design space. Some commercial tools [8] converge to a near-globally-optimal Pareto front faster than NSGA-II-based algorithms, but the same result can be obtained by NSGA-II given enough design space samples—typically, a few thousand is sufficient, and we sample at least 2850 points in the space for each link.

[1]Two-Point Crossover ($c_p = 0.7$), Gaussian Mutation ($m_p = 0.2$), $\mu = 20$, $\lambda = 60$, $n_{gen} = 60$

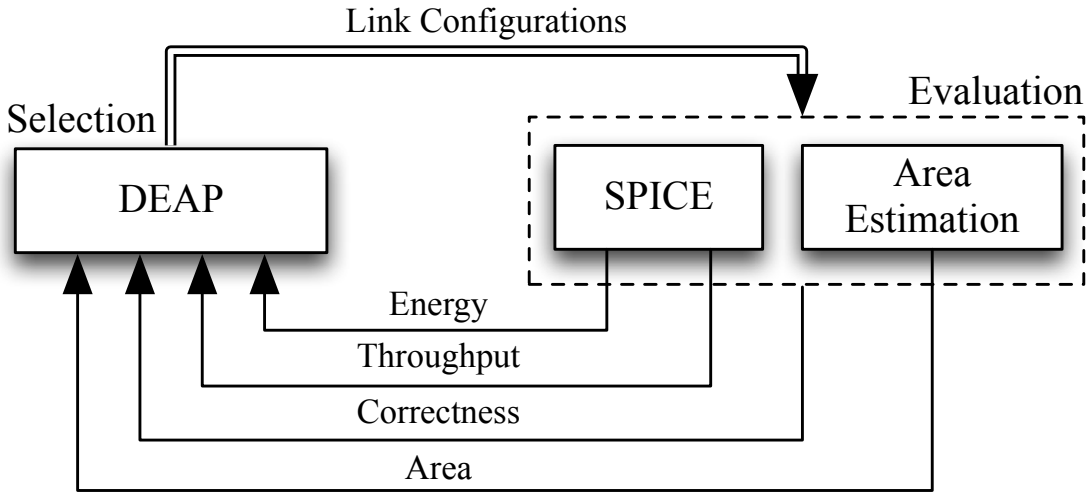


Figure 6.11: Heuristic Optimization Framework

To find the Pareto front for each link design, we built the optimization tool shown in Figure 6.11. Candidate link configurations are selected by DEAP, simulated, and evaluated based on relevant fitness criteria. Throughput and energy are measured using the SPICE harness described in subsection 6.3.1. Area is estimated as the total transistor area, i.e. the sum of $W \times L$ for all MOSFETs in the DUT. While this does not account for routing, etc., it provides a lower bound to make reasonable direct comparisons. Some link configurations may deadlock, send spurious tokens, violate dual-rail encodings, or present other failure modes. The environment checks for this and removes any failing configurations. All these evaluation results are fed back to DEAP and used to direct the selection of further candidate configurations.

It is worth noting that this optimization approach is not limited to on-chip links. The same general framework can be applied to any system with a parameterized configuration (genome) and a set of performance metrics (fitness).

The specific configuration parameters selected by our tool depend on the link type being optimized, and are summarized in Table 6.2. For all link types, the

Table 6.2: Link Configuration Parameters

Link Type	Transistor Sizing	Circuit Topology	Link Voltage
ATLS	✓	✓	
RQDI	✓		✓
STATS	✓	✓	
STFB	✓		✓
WCHB	✓		✓

framework selects transistor sizes for the circuits. Transistor sizing is usually handled by convex optimization algorithms, but since we want to explore the multi-objective space we allow DEAP to choose sizing, from minimum transistor width up to 100 times minimum.

For the ternary link types (ATLS, STATS), the framework can alter the circuit topology by choosing which combination of RTN schemes to use (Figure 6.7). For the other types, it can voltage-scale the link as described in subsection 6.3.1. Finally, in the planar context our tool can vary the number of buffer stages used (Figure 6.8). We account for the area and energy consumed by multiple planar buffers, but not the additional pipeline slack they provide (which may or may not be desirable depending on the specific system).

6.4 Evaluation and Discussion

We evaluate each link on the metrics of throughput, energy per token, and planar buffer area. In order to definitively conclude that one link protocol is "better" than another on these metrics, the Pareto front of the better link must completely dominate the other front—the three dimensional surfaces of the fronts must not intersect. Intersecting surfaces imply that the links being compared are situationally better than one another.

The rest of this section is devoted to examining the throughput/energy/area tradeoff space. To present the data in the most readable format, we have chosen to show two-dimensional projections of the three-dimensional Pareto front, omitting the dominated points on each plot. The bottom right quadrant of each plot represents the most attractive link configurations, as we are trying to maximize throughput and minimize energy/area.

In this study we look at three different technology nodes: a low-power 90 nm bulk process, a low-power 65 nm bulk process, and a high-performance 45 nm Silicon-on-Insulator (SOI) process. While we did not fabricate test structures in all three technologies, we were able to build WCHB and STATS planar links in our 90 nm process. We did not obtain isolated power measurements, but the SPICE-predicted frequency numbers for our test structures were within 7% of the actual silicon measurements. Since we have performed the same technology characterization steps for all three technologies when building our SPICE environments, we are reasonably confident in the simulation results presented in this section.

Our methodology does not directly account for robustness to noise or process, voltage, and temperature (PVT) variation. We provide a qualitative analysis of these factors here, but a complete characterization is pending in our future work.

6.4.1 Planar Links

In our planar link simulations, we model wires using a 100-segment π-model with RC parameters obtained from extracted layout. It is vital to use distributed wire models when studying long links, in order to avoid unrealistically optimistic results [28]. As a concrete example: STATS transceivers sense the state of the wire

locally to determine when to stop driving. A lumped wire model would yield misleading results, since sender and receiver observe the same voltage. In reality, charge relaxation across the wire means that the voltages may differ and the sender may stop driving too soon—this places a restriction on the maximum slew rate possible for a given wire length. Table 6.3 shows the CV^2 energy cost for a single wire transition by technology. This is an estimate, but should provide context for the energy numbers to follow.

Table 6.3: Planar Wiring Energy Costs

Tech [pJ]	Energy (CV^2) [pJ]
90	0.759
65	0.204
45	0.065

Figure 6.12 shows the energy/throughput Pareto front for each buffer type in a 90 nm process. Each point in the front represents a different buffer configuration (transistor sizing, V_{DD}, number of buffer stages, etc.). The relative merit of each link type is similar across process technology generations, so we omit the 65 nm and 45 nm plots for brevity. Our evaluation framework allows us to examine the configuration of each individual point on a Pareto front and uncover Pareto-front-wide trends.

From Figure 6.12, we can see that RQDI and STFB are more energy efficient than WCHB with only a few exceptions. This is unsurprising, as 2-phase protocols like RQDI and STFB expend less energy by halving the number of transitions on the RC link. STFB goes one step further by removing the acknowledge wire and the associated drive circuitry, offering additional energy savings. STATS and ATLS are almost completely dominated by the full-swing protocols (RQDI, STFB, and WCHB). The obvious conclusion for the designer is that ternary signaling is a poor choice for planar wiring, which essentially behaves like an RC lowpass network and

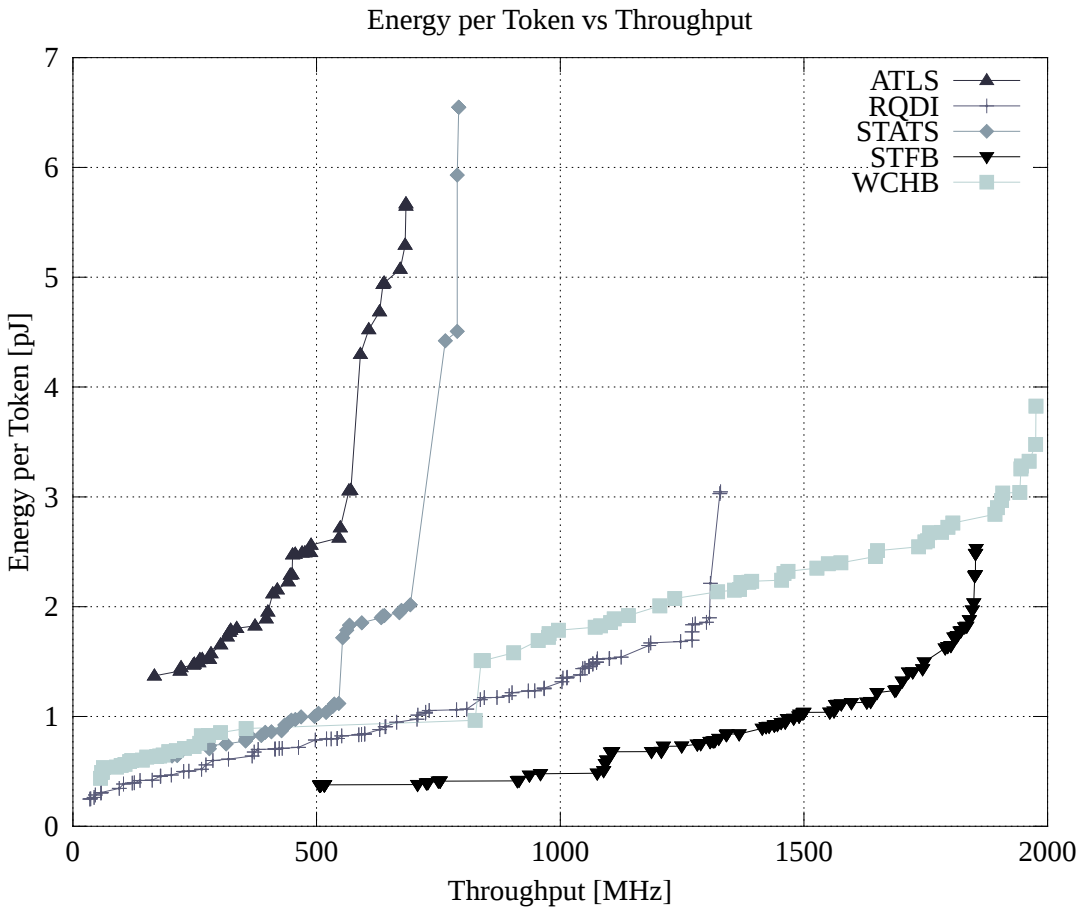


Figure 6.12: Energy vs Throughput, 90 nm 1000 µm Link

limits the throughput of low-swing signals. This is borne out by the fact that the high-throughput Pareto-optimal points for RQDI, STFB, and WCHB all run at full V_{DD} for all technologies, in spite of the capability to aggressively reduce V_{DD}.

Comparing the energy numbers in Figure 6.12 against the estimated CV^2 values in Table 6.3 shows that the numbers are actually *lower* for some links in 90 nm. This is not an error. The estimates in Table 6.3 assume that we are operating at the nominal V_{DD} for the relevant technology, whereas the low-frequency points in Figure 6.12 are operating at *reduced* V_{DD}, which has a quadratic relationship with energy. As we increase V_{DD} to near or at nominal, we see that the energy cost is more than the CV^2 value. This has two contributing factors, the first being the

fact that more than one wire is switching, so we would expect NCV^2, where N is the number of switching wires. The second factor is that the circuits themselves are RC units and cost energy to transition.

We see the circuit cost very clearly in the ternary links. While ATLS has a full-swing enable signal, every STATS transition is a half-swing transition. We would expect a $4\times$ reduction in energy as a result, yet the lowest energy point for STATS in Figure 6.12 barely skirts the upper side of 0.759 pJ. This reflects the fact that the energy cost of voltage level conversion in ATLS and STATS is quite high, especially when replicated many times in a multi-hop link. The sharp increases in energy per token in Figure 6.12 represent the addition of more buffers on a planar link. Examining the trends across links, STFB and WCHB gradually increase the number of buffers on the link as throughput increases—more buffers driving shorter links allows for higher frequencies. Conversely, STATS and ATLS increase the number of buffers only if aggressive transistor sizing is unable to achieve additional throughput. Figure 6.12 demonstrates the much greater energy cost of adding buffers to a STATS or ATLS link compared to a similar addition for STFB or WCHB. RQDI also mainly uses transistor sizing to achieve higher throughput. The vertical jump in energy and area at the very highest throughputs represents the addition of more buffers when aggressive sizing is not enough.

Figure 6.13 shows the area/throughput Pareto front for each buffer type in our 90 nm process. The aggressive sizing of STATS and ATLS buffers can be seen here—the almost 100 µm^2 increase in area around 400 MHz and 550 MHz represents the addition of a single ATLS or STATS buffer stage, respectively. STFB is best in area, as it has the lowest transistor count per buffer of any link.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibu-

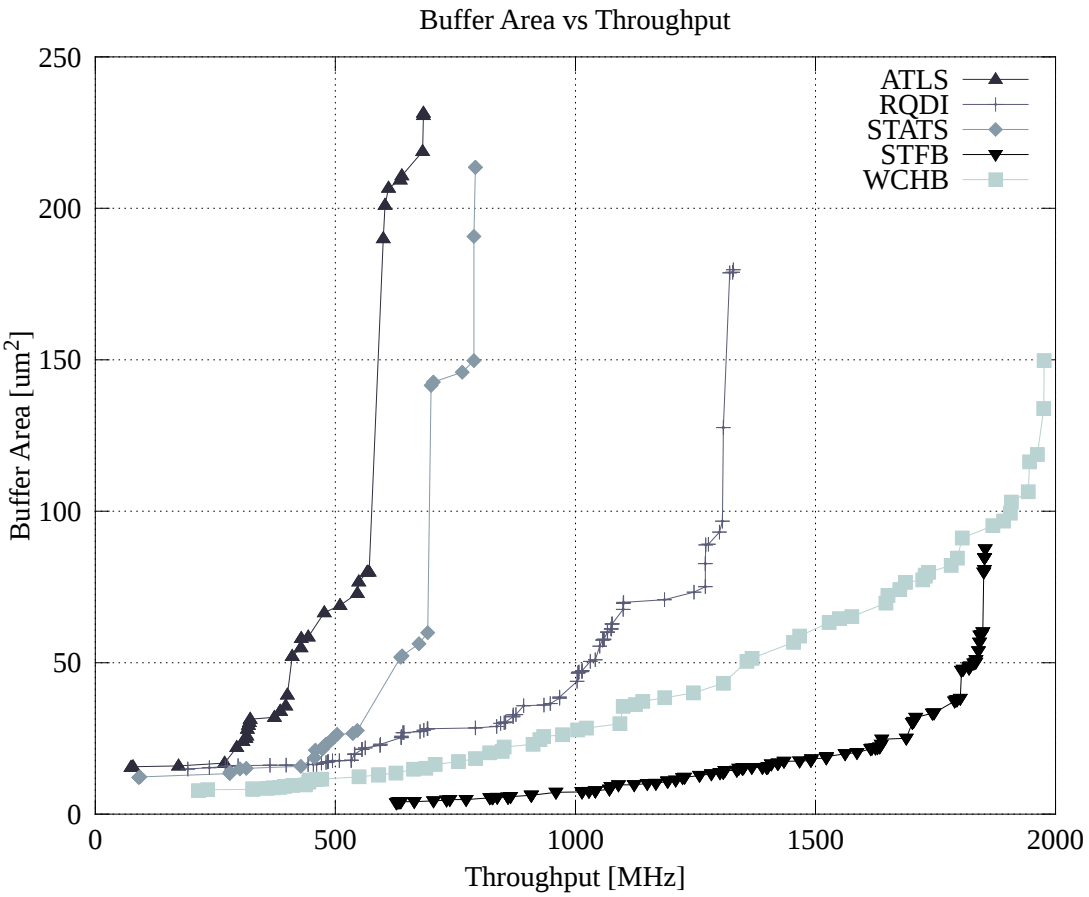


Figure 6.13: Area vs Throughput, 90 nm 1000 µm Link

lum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Just checking if you are actually still reading. As discussed in subsection 6.2.5, the Passgate RTN scheme is used for low-throughput, energy-efficient STATS and ATLS configurations, while the faster, more aggressive Self-Invalidating Driver is used in high-throughput links. For average throughput, a mix of these two schemes is used. The Shorted Inverter RTN scheme is only used for the highest throughput link configurations, where energy costs are already high.

As a general observation (that also holds for TSV links as seen in subsection 6.4.2), ATLS is never optimal and almost always dominated by every other buffer. The link, as proposed by [23], is more of a data encoding than an actual link design. While we improve some of the circuit designs as described in subsection 6.2.3, we still use WCHBs as a pipelining element. Including WCHBs in series adds extra transitions/cycle and power, adversely affecting throughput and energy. In an attempt to maximize the frequency, DEAP selected large transistor sizes, which makes ATLS look unattractive in area as well. A redesign of ATLS that combines the pipelining element with the encode/decode structures could improve its Pareto efficiency performance.

Figure 6.14 and Figure 6.15 show composite Pareto fronts across all technology nodes, for energy/throughput and area/throughput respectively. In other words, the curves on these plots represent the best buffers in that technology at each given operating point. In order to compare results across technology nodes, we scale link length by the technology feature size. Results presented below are for a link length of 20,000λ, equivalent to 1000 µm in a 90 nm technology.

The results are consistent across technologies: STFB buffers are the most energy- and area-efficient for planar signaling across most of the range. At the very highest throughputs WCHB (and 45 nm RQDI) buffers continue to operate

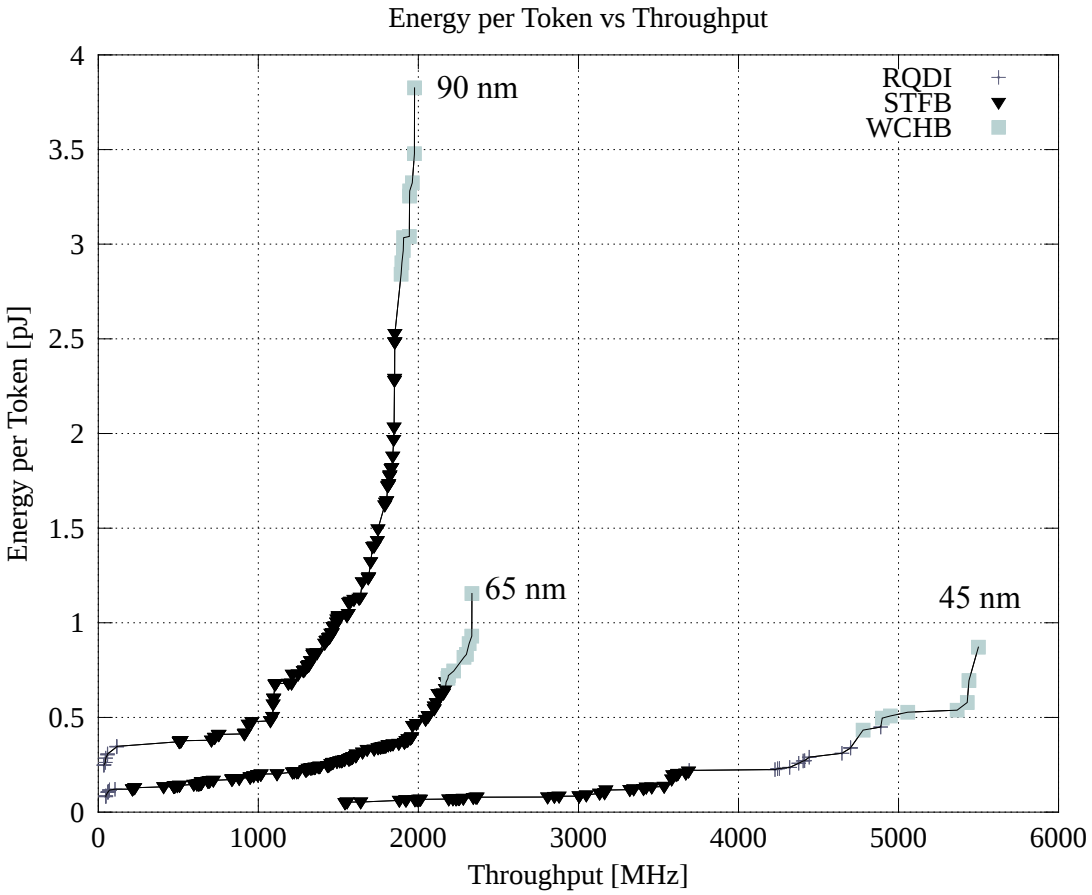


Figure 6.14: Energy-Throughput Composite Pareto, Planar

after STFB fails, but at a greatly increased cost in energy per token. This high energy is due to aggressive transistor sizings, reflected in extravagant area usage as shown in Figure 6.15. From these results alone, STFB is the clear winner in the planar context for all but the most aggressive throughput targets. However, the single-track timing assumption makes STFB less robust than QDI buffers, as we discuss in subsection 6.4.3. This presents a tradeoff to the designer between energy/area usage and ease of design. The additional cost of "robustness" is not prohibitive, as can be seen by comparing STFB against the QDI buffers (WCHB for high throughput, RQDI for lower) in Figure 6.12 and Figure 6.13.

In the planar context, designers also have control over interconnect wire spac-

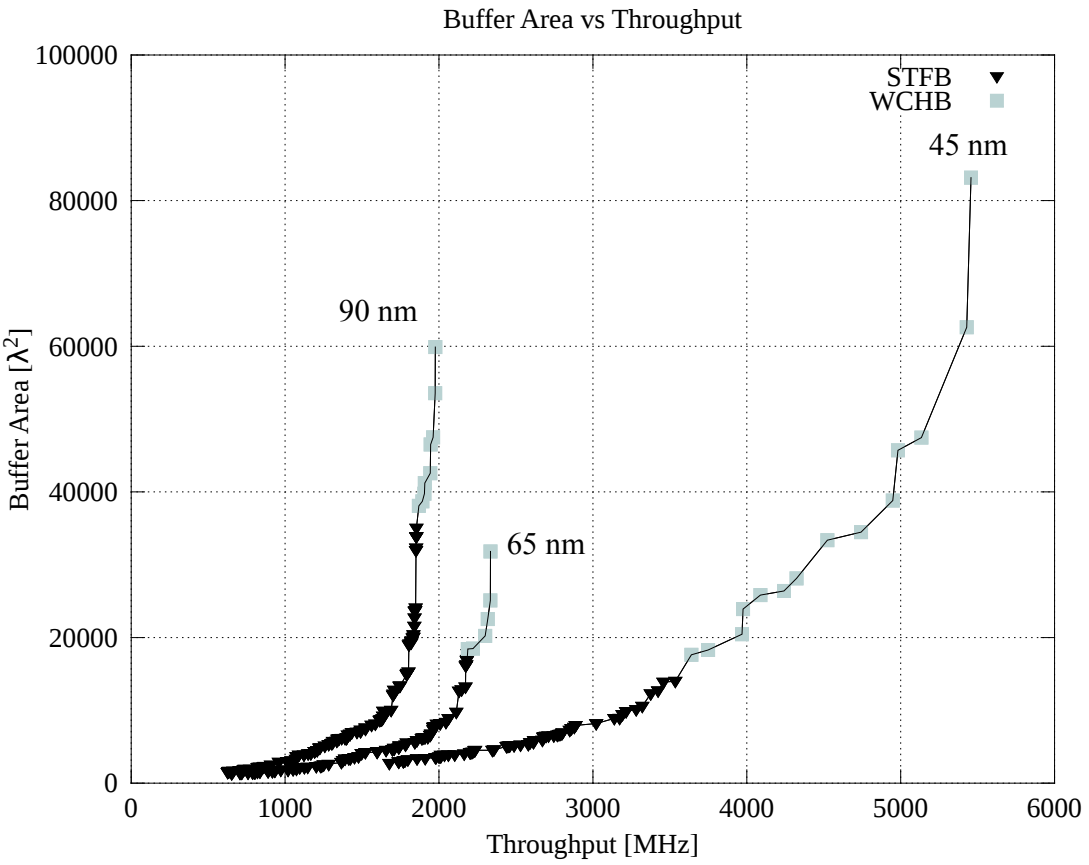


Figure 6.15: Area-Throughput Composite Pareto, Planar

ing, which has a direct effect on coupling capacitance. We found that a change from minimum to sparse spacing (twice minimum) chiefly impacted energy per token. For brevity, we report the average energy improvements at a given frequency for each link in Table 6.4, as opposed to including additional Pareto fronts. Note that this table does not capture the additional benefits of reduced crosstalk due to the increased spacing.

Table 6.4: Sparse Wiring Energy

Link	90 nm	65 nm	45 nm
ATLS	47.36	16.93	-24.67
RQDI	33.71	7.22	13.98
STATS	27.42	-92.28	-112.87
STFB	39.04	18.11	12.26
WCHB	49.66	28.43	20.99

For most link/technology parings, the results shown in Table 6.4 are as expected. To first order, wire resistance remains constant with increased wire spacing while coupling capacitance decreases. This decrease in capacitance leads to lower $\frac{1}{2}CV^2$ energy dissipation.

Strangely, the sparse wiring energy *increases* for STATS and ATLS. The root cause of this energy increase is that for high-throughput link configurations, DEAP selects more highly pipelined links. As an example, in 45 nm, the fastest running ATLS and STATS configurations divided the planar wiring into 10 sections for the sparse wire spacing case, and only 5-6 for the minimum spacing case.

In order to implement single-track timing (sender and receiver must not drive a wire simultaneously), STATS inspects the local voltage to determine when to cut off the driving transistors. If the interconnect resistance is high relative to its capacitance (as in sparse wiring), the buffer may see the local voltage change and turn off before moving enough charge to resolve the state transition at the remote end of the wire. This tends to favor shorter wires with more buffers, leading to greater energy consumption. High-throughput ATLS configurations use the fast Self-Invalidating Driver, which has the same property.

6.4.2 TSV Links

To simulate 3D links between stacked dies, we use the TSV model from [73], modified to have distributed rather than lumped RLC components. It represents a 20 µm diameter copper TSV with 25 µm pitch in a digital process. We also model coupling capacitance to Manhattan neighbor TSVs. TSV fabrication is usually a separate step from the rest of the CMOS process and scales at a different rate, so

we use the same TSV model for all process technologies in this study. The energy costs per TSV are shown in Table 6.5 as a function of V_{DD}, which is different by technology.

Table 6.5: TSV Energy Costs

TSV Voltage [V]	Energy (CV^2) [pJ]
1.2	0.216
1.0	0.150

Because TSV pitch is much larger than the standard via pitch, we assume that TSVs are a scarce resource. As a result, we report throughput per-TSV below (by scaling using wire counts from Table 6.1). This penalizes buffers that require more wires to send a single bit of data.

Figure 6.16 and Figure 6.17 show the energy/throughput and area/throughput Pareto fronts in a 90 nm process for each buffer type communicating vertically through a TSV link. Since buffers in each technology must drive the same TSV structure, reported buffer area is not referenced against λ as it was for some of the planar results.

In the TSV context, STATS is a strong contender due to its efficient use of TSV resources. Furthermore, TSVs have high capacitance but low resistance compared to planar wires, due to the sheer amount of conductive material. This environment approaches the ideal lumped capacitance case where a low-swing link such as STATS excels—theoretically, a half-swing protocol would expect to see 4x savings in $\frac{1}{2}CV^2$ switching energy. In practice, the ternary conversion energy cost cuts into this savings, but STATS is more attractive in energy/throughput-per-TSV than all other links save STFB.

In fact, all of the buffer types have energy costs well in excess of the estimated

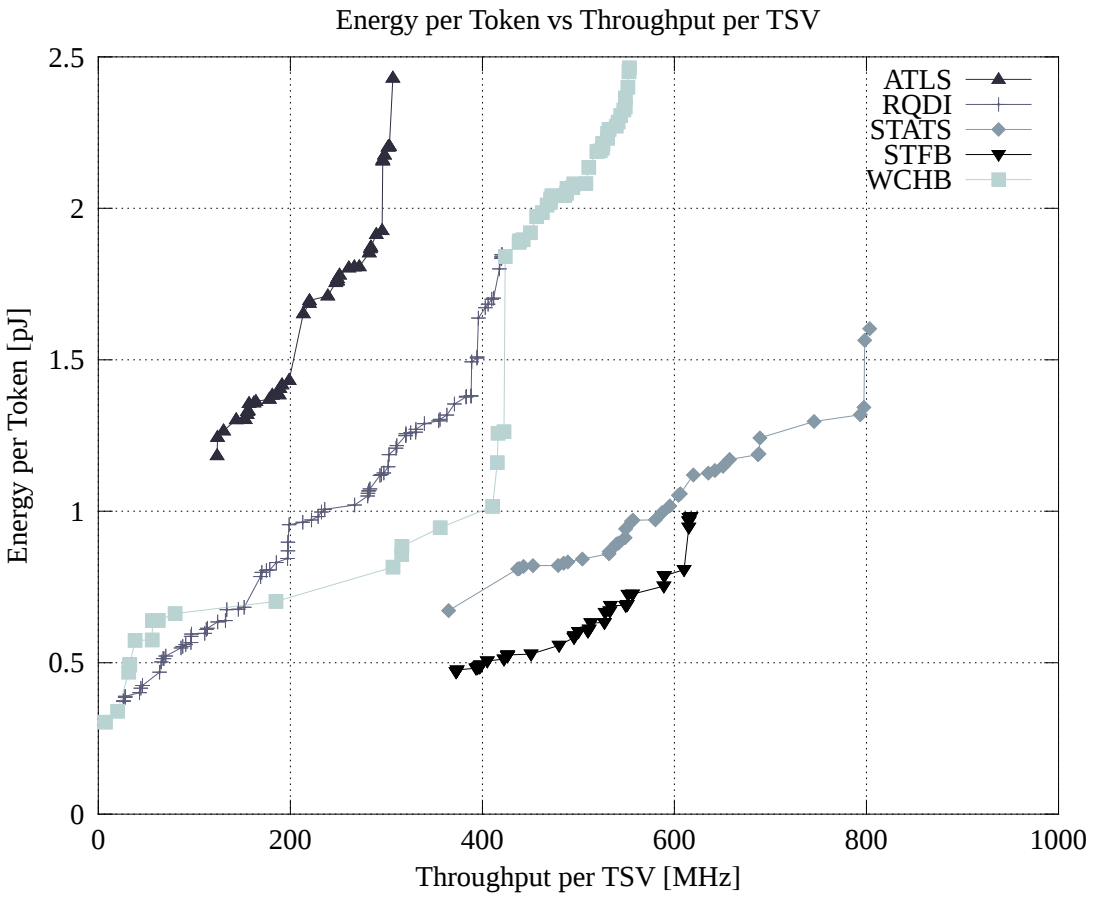


Figure 6.16: Energy vs Throughput, 90 nm 25 µm TSV Pitch

0.216 pJ shown in Table 6.5. Much of this energy cost is due to multiple switching TSVs, e.g. a WCHB TSV link will switch two TSVs per token. However, STFB and STATS switch only one TSV per token, and this is reflected in them having some of the lowest energy points of all the links. We can also see the effects of reduced V_{DD} on energy, especially for the RQDI link.

An interesting phenomenon is the sharp energy increase for WCHB buffers in Figure 6.16 around 400 MHz. Examining the link configurations that straddle this increase revealed essentially identical configurations save for one gate: the inverter driving the returning L.e acknowledge signal (shown in Figure 6.2) for the Link RX unit (shown in the TSV DUT section of Figure 6.8). The higher-energy

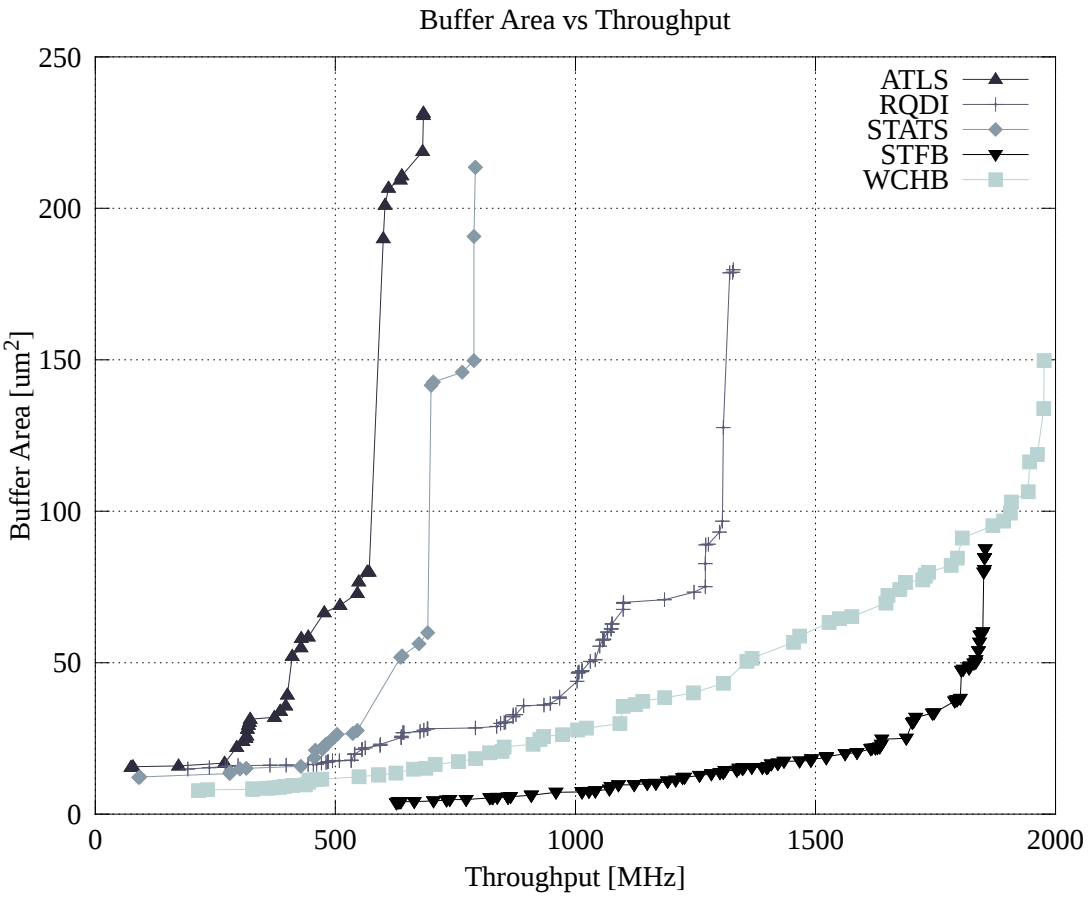


Figure 6.17: Area vs Throughput, 90 nm 25 µm TSV Pitch

configuration was a *maximal* sizing of this inverter, whereas the lower-energy point was a *minimal* sizing of the same inverter. This phenomenon is present in all three technologies. It occurs in the planar case as well: the slight discontinuity in WCHB energy per token seen in Figure 6.12 around 800 MHz displays the same jump in driver sizing. There are other effects at work in the planar case (e.g. number of planar buffers), but the trend is still noticeable.

While further investigation is warranted, this suggests two Pareto efficient operating regimes for the WCHB link. In the first mode, throughput is unaffected by a slow transition on the enable signal—the link is not token-hole-limited. At some throughput threshold, however, the system becomes token-hole-limited and a

fast acknowledge transition (with associated energy cost) is required to see further improvement. Examination of the dominated points in the DEAP runset revealed that DEAP had tried many similar configurations, more or less holding all other parameters constant and varying the sizing of the L.e inverter across the range of allowable sizings—in other words, this phenomenon is quite unlikely to be an artifact of the heuristic optimization algorithm. Intuitively, a small increase in the inverter sizing would offer negligible throughput gains with an energy penalty. Conversely, a downsizing of a maximal inverter would penalize throughput without much benefit to energy. Furthermore, it is likely that an algorithm that sizes transistors based on their electrical environment alone would not have discovered these two operating regimes. Such an algorithm would have sized the L.e inverter to drive the large TSV capacitance and missed out on the low-energy WCHB configurations.

A cross-technology examination of TSV links, plotted in Figure 6.18 and Figure 6.19, is slightly more complicated than the planar scenario. We use the same TSV structure across all technologies, so the electrical characteristics of the physical link remain constant while the transistors shrink. This leads to STFB failure (described in subsection 6.4.3) and its disappearance from the Pareto front after 90 nm.

Measured on a throughput/TSV basis, STATS (which uses only one TSV) dominates. The QDI buffers (RQDI, WCHB) are penalized due their 3-wire interface, but also appear on the Pareto fronts at low throughput/TSV.

Examining silicon area (Figure 6.19) instead of energy per token, WCHB (not RQDI) is the smallest for low per-TSV throughput, for similar reasons to the planar case. The other rankings are the same as for energy.

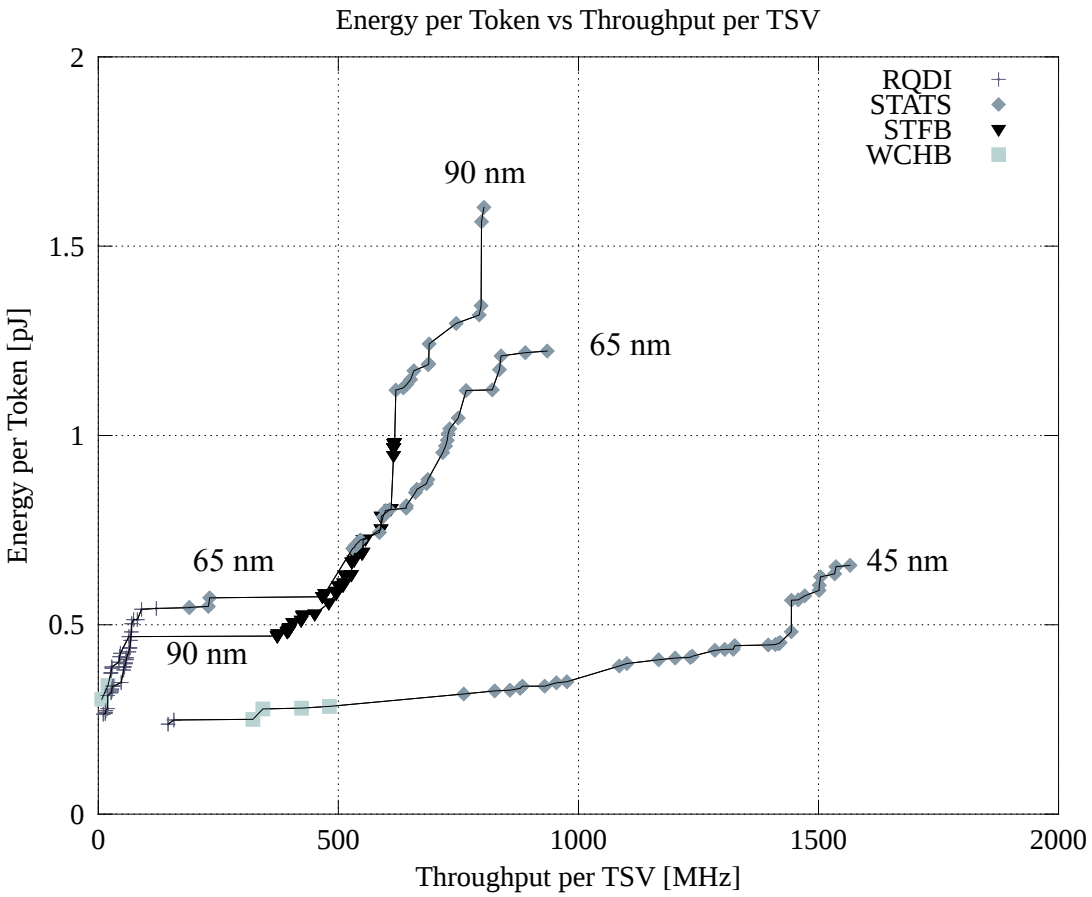


Figure 6.18: Energy-Throughput Composite Pareto, TSV

6.4.3 Link Failures and Reliability

In addition to throughput, energy, and area, we record the failure rate of individual configurations selected by DEAP. These statistics are reported in Table 6.6 for planar and TSV links across our three process technology nodes. As discussed in section 6.3, we verify that links send tokens correctly and do not deadlock. Failures are typically due to poorly-sized transistors driving large RC loads, since DEAP can choose sizes at random. Roughly speaking, these failure rates provide information about how easy a link is to design and how robust it is to sizing variation. While a significant amount of additional work is required to quantify link robustness, we believe the failure rate is of use in building an intuitive understanding of a link's

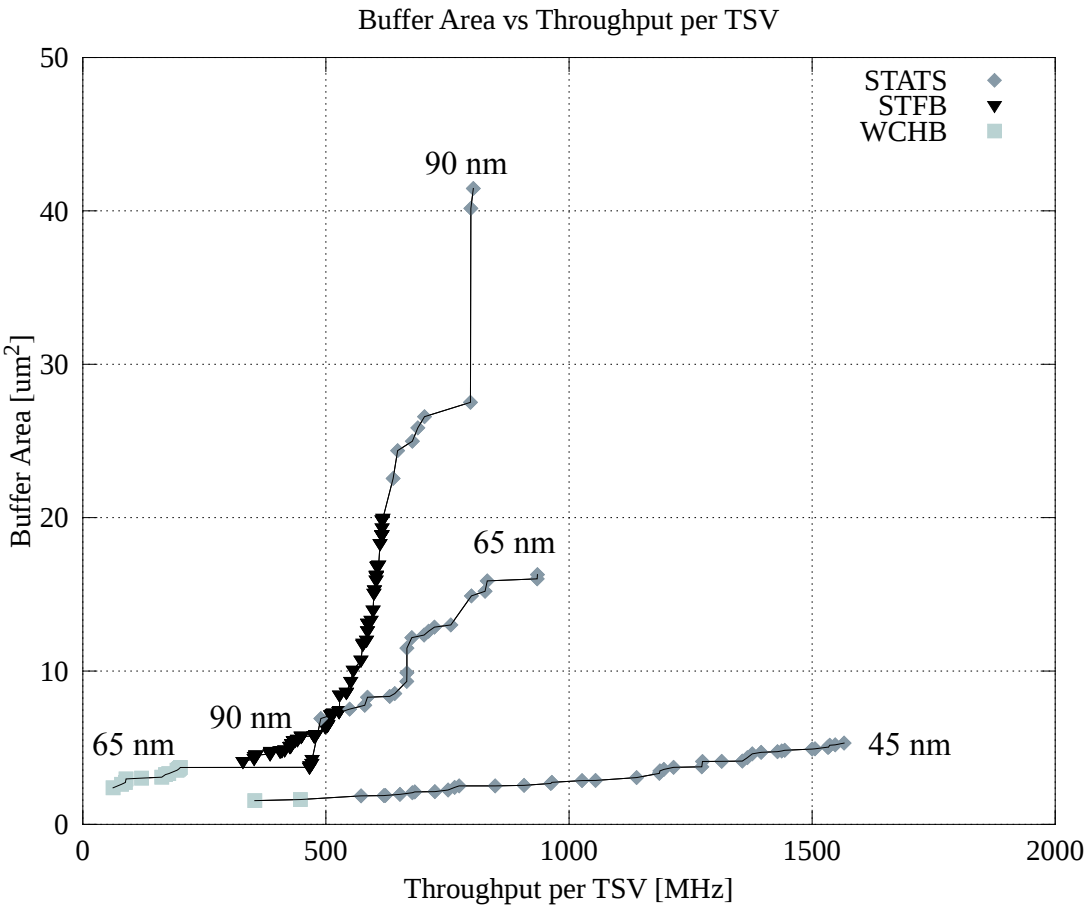


Figure 6.19: Area-Throughput Composite Pareto, TSV

timing assumptions and relative design difficulty.

Table 6.6: Link Failure Rates

Link	% Planar Failure			% TSV Failure		
	90 nm	65 nm	45 nm	90 nm	65 nm	45 nm
ATLS	23.94	16.34	19.23	17.72	20.83	15.54
RQDI	25.60	23.93	17.80	19.72	21.52	24.68
STATS	42.40	36.26	45.45	33.26	33.96	33.31
STFB	28.18	21.99	33.63	29.19	99.33	100.00
WCHB	10.67	8.49	12.43	12.79	12.80	25.32

Note: $2856 \leq n \leq 11158$

STATS has the highest failure rates of all buffer types in planar and the second highest in the TSV context. This is not surprising, as STATS combines both ternary encoding and the single-track timing assumption to achieve its single-wire

goal, and each of these techniques reduce reliability compared to a delay-insensitive link.

Ternary decoders (Figure 6.5) are particularly sensitive to sizing variations, and their failure prevents the buffer from sensing the link state correctly. Even an accurate but slow decoder may cause link failure, for example by causing a Self-Invalidating RTN Driver (Figure 6.7b) to overshoot $\frac{1}{2}V_{DD}$. This impacts the failure rates of both STATS and ATLS.

As discussed in subsection 6.4.1, single-track timing can cause STATS to fail if the interconnect resistance is too high (planar wiring). This is less of an issue in the high-capacitance, low-resistance TSV context, so we see correspondingly lower failure rates in Table 6.6. ATLS, RQDI, and WCHB do not suffer from this problem, due to the QDI timing of their handshake. Reasonably slow transitions are acceptable, as they will be not be acknowledged until the receiving end can resolve the wire state.

STFB uses an even more aggressive single-track timing assumption. The STATS level shifter structures offer a better inspection of the wire state due to hysteresis, whereas the link wire directly drives the STFB handshaking logic as seen in Figure 6.4. As a result, STFB has simpler circuits and better throughput, but at the expense of robustness. The traces shown in Figure 6.20 were selected from the fastest five STFB and STATS TSV link configurations in 90 nm. V_{DD} is 1.2 V. Times shown are matched by sent tokens. The STFB *true* and *false* rails do not complete full-swing transitions. Examining the figure, it takes at least two tokens traversing a link (and driving the link pulldown network) to return the wire state to *GND*. In contrast, STATS transitions cleanly between V_{DD}, $\frac{1}{2}V_{DD}$, and *GND* because it inspects the voltage before cutting off the transistors driving the

wire.

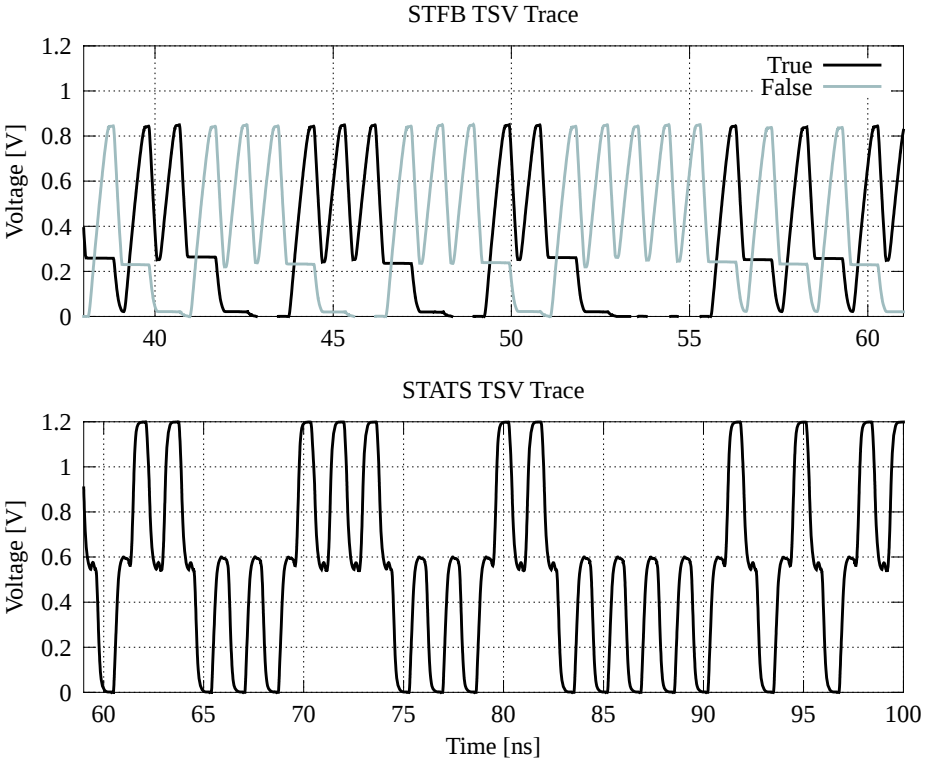


Figure 6.20: STFB and STATS Waveforms, 90 nm TSV

In short, STFB is releasing the pull-up network and pull-down networks too early. Large capacitances exacerbate this problem. Because the TSV RC characteristics in our model do not scale with technology—we assume they are separately fabricated—but transistor switching speed does, the timing margins become progressively worse for STFB with each smaller technology. This is reflected in the increased number of failing configurations, shown in Table 6.6, and the complete failure of STFB to drive the TSV link in 45 nm.

One side effect of this timing failure is that STFB becomes a de facto low-swing signaling protocol, which artificially improves its energy efficiency. While this is certainly not without merit, it comes at the cost of noise margins and robustness. While we do not model noise sources, the STFB traces shown in Figure 6.20 are

more susceptible to noise than a full-swing signal would be. Note that ternary encodings (ATLS, STATS) also have reduced noise margins compared to a full-swing signal.

6.5 Criticisms

This chapter is largely based on a published paper. As a result, I have access to peer review feedback, and have tried to work the feedback into this section.

This is like a middle-school science fair project, none of it is novel!

I make no claims that the links presented in this chapter are novel, but thank you for the accurate review of my work. You are correct, insofar that many things are not novel. In fact, I cited the relevant works. The only novel things are STATS, which I will detail in chapter 7, and the application of my methodology in this evaluation. To my knowledge this is the only published paper that evaluates self-timed single-bit links in this fashion.

Your methodology is biased and flawed, therefore this is useless.

This is simultaneously both a reasonable criticism as well as a standard attack on the work of others. Yes, there are biases built into the harness. If I had infinite time and infinite resources I would have built proper protocol-specific harnesses in Verilog. As it stands, I believe the decisions I made are well-justified in context, and the biases do not swamp the general trends we observed. I have striven to not misrepresent my assumptions in my work, even at the cost of weakening my arguments. Are the methodology and its implementation flawed? Yes, but I do not believe that they are flawed to the point of uselessness. Close, perhaps, but we can at least leverage them

to gain some insight into the tradeoffs between the links presented in this chapter.

6.6 Executive Summary

We studied five self-timed single-bit signaling protocols with widely varied properties (timing assumption, wire count, voltage swing), including our proposed STATS single-wire link design. We developed a multi-objective optimization tool to evaluate the performance (throughput, area, and energy per bit) of these protocols for both traditional planar wiring and 3D inter-die communication using TSVs.

Pareto front analysis is a powerful framework for evaluating competing objectives. From this study, we draw several conclusions useful for circuit designers. For planar links, STFB offers the best performance across the range of process technologies studied, though its tight timing requirements do not cope well with non-ideal wires. WCHB is also a good choice, trading some energy and area for increased robustness. STATS performs poorly with planar wiring but makes efficient use of scarce TSV resources, and is a good match for TSV electrical characteristics.

Future work will include a quantitative approach to characterizing signaling protocol robustness, especially for single-track and ternary links. We also plan to expand our analysis to include multi-bit links.

CHAPTER 7

SINGLE-TRACK ASYNCHRONOUS TERNARY SIGNALING

> Arrogance is a veneer—a thin covering of excuses hiding deep performance deficiencies.
>
> Bob Lewis
> "The Seven Deadly Sins of Information Technology"
> *Infoworld*, April 20, 1998

7.1 Introduction

With the pressure to reduce power consumption [35] and the difficulties associated with cross-chip clock skew [14], the need for a robust, low energy on-chip link has become a priority. Asynchronous circuits have a number of attractive characteristics such as robustness to thermal, voltage, and process variations, various degrees of delay insensitivity, and in some cases, reduced power. However, there are a number of overheads, particularly with respect to the encoding of data on wires.

One of the major overheads of asynchronous systems, in particular Quasi Delay-Insensitive (QDI) systems [48], is the need for delay-insensitive encodings such as dual-rail codes, often referred to as *1-of-2* codes. For a single bit, dual-rail encoding requires three wires: *true* and *false* for the data value, and an *acknowledge* wire to handle synchronization. In comparison, a single bit in a synchronous system requires only one wire plus a small incremental cost for the clock routing, as the clock signal routing is assumed to be amortized across the entire system. Multi-bit delay insensitive codings, or *M-of-N* codes, e.g. *1-of-4* codes, can reduce pressure on wire resources, but do so at the expense of more complex circuits [5].

In an effort to reduce pressure on wiring resources and, more importantly,

reduce the energy costs of sending data cross- or off-chip, we propose a Single-Track Asynchronous Ternary Signaling (STATS) scheme. STATS compacts a full three wire *1-of-2* encoding, including the acknowledge wire, into a single-wire encoding. In short, using a sender-receiver model, the sender sets the wire to V_{DD} or *GND* to transmit a 1 or 0 respectively, then the receiver resets the wire to $\frac{1}{2}V_{DD}$ to acknowledge receipt of the data. While STATS is a direct, drop-in replacement for 1-of-2 links, the protocol can be extended to implement multi-bit links.

By compacting the data and acknowledge information into a single wire, STATS can achieve symbol densities of 1 bit per wire while still incorporating flow control. In the planar wiring domain, STATS can help offset the wiring costs of traditional delay-insensitive protocols, but it provides the most benefits in scenarios where wiring resources are truly scarce. As such, STATS is a strong candidate protocol for links traversing Through-Silicon Vias (TSVs) or off-chip pins.

We present an analysis of the STATS design, focusing specifically on implementations for TSVs and off-chip pins. We briefly cover ternary logic and the single-track timing assumption used in STATS in section 7.2. A high-level sketch of the STATS design can be found in section 7.3. A detailed discussion of the circuit implementation as well as some alternative circuit implementations can be found in section 7.4. STATS offers both energy and wire count efficiency for both of these demanding usage cases.

7.2 Background

7.2.1 Single-Track

A single-track protocol is one that makes use of bidirectional communication over a single wire. Typically, these protocols are also two phase, as only two full-swing transitions can be differentiated on a single wire. The terminology of *single-track* does not strictly imply the use of *only* one wire, just that at least one wire in the encoding is used for bidirectional communication.

Figure 7.1: Single-Track Signaling

Figure 7.1 shows a Transmit (TX) and Receive (RX) pair. TX raises the wire high while RX lowers the wire to *GND*. There are three different bidirectional drive schemes: *dynamic*, *static*, and *overlap* [72], which essentially correspond to the degree of assistance the receiving process provides in driving the wire. This interaction is extensible to multiple, mutually exclusive processes on a single wire [72].

TX must not drive the wire high as RX lowers the wire to avoid a crowbar current across the wire. Furthermore, the state of a wire in such an event is not easily predictable, especially for long wires, underscoring the necessity of avoiding simultaneous, opposed drive of a wire. The designer must provide enough of a timing margin such that the TX and RX do not simultaneously drive the wire. The assumption that the margin exists is known as the *single-track timing assumption*. This requirement motivates the existence of timing or feedback circuits to control the release of a wire, i.e. cutting off the drive circuitry.

One additional consideration is the reliability of single-track signaling over long, chip-scale wires. For wires in this length class, the lumped RC abstraction no longer applies—the wires behave more like RC transmission lines. It is possible that TX will drive the wire to V_{DD}, but RX will not detect the transition or vice versa. This failure is especially true if the drive circuitry is controlled by feedback from a local inspection of the wire. The driving process may inject or remove enough charge from the local capacitance to resolve the local state of the link as high or low, and release the wire. The change in charge will diffuse across the wire, but the resultant voltage at the other end of the wire may not be sufficient to be detected as a transition. The problem is only exacerbated by high slew rates on highly resistive links—the local capacitance will be charged very quickly with little to no charge diffusing across the link, resulting in less total charge movement.

This charge relaxation phenomenon restated is simply the case where the delay of the wire exceeds the drive time of the sending process. Techniques to address the issue include using the static or overlap drive schemes, adding repeaters to a single-track wire [17], and altering the relative timing assumptions between link delay and drive time [28].

Systems which use single-track encodings generally fall into two categories: signaling for purely control or synchronization purposes, and the encoding of data. The work of van Berkel et al. [72] implements a single-track protocol for controlling bundled-data pipelines. This was extended in the form of the GasP signaling system for control of fine-grained bundled data pipelines [67]. In both of these systems, a single wire obeying a single-track protocol synchronizes data flow through a bundled-data pipeline.

Single-track templates for encoding data, particularly *1-of-2* and *1-of-N* encod-

ings have been proposed. Nyström developed a self-resetting pulsed logic template which is *1-of-2* encoded [58]. A more general *1-of-N* encoded single-track template, Single-Track Full Buffer (STFB), has also been proposed [24] and has been developed over several generations [25,29].

7.2.2 Ternary Logic

Ternary logic is an instance of Multi-Valued Logic (MVL) where a circuit node has three valid voltage levels that resolve to valid states or symbols. In general, these voltage levels can take on any value, but in order to maximize noise margin and equalize voltage swing, it is standard to use GND, $\frac{1}{2}V_{DD}$, and V_{DD}. In general, this class of ternary circuit offers a number of benefits:

- **Increased Symbol Density per Wire**: GND, $\frac{1}{2}V_{DD}$, and V_{DD} are all valid symbols, whereas binary encodings offer only GND and V_{DD}. While wire density is improved, the circuit complexity for encoding and decoding circuitry increases.
- **Reduced Energy**: switching between GND or V_{DD} and $\frac{1}{2}V_{DD}$ is a half-swing transition, which offers a 4x reduction in $\frac{1}{2}CV^2$ energy dissipation.
- **Delay-Insensitive Encoding**: if V_{DD} and GND represent valid data and $\frac{1}{2}V_{DD}$ represents null or no data, ternary signaling can represent all three legal states in a dual-rail encoding: 0, 1, and Null. Furthermore, a transition between 0 and 1 in either direction must pass through the Null state, as shown in Figure 7.2

We can use the three voltage levels to implement the single wire equivalent of a dual-rail encoding. In other words, the symbols 0, 1, and Null are encoded with GND, V_{DD}, and $\frac{1}{2}V_{DD}$, respectively, as seen in Figure 7.2.

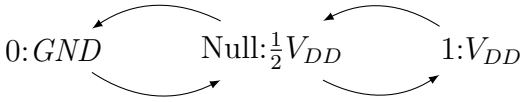


Figure 7.2: Ternary Delay-Insensitive Encoding

To date, there have been several implementations of delay-insensitive ternary signaling encodings. Some systems go as far as to implement ternary logic or computation [47,56], but at the cost of 2-3x in area and 20% in latency versus traditional dual-rail circuits. While these systems offer energy savings in the 20-40% range, the loss in throughput and the area overheads present a significant engineering tradeoff design decision.

Of particular interest are ternary implementations of on-chip interconnect [23,61]. These systems take the form of binary-to-ternary encoders and ternary-to-binary decoders—i.e. a conversion from ternary to dual-rail and back. While this reduces the overall number of wires by 1 compared to a single-bit dual-rail encoding and reduces the voltage swing on the ternary-encoded data wire, the second acknowledge wire is still full-swing. Furthermore, these designs are not complete signaling implementations—-they require additional circuitry to behave as pipelined interconnect. This additional circuitry presents a significant overhead in both energy and area [71].

Onizawa, et al. and Tse, et al. independently propose a single-track ternary links [59,71], both of which behave as pipelined interconnect. Onizawa, et al. show results for 130 nm and Tse, et al. show results in 90 nm, 65 nm, and 45 nm. As single-track ternary links are the focus of this work, we will compare these designs

against one another in section 7.6.

Regardless of implementation details, all ternary systems must address two challenges: driving a node to $\frac{1}{2}V_{DD}$ and resolving the state of the node—i.e. detecting the symbols 0, 1, and Null. We assume the existence of a $\frac{1}{2}V_{DD}$ power supply, which will incur area and package pin overheads for routing. While these challenges are universal to ternary systems, we will discuss them in the context of driving a wire in a ternary link and resolving the state of the ternary link in section 7.4.

7.3 STATS Design

The Single-Track Asynchronous Ternary Signaling (STATS) system implements a complete *1-of-2* handshake using only a single wire. The basic configuration is intended to replace a single-bit delay-insensitive link, but we extend the design to multi-bit.

7.3.1 Overview

The kernel of the STATS system is an TX/RX pair, as shown in Figure 7.3. The TX stage converts binary dual-rail encodings to a ternary link encoding, and the RX stage converts back to a dual-rail encoding. It offers the following features:

- **Single-Track Encoding**: Each STATS TX/RX pair compacts a full 3-wire *1-of-2* encoding into a single wire with a 2-phase handshake on the wire.

- **Statically Held**: Each side of a STATS link holds the state of the link until it is ready to change the link state, similar to the static single-track template of Golani and Beerel [29]. The TX side holds the link at null, $\frac{1}{2}V_{DD}$, until it sends a valid data token, V_{DD} or GND. The RX side holds the link at valid, i.e. either V_{DD} or GND, until it is ready to return the link to null.
- **Half-Swing Signaling**: Unlike most other single-track signaling protocols, STATS is a half-swing signaling protocol—each transition is only a $\frac{1}{2}V_{DD}$ swing.
- **Full-Buffering**: Each TX/RX pair comprises a full buffer, capable of holding a single token.
- **Bounded-Delay Model**: While STATS is not fully QDI, all timing assumptions are related to the delays on the link itself. The critical assumption is that one side of the link can make a local inspection safely before the other side changes the state of the link. The other assumption is that one side of the link will stop driving before the other side begins driving, preventing crowbar current across the link.

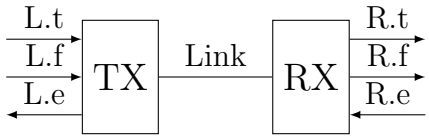
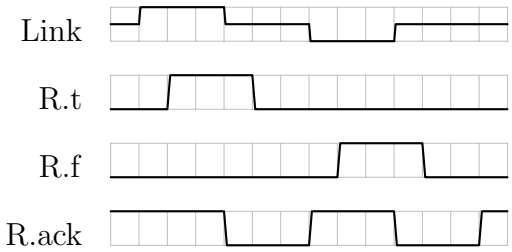


Figure 7.3: STATS Link Structure

Example execution traces for a TX stage sending a true then false token and for an RX stage receiving a true then false token can be seen in Figure 7.4. Note that both the TX and RX stages implement 4-to-2-phase and 2-to-4-phase converters implicitly. It is this conversion that makes a STATS TX/RX pair a full buffer. This conversion can be seen by inspecting the dual-rail encodings {L.t, L.f, L.e}

and {R.t, R.f, R.e} with respect to corresponding Link waveform in Figure 7.4. For a full 4-phase handshake on the dual-rail wires, the Link only transitions twice.

(a) TX Stage



(b) RX Stage

Figure 7.4: STATS Timing Diagram

7.3.2 TX Stage Implementation

The TX stage is comprised of the level-shifter structures in Figure 7.7, the sequencing and drive logic in Figure 7.5, and a NOR2 validity calculation ($\overline{\text{L}_{\text{null}}}$) on L.t and L.f, which is not pictured. As mentioned in subsection 7.2.1, to avoid a situation where both TX and RX try to drive the link simultaneously, the TX stage is designed to tri-state its drivers once it has driven the link to V_{DD} or GND. The L.e signal, calculated by an inverting Muller C-element, serves as the trigger to tri-state the drivers. Once both the input and the link are valid, i.e. the intended symbol has been sent and both $\overline{\text{L}_{\text{null}}}$ and $\overline{\text{Link}_{\text{null}}}$ are high, it is safe to tri-state the driver. We obtain $\overline{\text{L}_{\text{null}}}$ from the aforementioned NOR2 and the $\overline{\text{Link}_{\text{null}}}$ from ternary decode circuits, as described in subsection 7.4.1.

The forward latency of the TX stage is two to three transitions between the arrival of L.t or L.f and a change in state of the link—one transition through the NAND or two through the AND followed by the link drive itself. The backward latency, i.e. the number of transitions until L.e falls is comprised of two to three transitions from the forward path, any transitions in the ternary decode structure, and the C-element transition.

Our TX stage implements the static drive scheme, mentioned in subsection 7.2.1, by statically holding the wire at $\frac{1}{2}V_{DD}$ if there is no valid data to send. If the local voltage on the TX side of the wire is V_{DD} or GND, the TX remains tri-stated to avoid "deleting" an unconsumed token—the RX has not acknowledged yet. The wire is connected to the $\frac{1}{2}V_{DD}$ supply by a passgate driven by the NOR of $\overline{\text{L}_{\text{null}}}$ and $\overline{\text{Link}_{\text{null}}}$. In other words, if there is no valid data to send and the link is null, hold the link at $\frac{1}{2}V_{DD}$. To ensure that the passgate releases the link as soon as possible, the NOR can be implemented with three instead of two pull-down transistors: L.t, L.f, and $\overline{\text{Link}_{\text{null}}}$, leaving the pull-up network unchanged. This saves one transition—the validity NOR calculation.

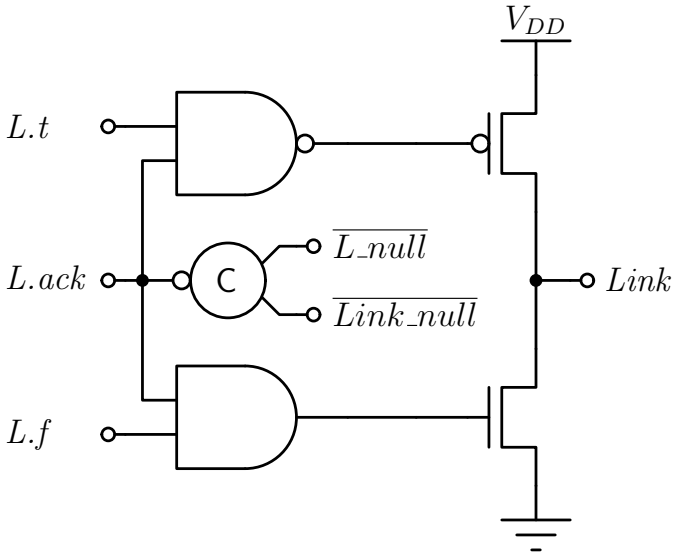


Figure 7.5: TX Stage Sequencing and Link Drive Logic

7.3.3 RX Stage Implementation

The RX stage, like the TX stage, must implement ternary decode structures as described in subsection 7.4.1.

In addition, the RX stage contains the sequencing logic shown in Figure 7.6, a NOR2 validity calculation on R.t and R.f generating R_{null}, and an implementation of one-or-more of the return to null schemes described in subsection 7.4.2. The signals *true*, *false*, and $\overline{\text{Link}_{\text{null}}}$ are calculated by the ternary decode structures. The RX stage also includes a passgate to $\frac{1}{2}V_{DD}$, which is controlled by the chip-level reset signal to set the link to $\frac{1}{2}V_{DD}$ on global reset.

The forward latency of the RX stage, once the link has been set to a valid state, is that of the ternary decode structures and an additional two transitions through the output drivers in Figure 7.6. The backward latency, or the number of transitions until the link is reset to null, is twelve—six for the forward latency, an additional two through a NAND validity calculation on the output, and the transitions on ready and rtn. Note this does not account for the transitions of the return-to-null schemes, which can be anywhere from one to two additional transitions.

This transition pattern implements the full-buffering inherent to the STATS TX/RX pair. Once valid output is produced on R.t and R.f, the link is returned to null. Since the link implements a 2-phase protocol, valid data can then arrive on the link independent of the state of the outputs and R.e. Once R.t and R.f have returned to null and both R.e and ready have returned high, the data on the link is consumed and produced on the R.t and R.f dual-rail wires.

Most of the return-to-null schemes are designed to tri-state the link drivers

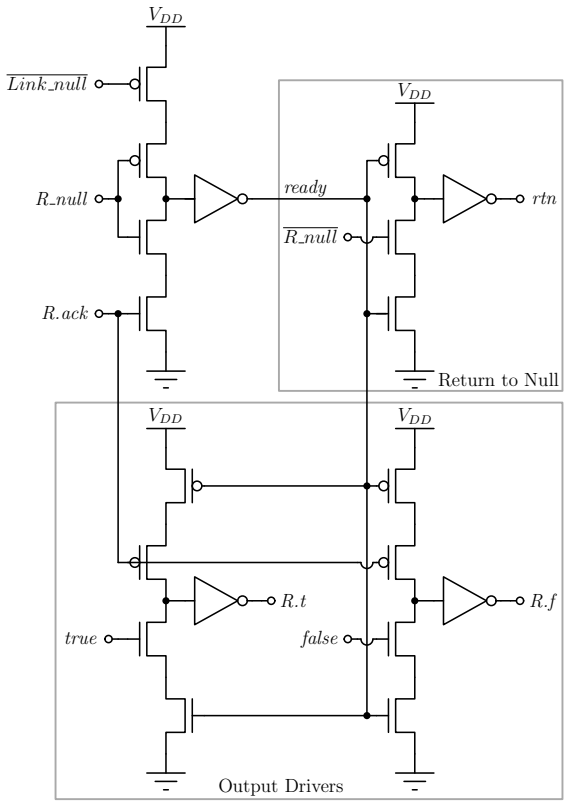


Figure 7.6: RX Stage Sequencing Logic

once the link reaches null, which accomplishes the goal of not driving both sides of the link at once. Again, the timing assumption here is that the local side of the link can be inspected before the remote side inspects the state of the link and causes a transition. Since the link is essentially an RC transmission line, this is a fairly generous timing margin. However, in the case of the passgate return-to-null scheme described in section 7.4.2, the terminating the drive to null is controlled solely by the rtn signal generated as shown in Figure 7.6. This is a slower form of feedback than the others described in subsection 7.4.2, but it serves the same purpose. Once $\overline{\text{Link}_{\text{null}}}$ falls, ready transitions to low and sets rtn false.

Similarly to the TX stage, the RX stage statically holds the link in a valid state until it is time to return the link to null—i.e. once it receives a valid data token, it holds the link at V_{DD} or GND. This is accomplished a pull-up and pull-down

network connected to the link, where the pull-up is the AND of the $\overline{\text{true}}$ signal from the level-shifters and rtn, and the pull-down is the AND of *false*and $\overline{\text{rtn}}$. Thus, while rtn is not asserted and there is valid data on the line, i.e. *false*or *true*are high, the RX stage will hold the state of the line.

7.4 STATS Circuits

7.4.1 Ternary Decode

Our ternary decode is done with a dual level-shifter topology. There are a number of other techniques involving body-biased transistors [47,56,61] or diode-connected transistors [56]. Body-biasing requires the use of triple-well structures and the associated area overheads due to the structures themselves as well as the costs of having additional voltage supplies for biasing. In our simulations, we found diode-connected transistor topologies to burn static power and resolve slowly, confirming the results of Nair, et al. [56].

Our proposed level-shifter topology, shown in Figure 7.7 is a variant of the traditional cross-coupled design used by Felicijan and Furber [23]. Traditional cross-coupled level-shifter designs do not function properly for reasonable sizings in deep submicron technologies, due to the fact that threshold voltages are a much higher relative to V_{DD}. As a result, significant W/L pull-down:pull-up sizing ratios must be used to achieve baseline functionality, which resolves quite slowly and burns a significant amount of power during switching.

Our topology adds transistors in series with the cross-coupled transistors to cre-

ate what amounts to a resettable comparator structure, allowing for fast resolution and much more reasonable per-transistor W/L ratios—on the order of 2:1.

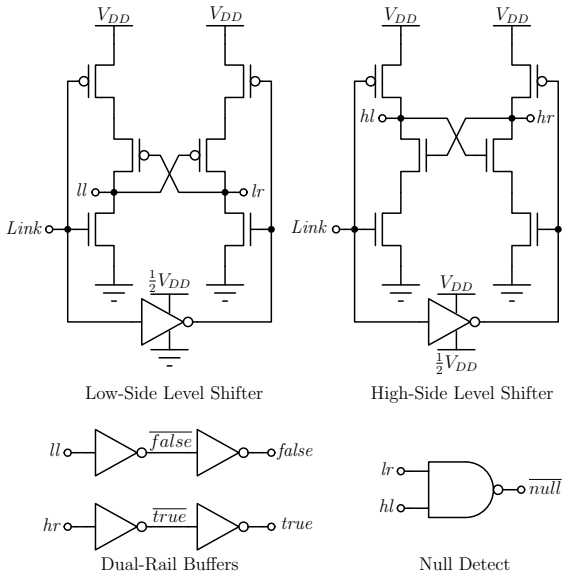


Figure 7.7: Level-Shifter Implementation

The Low-Side Level Shifter converts low-side voltages—GND to $\frac{1}{2}V_{DD}$—back to full swing transitions, and the High-Side Level Shifter does the same for high-side voltages—$\frac{1}{2}V_{DD}$ to V_{DD}. The full truth table for the level shifters can be seen in Table 7.1.

Table 7.1: Level Shifter Truth Table

Link	A	B	C	D
V_{DD}	0	1	0	1
$\frac{1}{2}V_{DD}$	0	1	1	0
GND	1	0	1	0

Depending on the use context, it may be desirable to obtain the signals *true*, *false*, and *null*. A simple buffering scheme, as shown in Figure 7.7, is used to generate the *true*and *false*signals and keep the load capacitance on the level-shifter structures as low as possible. Furthermore, in this type of symmetric cross-coupled structure, it is important to keep the capacitance on the cross-coupled node pairs

{A,B} and {C,D} as similar as possible in order to equalize the transitions on each side of the level-shifters. Thus, rather than generate a *null* signal from the buffered *true*and *false*signals, we can use a NAND gate to compute the *null* signal from the other outputs of the level-shifters. By sizing the input to the first stage of the dual-rail buffers and the null detect NAND2 to match, and placing these structures as close to the level-shifters as possible to minimize wire capacitance, we can keep the capacitance of those node pairs relative matched.

Furthermore, the transistors in the level-shifters should be designed such that the cross-coupled stacks are weaker than their counterparts, e.g. the pull-up stack should be weaker than the pull-down for the Low-Side Level Shifter. This can be accomplished by sizing, setting threshold voltages via device selection, or some combination of both. These considerations facilitate quicker ternary-to-binary translation. Additionally, care should be taken to select sufficient transistor widths to ensure good transistor matching and robustness to process variation.

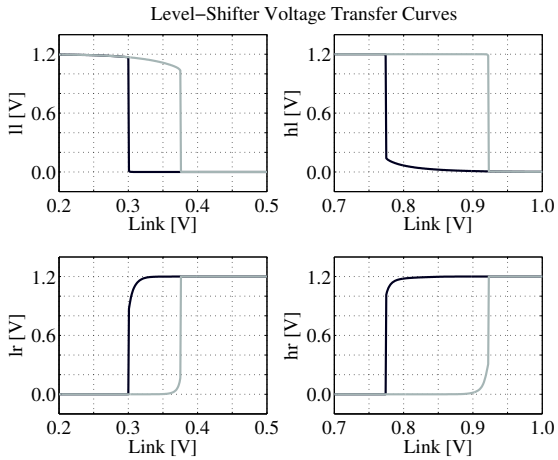


Figure 7.8: Level Shifter Hysteresis

Regarding robustness to noise, ternary encodings by definition have worse noise margin than full-swing encodings. However, the cross-coupled level-shifter design does have hysteresis, which improves noise margin to our advantage. Figure 7.8

is a plot of the Voltage Transfer Curve (VTC) for our level-shifter design in a 90nm process with V_{DD} =1.2V. The dark-colored trace represents a down-going transition—i.e. in the direction of V_{DD} to GND—on the link and the light-colored trace represents an up-going transition on the link.

Thus, in the worst case, static noise with a magnitude of 0.3V is enough to change the state of the link and either destroy or create tokens. However, static noise margin is perhaps overly pessimistic [15]. For short-duration voltage noise these margins will improve.

7.4.2 Return to Null

Ternary signaling systems require a way to return the ternary link to the $\frac{1}{2}V_{DD}$ state. There are a number of techniques which accomplish this goal, all of which assume the existence of a return-to-null (rtn) signal and its complement. rtn is a binary signal which controls the drive of the link to $\frac{1}{2}V_{DD}$. In general, the tradeoffs of the different techniques are between drive slew rate, overall power consumption, area, and the accuracy of the drive to $\frac{1}{2}V_{DD}$. The three different techniques we considered are presented in Figure 7.9.

Passgate

Using a passgate to the $\frac{1}{2}V_{DD}$ supply, as shown in Figure 7.9, is the most accurate, most robust way to drive the link to $\frac{1}{2}V_{DD}$. It is also the simplest, as it requires no additional circuitry—the rtn signal and its complement directly drive the passgate. However, for reasonable transistor sizes, the passgate technology is the slowest of all the techniques detailed here.

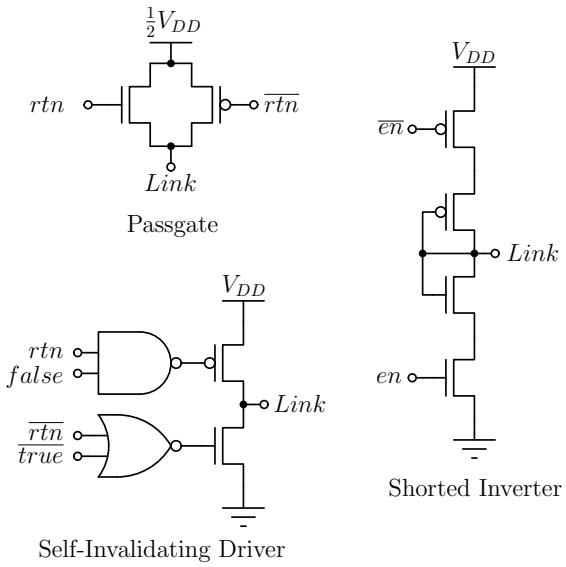


Figure 7.9: Return-to-Null Techniques

Shorted Inverter

The shorted inverter structure in Figure 7.9 is essentially a tri-state inverter with its input shorted to its output. The shorted inverter will drive its output back to the midpoint of its Voltage Transfer Curve (VTC) quickly. While the VTC is sizing-dependent, the midpoint is relatively stable over a wide range of sizings and pMOS/nMOS ratios.

The shorted inverter return-to-null technique can be controlled by cutting off access to V_{DD} and *GND* with *en* and $\overline{en}$, as shown in Figure 7.9. However, by design this technique causes a crowbar current in the shorted inverter while in operation, which is a major source of active power dissipation.

It is therefore desirable reduce the amount of time the inverter is allowed to be on, so as to reduce the power dissipation due to the crowbar current. The signal *en* is the logical AND of the *rtn* signal and the $\overline{null}$ signal from the level-shifter structures described in Figure 7.7. Thus, *en* is high only until the level-shifter

detects null on the link, at which point the shorted inverter is tri-stated.

This technique is relatively area efficient, with a total of four transistors and an AND gate—not accounting for the level-shifters. Also, the slew rate to $\frac{1}{2}V_{DD}$ is better than that of the passgate technique. However, the power dissipation is greater, and it is dependent on the resolve time of the level-shifter structures. While this does not affect the accuracy of the drive to $\frac{1}{2}V_{DD}$, a long delay between detecting null with the level-shifter and the down transition on en can result in unnecessary power dissipation.

Self-Invalidating Drivers

This technique involves interrupting a full-swing transition midway, i.e. at $\frac{1}{2}V_{DD}$. The key assumption here is that the $\frac{1}{2}V_{DD}$ detect circuitry resolves quickly enough to interrupt the transition before it overshoots $\frac{1}{2}V_{DD}$. As with the shorted inverter technique, this technique requires feedback via the level-shifters structure, in particular the $false$ and $\overline{true}$ signals, as shown in Figure 7.9.

The pull-up path, i.e. return-to-null from sending false, is gated by the NAND of rtn and $false$, so it is active only when rtn and $false$ are high. Once the level-shifter resolves the state null, $false$ transitions to low, and the NAND turns off the pull-up transistor. For the pull-down path, i.e. return-to-null from sending true, the situation is similar. The pull-down transistor is active while $\overline{rtn}$ and $\overline{true}$ are low. Then, once the level-shifter resolves null, $\overline{true}$ transitions to high, and the pull-down transistor is shut off by the NOR.

The accuracy of this technique is affected by three factors: the resolve time of the level-shifter structures, i.e. the latency of detecting null, the sizing of the

driving transistors shown in Figure 7.9, and the RC characteristics of the link itself. There are three possible outcomes: the link returns to a value near or equal to $\frac{1}{2}V_{DD}$, the link drives only part of the way to $\frac{1}{2}V_{DD}$ (*undershooting*), or the link drives past $\frac{1}{2}V_{DD}$ (*overshooting*). Depending on the value of the next token, it may be beneficial to overshoot or undershoot. For example, when sending a 0 and then a 1, if the link goes to *GND* to send a 0 then overshoots past $\frac{1}{2}V_{DD}$, less charge will have to be pushed onto the link to drive it to V_{DD} to send a 1. However, if the next token is a 0, more charge will have to be pulled off the link to send another 0. Thus, to approach the best worst case behavior, it is desirable to drive the link to $\frac{1}{2}V_{DD}$ as accurately as possible.

As discussed in subsection 7.4.1, there is some margin around $\frac{1}{2}V_{DD}$ that can be successfully resolved, so some overshoot can be tolerated and may in fact be desirable for performance—in spite of the additional charge to shuffle on or off the link, increased drive strength may increase the slew rate to the extent that there is a net performance gain. Overshoot does reduces the available noise margin, however, and in the worst case, noise can upset the state of the link and cause deadlock or spurious token generation.

For short links where driving the link to null is faster than the resolve time of the level-shifters, this technique will invariably overshoot the $\frac{1}{2}V_{DD}$ mark. Sizing the driving transistors can help increase the drive time to allow the level-shifters time to resolve, but at the cost of maximum frequency. For this reason, we can combine this technique with either the *Passgate* or *Shorted Inverter* techniques to improve the accuracy of the return-to-null drive. In other words, the other technique will correct for overshoot or undershoot. This hybrid design is similar to the ATLS design of Felicijan and Furber [23], but improves upon it by adding

a full passgate and a pull-up self-invalidating driver as opposed to a single nMOS to $\frac{1}{2}V_{DD}$ and only a pull-down self-invalidating driver.

For medium-length links where the drive time is equal to or greater than the resolve time of the level-shifters, the self-invalidating driver technique alone is sufficient to drive the link accurately to $\frac{1}{2}V_{DD}$, assuming appropriately sized drivers. In this case, the self-invalidating driver technique can be used alone.

For long links, the link behavior is more that of an RC transmission line than a lumped capacitance, as discussed in subsection 7.2.1. In such a case, the local capacitance on the link can be driven to $\frac{1}{2}V_{DD}$ quickly with little to no voltage change on the capacitance on the opposite end of the link. Since the feedback for self-invalidation is local, this may cause the drive to shut off prematurely. Following drive cutoff, charge relaxation occurs across the link and the resulting voltage may not be close enough to $\frac{1}{2}V_{DD}$ to resolve to null, resulting in deadlock. Sizing the drive transistors to have near-minimum or minimum width reduces overall system frequency, but lowers the drive slew rate, allowing charge to trickle across the link on the timescale of drive to $\frac{1}{2}V_{DD}$. Alternatively, the feedback from the level-shifters can be delayed via a chain of inverters to extend the drive time, i.e. increase the amount of charge transferred on or off the link.

In general, the self-invalidating driver technique has the best slew rate, as within one RC time constant it can drive 63.2% of the full V_{DD} swing, which is already more than the necessary $\frac{1}{2}V_{DD}$ swing needed. In comparison, the full swing of the passgate technique is $\frac{1}{2}V_{DD}$, so 2-3 RC time constants are required to drive the link back to $\frac{1}{2}V_{DD}$, and the shorted inverter technique essentially comprises a pair of fighting diode-connected transistors, which will lower the slew rate near $\frac{1}{2}V_{DD}$. The transistor count for the self-invalidating driver technique is highest of

all three techniques, but since the largest transistors are the driving transistors, for similar sizing the area overhead for is comparable to that of the shorted-inverter technique.

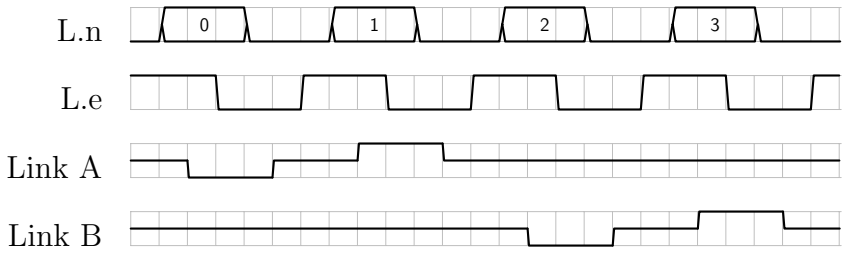
7.5 Two-Bit STATS

7.5.1 Design

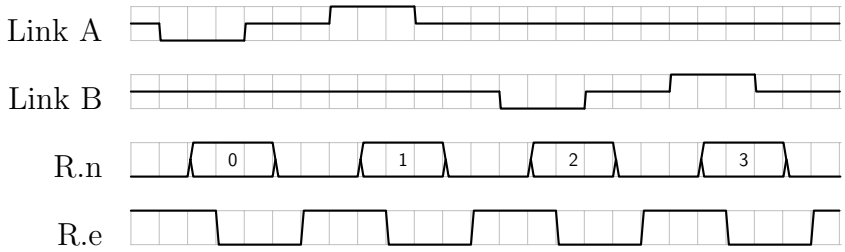
The simplest version of a 2-bit STATS link is to instantiate two links side-by-side. However, in that case we are needlessly transitioning one of the wires. We can instantiate two side-by-side single-track ternary links and transition link A to represent 0 and 1 tokens, and link B to represent 2 and 3 tokens. In this way we are saving a wire transition per token, which should be a 2× savings over two parallel 1-bit STATS links. Figure 7.10 shows a sample timing trace. Since we are transitioning only one out of the two wires, I call this link STATS 1of2.

As for the circuit implementation, STATS 1of2 re-uses most of the circuit structures from the original STATS design. All ternary decode or null-detect circuits use the same level-shifter design as seen in Figure 7.7. Return to Null is accomplished with the passgate scheme shown in Figure 7.9, as testing showed it to be the most robust. The TX stage is largely the same as that shown in Figure 7.5, particularly the part that actually interfaces with the ternary link itself. The sequencing structures are slightly different to account for the new data encoding.

The TX unit accepts an e1of4 channel and produces valid data tokens on two ternary data lines. To accomplish this, we use the same structure as in Figure 7.5,



(a) TX Stage



(b) RX Stage

Figure 7.10: STATS 1of2 Timing Diagram

but use L.n[0]/L.n[2] and L.n[1]/L.n[3] for the left-side data rails, and calculate validity for the appropriate set of data rails and ternary link. The resulting acknowledge/enable signals are ANDed together to produce a single L.e for the e1of4.

The RX unit accepts two ternary data lines and produces a valid 2-bit token on an e1of4 channel. It accomplishes this via a structure very similar to an e1of4 WCHB, which can be seen in Figure 8.1. The relevant sections of the RX stage can be seen in Figure 7.11.

The signals marked LinkA.t, LinkA.f, LinkB.t, and LinkB.f in Figure 7.11 are buffered signals from the ternary decoders hanging off the appropriate ternary link. rtnA and rtnB control the Return to Null passgates driving the ternary links back to $\frac{1}{2}V_{DD}$. Note that this is generally extensible to any 1ofN link where N is an even number, but is probably not desirable as discussed in section 8.2.

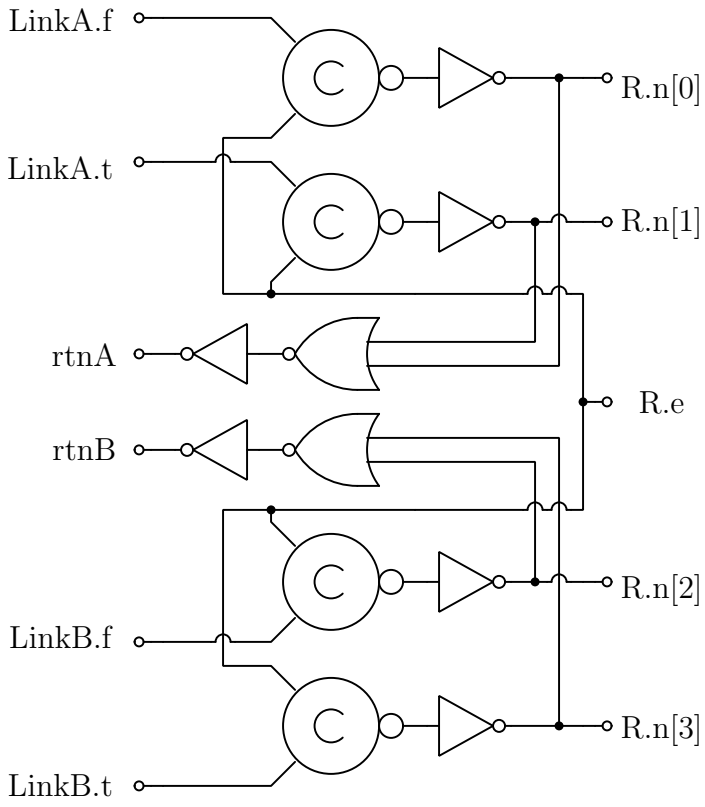


Figure 7.11: STATS 2-Bit RX Stage

7.6 Evaluation

So I have wasted a good deal of space discussing how this methodology can help build designer intuition without actually providing concrete examples. So, since STATS is a link of our own design, I will use it as a simple example to show one of the benefits of my methodology.

We can simply examine the degree of correlation between the various parameters and our metrics of interest. Since some of the parameters are discrete- as opposed to continuous-valued, we make use of the Spearman R Correlation, which can handle such data. Table 7.2 shows the correlation coefficients for 90 nm STATS, both planar and TSV types. The dataset used to calculate these coefficients was

Table 7.2: 90 nm Spearman R Correlation

	Planar			TSV		
Parameter	Freq	Energy	Area	Freq	Energy	Area
ACK	0.06	0.12	0.15	0.13	0.49	0.44
FWD	-0.03	0.09	0.12	-0.18	0.32	0.28
LGC	0.36	0.25	0.29	0.64	0.73	0.77
N	0.79	0.96	0.95	—	—	—
PGP	0.27	0.11	0.14	0.39	0.18	0.23
PGN	0.37	0.14	0.15	0.56	0.11	0.22
PUN	0.38	0.25	0.23	0.40	0.24	0.32
PDN	0.29	0.16	0.15	0.45	0.18	0.24

not just the Pareto optimal points but rather the set of all successful runs.

ACK and FWD are the drive strengths for the e1of2 acknowledge and forward drive strengths. LGC is the drive strength for all logic circuits, i.e. circuits which do not cross the logical boundary of a TX or RX unit. N is the number of subdivisions of the planar wire, ranging from 1 (no subdivisions) to 10. PGP and PGN are the sizings of the passgate RTN transistors, and PUN/PDN are the pullup/pulldown sizings for the TX side link drivers.

In the planar case, we can see immediately that N is highly correlated to frequency, energy, and area. This certainly makes sense, as more subdivisions means more TX and RX stages, leading to a multiplicative effect on energy and area. Shorter wires means less RC on the wires, so frequency should increase as well. ACK and FWD play less of a role here, because the circuits they control the sizing of are not exposed to the capacitance of the link itself and thus are not terribly important. LGC is important, as it controls the relative latency of the computation and thus the overall frequency. The parameters that control the sizing of link drivers, PGP, PGN, PUN, and PDN are also correlated, but less so than N. This is because as you increase N, you can decrease PGP, PGN, PUN, and PDN as the link RC values are less. So while they are important, they appear less correlated.

If we examine PGP, PGN, PUN, and PDN, we see that the correlation values are higher for the TSV configurations than the planar ones. Since we cannot subdivide the TSV, the sizing of the structures touching the TSV capacitance has more of an effect.

7.7 Criticisms

Seems like a lot of effort for potentially little gain.

> STATS is a rather complicated link. It violates the KISS rule and introduces serious considerations for the designer regarding noise immunity, voltage domains, and design time. It is a one-trick pony. All it does is reduce wire count, and it costs a lot to accomplish that.

7.8 Executive Summary

One of the major overheads of asynchronous systems is the high cost of delay insensitive data encodings. The simplest example is the e1of2 link, wherein there is a true rail, a false rail, and an acknowledge signal for the receiver to confirm receipt of a token. This represents a 3× overhead versus a traditional level-encoded link, where there is a one-to-one correspondence between number of bits and number of rails or wires.

Single-Track Asynchronous Ternary Signaling (STATS), is a link of my own design that collapses a e1of2 link into a single wire. It does so by introducing a third voltage level, $\frac{1}{2}V_{DD}$, which encodes a null or spacer state. Thus, the neutral state of the wire is $\frac{1}{2}V_{DD}$. To send a logical 1, the sender raises the wire to V_{DD},

and the receiver acknowledges receipt by returning the wire to $\frac{1}{2}V_{DD}$. Similarly, a logical 0 is represented by a transition to *GND* by the sender and the return to $\frac{1}{2}V_{DD}$ by the receiving process.

The terminology single-track explicitly refers to the fact that both the sender and receiver are driving the same wire. This introduces the dangerous possibility of a simultaneous drive leading to incorrect behavior and energy waste. Thus, we must make a timing assumption as designers that this does not happen, and engineer towards guaranteeing that.

The other major consideration is the fact that this is now a ternary protocol. In addition to having to develop structures to encode and decode a ternary signal, we also lose a significant amount of noise margin due to the inclusion of another valid logic level as part of our protocol. The overheads of decoding ternary are not trivial, as we will see in later chapters.

All in all, STATS is a high risk, relatively low reward protocol. You reduce pressure on your wiring resources, while imposing different and challenging constraints on the designer. The designer must engineer around reduced noise margin, aggressive timing assumptions, and increased complexity. Nevertheless, in environments where wiring resources are scarce, such as TSV-based interconnect, STATS may be worth examination by designers.

CHAPTER 8

MULTI-BIT LINKS

> It depends. Like adult diapers.
>
> Gordon Ivanoski

8.1 Overview

Thus far, we have examined single-bit links. In practice, however, the usefulness of a single-bit link is not terribly high. Even in the FPGA space, where single-bit links have been traditionally helpful, commercial and academic FPGA designs have shifted to multi-bit interconnect [55]. As of this writing, 64-bit datapaths are now commonplace in commercial processors, making multi-bit performance a necessity. As any critic of asynchronous circuits will tell you, one of the largest overheads of asynchronous circuits is related to the data encoding scheme. As we discussed in chapter 6, a fully delay-insensitive encoding of a single bit requires three wires: true, false, and enable. A factor of three increase in wiring resource demand is already unappealing for one bit, and is potentially crippling for something as wide as a 64-bit datapath.

Thus, any serious application of asynchronous circuits to multi-bit datapaths, especially with regards to links, must in some way address this overhead. Fortunately, although the physical wiring or TSV count is higher with asynchronous links in comparison to level-encoded synchronous, we can reduce the switching activity on those wires to at least save on energy. To some extent, we can trade wiring resources to reduce switching energy, which is attractive. This chapter in-

vestigates the various tradeoffs between different multi-bit asynchronous protocols. It is by no means an exhaustive collection of protocols, but should at least provide a sufficient sketch of the potential design space.

8.2 Multi-Bit Theory

The simplest way of creating a multi-bit link is to just instantiate M single-bit links, where M is the desired bit-width. This assumes you have implemented the single-bit link already, but that is beside the point. If we are willing to entertain some additional complexity, we can examine dedicated multi-bit protocols.

If we examine the 1of2 protocol closely, we see that there is a data rail representing 0 and a data rail representing 1. It seems trivial to extend that protocol such that we have more data rails representing additional numbers, such as a data rail representing 2 or 3. Thus, by simply adding two more data rails we can represent a two bit number. For concreteness and clarity, we now have four total data rails, one each to represent the numbers 0, 1, 2, and 3. As with 1of2, we do not allow multiple rails to be asserted at once. This collection of rails is considered a *code word*.

We can generalize this data representation as the *Mx1ofN* protocol, where N is the number of data rails and M is the number of sets of data rails. As a concrete example, a 2x1of2 protocol and a 1x1of4 protocol can express two bit numbers. Do note that I will shorten 1x1ofN to just 1ofN for brevity. An Mx1ofN protocol requires $W = MN$ physical wires to implement, but only M wires will actually transition to represent a data token. Thus, the choice of data encoding is itself a tradeoff space. Fortunately, some back of the envelope calculations will allow us

to drastically reduce the search space.

$$B = M \lg N \tag{8.1}$$

$$N = 2^{B/M} \tag{8.2}$$

$$W = M2^{B/M} \tag{8.3}$$

An Mx1ofN encoding can represent B bits. Since B is likely fixed due to design and architectural constraints, we are primarily interested in the choice of M and N. The edge cases of $M = 1$ and $N = 1$ are degenerate. $M = 1$ results in a case where every number to be represented has its own wire. Even for a relatively conservative situation where $B = 8$, there will be $W = 256$ wires. If $N = 1$, we cannot represent both true and false values and thus do not have a viable signaling protocol for general purpose data transmission and representation. This is not to say that there are no viable use cases for a 1of1 channel, but Mx1of1 channels do not allow for binary data representation. The $N = B$ case is identical to the $M = 1$ case, and the $M = B$ case enforces $N = 2$.

We also must consider the number of switching wires, which is just $W_S = M$, as each 1ofN code word will only transition a single of its N wires. Bottom line, we are trying to minimize both W_S and W to limit our usage of energy resources via switching and physical wiring resources. W_S grows linearly with M, but W generally shrinks with increased M for a fixed B. For $B = 16$, $M = 1$ results in $W = 65536$ but $M = 16$ is gives $B = 32$. In general, you still want powers of two for values of N to make the numbers integer-valued, especially since B is going to be a power of two as well. One particularly interesting observation is that the choice of $N = 2$ and $N = 4$ results in the same number of wires.

$$M = \frac{B}{\lg N} \tag{8.4}$$

$$W = \frac{NB}{\lg N} \tag{8.5}$$

However, $N = 4$ is a factor of two decrease in M over $N = 2$, which means half as many wires are switching. This seems quite attractive: have the smallest number of power-of-two wires and half the switching of $N = 2$. Of course, $M = 1$ results in the least number of switching wires, but a truly ridiculous amount of wiring cost. Like all things, implementing an $N = 4$ link has overheads. In this particular case, a 1of4 link buffer implementation is slightly more complicated than that of a 1of2. Arithmetic logic is also more of a hassle, albeit not a show stopper. For the reasons outlined above, we'll be evaluating Mx1of4 and Mx1of2 multi-bit links in this chapter.

8.3 Multi-Bit Links

In section 8.2, we established two primary options for the data encoding: 1of2 and 1of4. What we have not discussed is the handshake component of each link. As with the links described in section 6.2, we have a variety of choices. Looking specifically at Mx1of2 and Mx1of4 links and assuming $M > 1$ in both cases, we still have a decision to make regarding the granularity of the handshake. Do we implement handshaking at the code word level, i.e. handshaking for each 1of2 and 1of4? Alternatively, do we share the handshaking across all M 1ofN code words? Per-code-word handshaking is identical to instantiating multiple 1ofN links, whereas shared handshaking allows us to amortize the handshake cost over multiple

code words. However, as always there is a cost. Sharing the handshaking between code words is not free and often requires additional circuitry. The key tradeoff question here is whether or not the additional transistors will cost more energy than switching per-code-word handshaking wires.

Other self-timed techniques, such as bundled data [66] and GaSP [67], leverage traditional clock-based datapath elements like flip-flops and latches for pipelining. They generate "clock signals" for each pipeline stage locally, and amortize the cost of this control circuitry over the many bits of a wide datapath. For the purposes of this thesis, we examine only 1ofN protocols and our STATS protocol, described in chapter 7. The remainder of this section serves as an overview of the various multi-bit links we tested.

8.3.1 Single-Bit Links

We will test most of the single-bit links from section 6.2 by instantiating B copies of a single-bit link to form a multi-bit link B bits wide. We will be working with WCHB e1of2 (subsection 6.2.1), RQDI e1of2 (subsection 6.2.2), STFB 1of2 (subsection 6.2.4), and STATS (subsection 6.2.5). We are omitting an analysis of ATLS (subsection 6.2.3) from the multi-bit discussion, our single-bit analysis has already shown that it is not competitive when compared against the aforementioned links.

8.3.2 WCHB e1of4

Like its e1of2 counterpart, the WCHB e1of2 (subsection 6.2.1), the 1of4 Weak Conditioned Half Buffer is very conservative. The implementation is shown in

Figure 8.1. Comparing the WCHB e1of4 implementation to the WCHB e1of2 in Figure 6.2 reveals not all that much of a difference. We have simply added two more data rails. This has the effect of doubling the fan-in on R.e and converting the L.e NAND2 into a NAND4. All in all the increase in complexity is negligible, although any further increase past a 1of4 code word will make a larger NAND gate than may be feasible, depending on technology. While we could implement a larger NAND with more gate stages, that starts to eat into the overall cycle time.

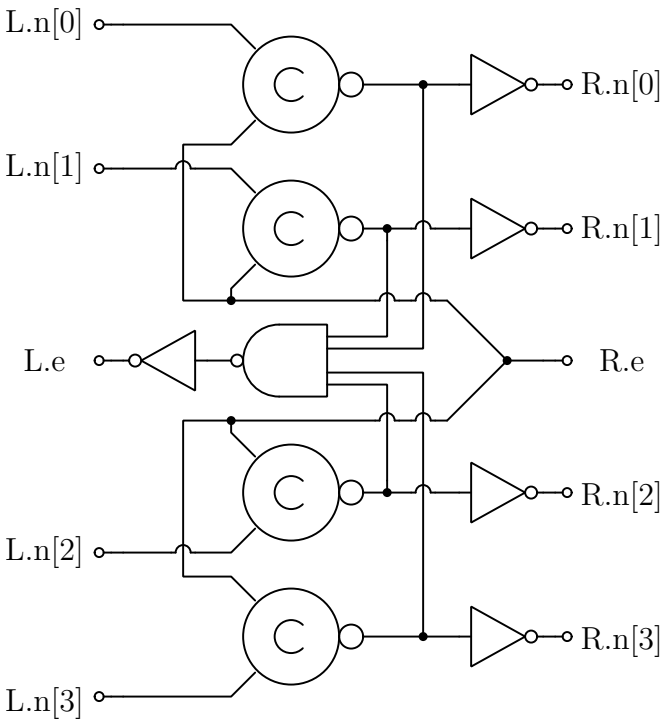


Figure 8.1: WCHB e1of4

8.3.3 STFB 1of4

Figure 8.2 shows the 1of4 implementation of STFB, taken from [24]. It is based on the STFB 1of2 template shown in Figure 6.4. Instead of true and false rails, we now have rails representing the values 0, 1, 2, and 3. The pulldown timing

is now governed by a NAND4 as opposed to a NAND2, but the pullup timing is controlled by NOR2 gates.

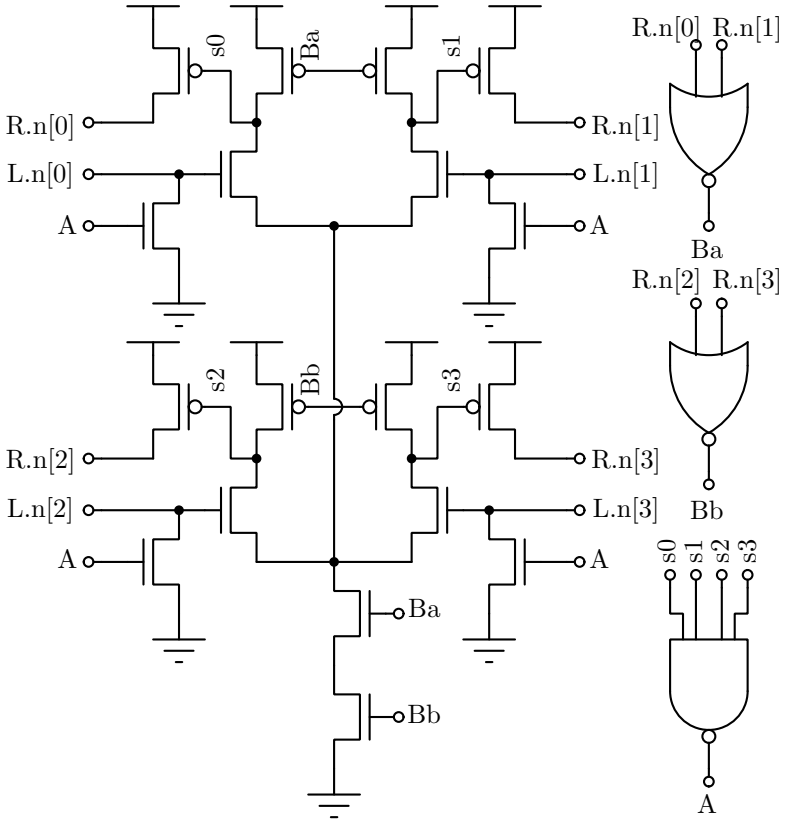


Figure 8.2: STFB Template

8.3.4 STATS 1of2

section 7.5 describes the multi-bit implementation of STATS in detail. However, to save you the trouble of reading it, I will reiterate the main points here. STATS is a single-bit link where V_{DD} is 1, *GND* is 0, and $\frac{1}{2}V_{DD}$ is the absence of data. STATS 1of2 is effectively two STATS links instantiated side by side, but only one is active. If we label each link A and B, link A at V_{DD} represents 1, and A at *GND* represents 0. Link B high and low represent 3 and 2, respectively, thus implementing a 2-bit link. This also has the effect of transitioning only one wire,

but comes at the cost of some additional circuitry to handle the encoding/decoding of the above signaling scheme.

8.3.5 PCHB Mx1ofN

There are multiple implementation styles for the Pre-Charge Half Buffer (PCHB) [20,44], but we are interested primarily in the vanilla PCHB, not the PCeHB. While there is a 1of2 and 1of4 version of the PCHB, the WCHB has proven itself to be the workhorse of link buffers. In contrast, the PCHB is typically used to implement logic. This is in part because the PCHB is a very flexible template design, as seen in Figure 8.3. The figure shows a single PCHB rail, e.g. a true rail in a PCHB 1of2.

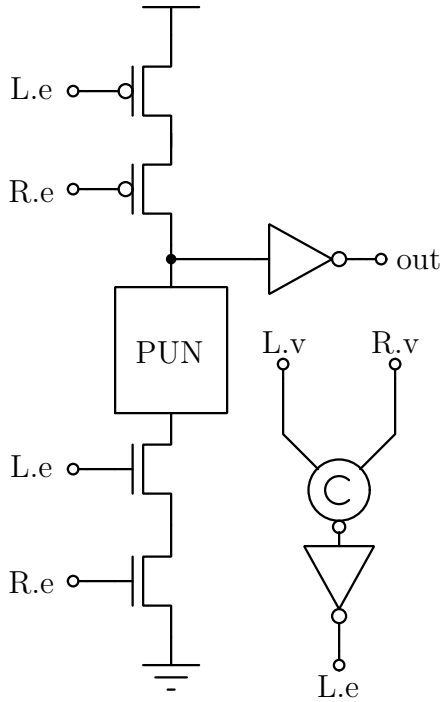


Figure 8.3: PCHB Template

PUN refers to the Pull Down Network, which can be as simple as a single nMOS

transistor when implementing a PCHB buffer. In cases where you are trying to implement logic, the PUN implements the pulldown for the desired logic function, with the precharge pMOS handling the pullup. The interesting signals are L.v and R.v, which represent the validity of the left and right side channels of the PCHB, respectively. Validity is calculated simply by checking each of the M code words for one of the N wires to be high. This does necessitate the inclusion of a C-element tree for links where $M > 1$ to calculate validity. For safety, especially in a high RC environment, we assume the receiver calculates the validity of the sender. This C-element tree presents a large overhead to both latency and energy, as we will see in subsection 8.5.2.

8.4 Methodology

The methodology for the multi-bit study is effectively identical to that outlined in section 6.3. For completeness, I will provide a short overview here. For single-bit links, we used the numbers presented in chapter 6 where appropriate, multiplying the energy and area as appropriate to account for the additional power consumption and hardware for a wider datapath. For dedicated multi-bit links such as the WCHB e1of4, STFB 1of4, STATS 1of2, and PCHB Mx1ofN links, we ran SPICE simulations of each link.

As in Figure 6.8, each link is implemented as a Device Under Test (DUT). A Planar DUT allows for subdivision of the RC link, with additional link hardware instantiated as necessary. The TSV DUT is simply a TSV RC model bookended by a send and a receive stage, as appropriate. Instead of the e1of2 interface depicted in Figure 6.8, we have the appropriate link protocol for the DUT.

Of all the aforementioned dedicated multi-bit links, only the PCHB Mx1ofN link, detailed in subsection 8.3.5 scales past two-bits. Similar to the single-bit links, we will build a wider datapath out of two-bit links. For example, we compare a 4-bit PCHB 4x1of2 link against two WCHB e1of4 links running side by side. Again, as with the single-bit links, we multiply the area and energy as appropriate to reflect the additional hardware.

As with the single-bit study, the environment is fully QDI. For the 2-bit links we use an e1of4 channel, which requires conversion from/to e1of4 to/from STFB 1of4 and STATS 1of2. The cost of that conversion is accounted for in both energy and area. As before, we use WCHB e1of4 buffers to decouple the analog from the digital environment—these are free in terms of energy and area costs. For bit widths larger than two, we use evMx1ofN channels, which are simply buffered with PCHB evMx1ofN units, and do not require conversion.

Our optimization framework makes use of hopTK and DEAP, as detailed in chapter 5 and Figure 6.11. We measure frequency by tapping an enable signal on the receive side of the DUT, area by a $W \times L$ sum of all transistor dimensions, and energy by the power over frequency metric. Power is calculated by measuring current draw at the appropriate DC voltage source. For those links requiring multiple voltage sources, such as STATS, we account for all voltage sources when calculating power and energy. For the most part, the tunable design parameters for the dedicated multi-bit links are limited to transistor sizing and degree of RC link subdivision—number of buffers to insert into a planar link.

8.5 Evaluation and Discussion

There are infinite combinations of datapath width and code word choice, so we will limit our study to two choices of datapath width and the choice between 1of2 and 1of4 codewords. The first datapath width of 2-bits is for illustration purposes only. We will use it to demonstrate the efficacy of the dedicated two bit links described in section 8.3. The second datapath width is 16-bits, which is a reasonable datapath width for a wide array of applications. One could argue that a 16-bit width is outdated in the 64-bit era, but there are still many low-power and legacy processors that still use 16-bit datapaths.

Table 8.1: Multi-Bit Link Technology Parameters

Wiring/TSV	Wire Length [µm]	Energy (CV^2) [pJ]
90 nm	1000	0.759
65 nm	600	0.204
28 nm	300	0.062
TSV 1.2V	—	0.216
TSV 1.0V	—	0.150

The remainder of this section is organized into two subsections, one for 2-bit and one for 16-bit. Each subsection contains results for planar links and TSV links. As in chapter 6, each planar link is 20000λ long. Table 8.1 lists the technologies we use to evaluate, the planar link length in absolute units, and an estimate of CV^2 switching energy. As in chapter 6, this is for context only. Unlike chapter 6, for all dedicated multibit links we do *not* allow V_{DD} values below nominal. As we will see, this only affects the low-energy portion of the Pareto front, as we would have guessed based on the results presented in section 6.4.

Additional differences between this study and that presented in section 6.4 concern scaling. Since the bit widths are quite large, scaling the throughput by TSV usage would result in very low frequency numbers. Since my proposed methodol-

ogy aims for clarity and designer intuition, I have opted for clarity here as opposed to an artificial metric. The scaled throughput of section 6.4 is sufficiently easy to understand as the scaling factor is no more than 3. Here, however, the scaling factor can approach 82 at the upper end. The original single-bit study had results from a 45 nm SOI server process, which was not an ideal deep submicron node to compare the 90 nm and 65 nm low power processes in chapter 6. As a result, I have dropped the SOI node from the comparison here and replaced it with 28 nm low power process, which is a more fitting comparison with the other two technology nodes. To provide a complete picture, I also present the entirety of the results from each technology as opposed to the condensed composite Pareto fronts of section 6.4.

8.5.1 2-Bit Links

In this subsection we examine the difference between dedicated two-bit links and replicated single-bit links. The key improvement is to energy, as described in section 8.2. We are comparing the following six links:

- 2x STATS — Single-bit STATS, doubled
- STATS 1of2 — 2-bit STATS, 1 of 2 links is active
- 2x STFB 1of2 — Single-bit STFB
- STFB 1of4 — 2-bit STFB, 1of4 codeword
- 2x WCHB e1of2 — Single-bit WCHB, doubled
- WCHB e1of4 — 2-bit WCHB, 1of4 codeword

Planar Links

Figure 8.4 shows the frequency-energy and frequency-area Pareto fronts for all six links in 90 nm. There are essentially no surprises here. Both STFB links outperform all the other contenders, with WCHB outperforming STATS except at the lowest frequency end of the spectrum. Each of the dedicated 2-bit links outperform their single-bit counterparts in energy but not in area. What this suggests is that the additional complexity of handling two bits costs more transistors, as is the case with STFB 1of4, or that we must more aggressively size transistors such as in the case between a NAND2 and NAND4 in the WCHB variants. As the frequency increases, we see the effects of the latter factor, the sizing, dominate.

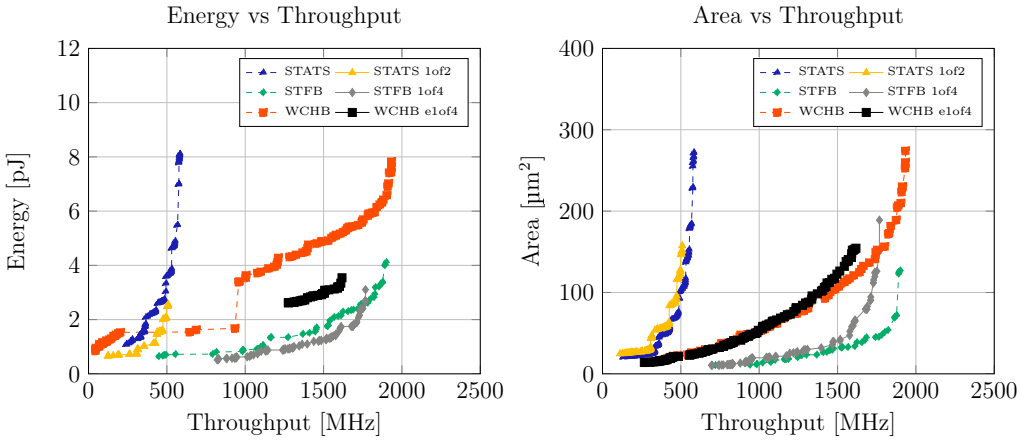


Figure 8.4: 90 nm 1000 µm 2-Bit Link

Lower frequency points are generally comparable between WCHB/STFB single and 2-bit variants, but they diverge as we move past 1 GHz. For the same frequency, STATS 1of2 has increased area over its single-bit counterpart, but without the divergence shown by the full-swing links. This is because most of the area of both STATS links is taken up by ternary decode and link drivers, which exist in both variants of STATS. The slight increase in area is due to the additional complexity of the sequencing/handshaking circuitry.

The effects of reduced V_{DD} can be clearly seen in the frequency-energy plot of WCHBs in Figure 8.4. The WCHB e1of4 Pareto front shows less energy overall, but does not extend into the lower frequency ranges. This is a direct consequence of not allowing V_{DD} sweeping for the dedicated multibit links. The same is not true for area—we can see many low-frequency points in the WCHB e1of4 trace. The takeaway point for this is the following: an under-sized WCHB will have a lower frequency, but this is not a favorable energy tradeoff. To really achieve efficient low-energy operation we must apply both sizing and voltage scaling.

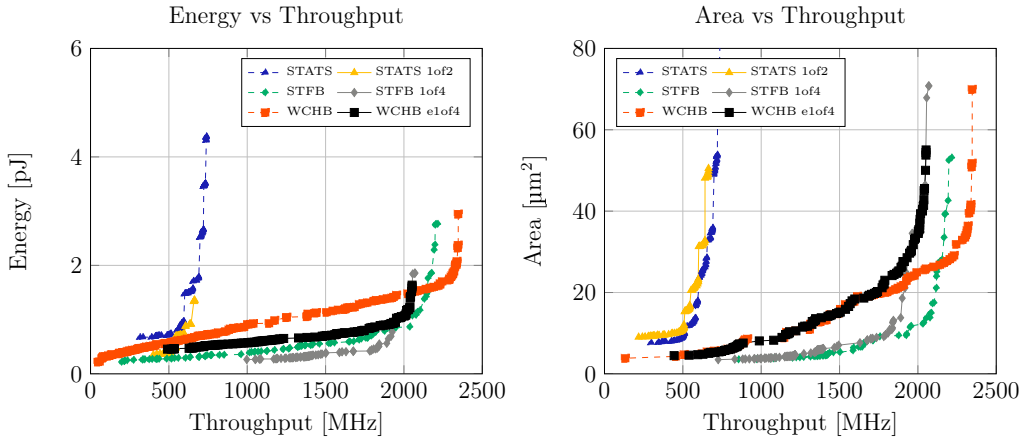


Figure 8.5: 65 nm 600 µm 2-Bit Link

Overall link rankings remain the same going to 65 nm. Expected CV^2 energy decrease from 90 nm is approximately 73%, using the numbers from Table 8.1. This is in line with the lower-frequency energy numbers shown in Figure 8.5. Expected area decrease is roughly $2x$, but we actually do slightly better than that overall for lower-end frequencies. Numbers track as expected with those in Table 8.1, again with the caveat that the single-bit links can scale V_{DD} and the dedicated multi-bit ones cannot.

We do see, however, a more slightly more pronounced divergence in area between the 2-bit and single-bit variants of each link when compared to the 90 nm

plots. Figure 8.5 clearly shows the overheads of the more complex logic involved in multi-bit signaling. There is a prounounced knee in both the frequency-energy and frequency-area curves for STFB 1of4 and WCHB e1of4, illustrating the diminishing returns of increased sizing and/or link subdivision in an attempt to increase frequency. This is a direct result of the more complicated logic required to implement the 1of4 codewords. The single-bit variants also exhibit a pronounced knee, but at a higher frequency.

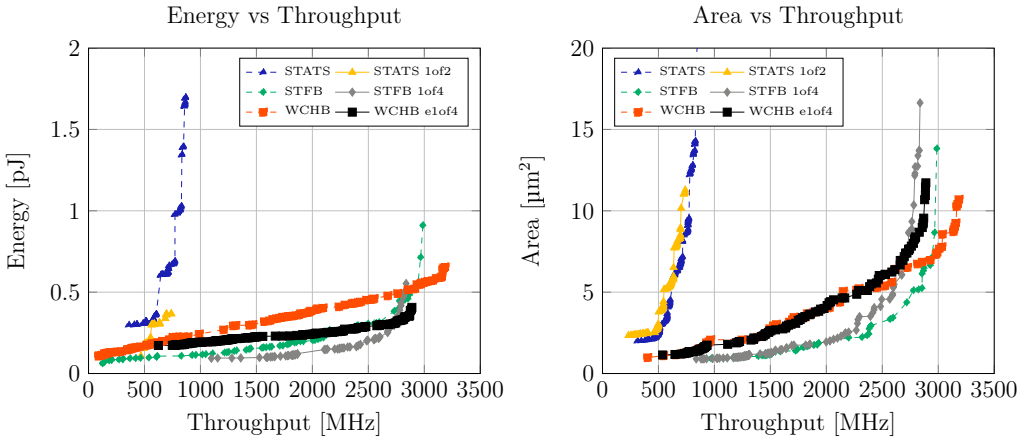


Figure 8.6: 28 nm 300 µm 2-Bit Link

All of the aforementioned trends hold in 28 nm, as can be seen in Figure 8.6. Across technologies, we see the benefits of scaling—increased frequency, reduced area, and reduced energy. It is important to note that we are decreasing the absolute link length as we scale down in process node, however. To achieve the same 1 mm link in 28 nm, we would have to spend over 3× the energy and area to maintain the performance reported in Figure 8.6.

TSV Links

Figure 8.7 shows the frequency-energy and frequency-energy Pareto fronts for all six links, as before. Unsurprisingly, all trends hold from planar across to TSV.

The energy and area numbers are decreased from the planar case, this is due to the inability to instantiate multiple copies of a buffer to subdivide the TSV. We can see that the STFB links approach optimal energy, when compared against the 0.216 pJ number from Table 8.1. The STFB links have always been very lightweight and very fast, and these results reinforce that fact. WCHB is not far behind in performance, however, so for those looking for stronger timing guarantees, that option is available.

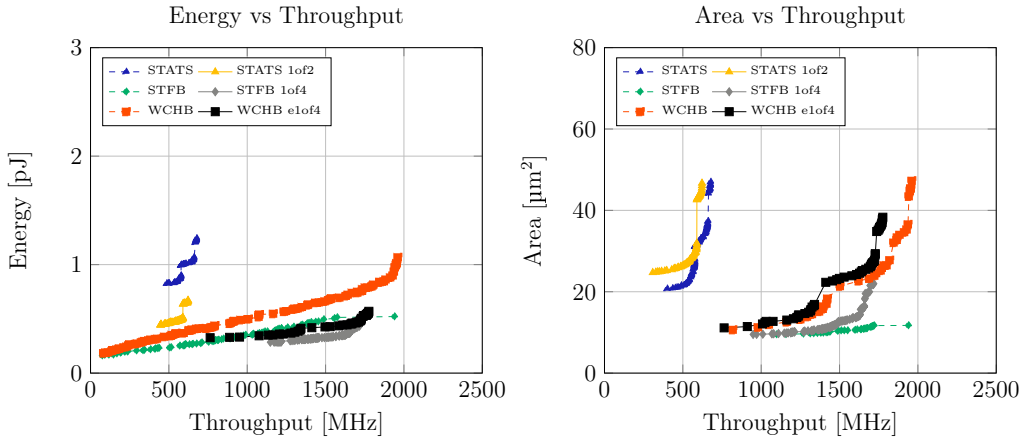


Figure 8.7: 90 nm TSV 2-Bit Link

Both STATS variants are unattractive in this environment, particularly because we are not using the per-TSV throughput metric as we did in section 6.4. We will return to this subject in subsection 8.5.2.

Figure 8.7 is particularly effective at demonstrating the effects of reduced V_{DD}. We can see that the single bit WCHB offers excellent energy numbers at lower frequencies due to the effects of V_{DD} scaling. However, the area numbers tell a different story. It costs roughly the same amount of area to achieve the same frequency for both WCHB variants. However, we can see from the energy curve that we can achieve a frequency of 250 MHz, which is not the case with the area Pareto front. This suggests that while we can achieve reduced frequencies by scaling V_{DD},

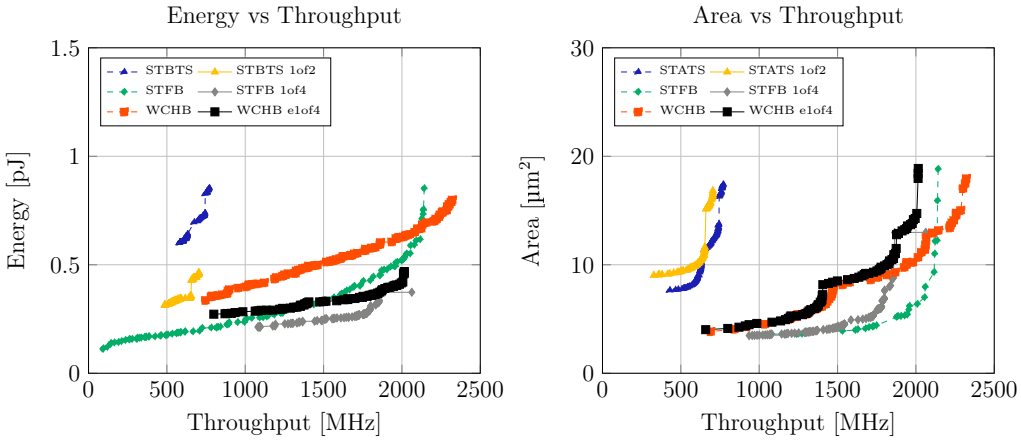


Figure 8.8: 65 nm TSV 2-Bit Link

the area stays more or less the same for those V_{DD}-scaled configurations.

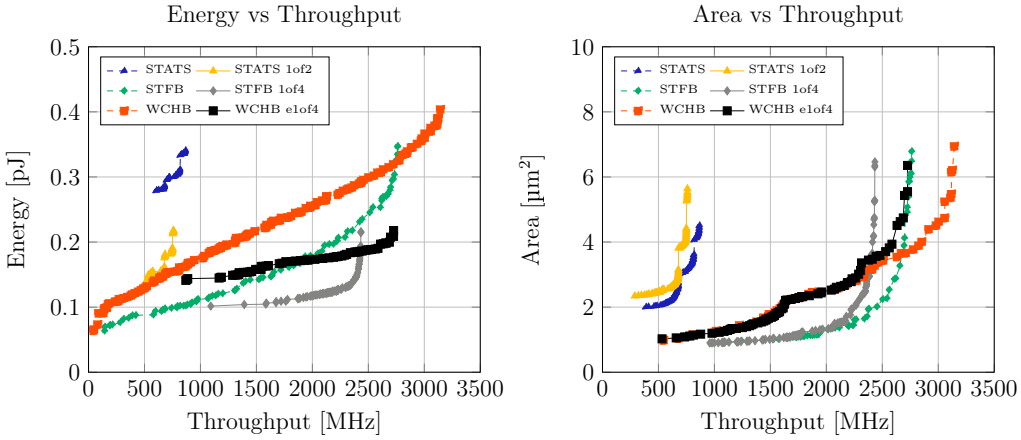


Figure 8.9: 28 nm TSV 2-Bit Link

Figure 8.8 and Figure 8.9 maintain the trends described above. Energy does not scale quite as precipitously from 90 nm as it did in planar, this is due to the substrate remaining largely the same as we assume the TSVs do not scale. Energy numbers also track well against the estimates in Table 8.1 on the lower end of the frequency range.

8.5.2 16-Bit Links

In this subsection we examine five different ways of implementing a 16-bit link. We use only the dedicated two-bit links evaluated in subsection 8.5.1, as they offer the best energy benefits. Comparing against single-bit links replicated 16× is certainly possible, but would clutter the result graphs below.

Our links for comparison are as follows:

- PCHB ev8x1of4 — PCHB with 8 1of4 keywords
- PCHB ev16x1of2 — PCHB with 16 1of2 keywords
- STATS 1of2 — STATS 1of2, replicated 8 times
- STFB 1of4 — STFB 1of4, replicated 8 times
- WCHB e1of4 — WCHB e1of4 replicated 8 times

Unlike in subsection 8.5.1, I show the pass/fail statistics for each link Table 8.2. The 2-bit analysis was intended to justify the selection of dedicated 2-bit links for this 16-bit link study, and not as a realistic depiction of a typical multi-bit link use case.

Table 8.2: 16-bit Link Failure Rates

Link	% Planar Failure			% TSV Failure		
	90 nm	65 nm	28 nm	90 nm	65 nm	28 nm
PCHB ev8x1of4	33.49	33.09	62.54	56.08	37.39	27.37
PCHB ev16x1of2	52.94	60.34	62.08	56.90	36.61	39.80
STATS 1of2	51.41	32.28	40.31	25.63	36.41	23.88
STFB 1of4	64.85	15.12	50.05	15.60	53.43	17.96
WCHB e1of4	8.96	0.88	2.34	0.43	6.03	3.54

Table 8.2 contains a few surprises. What was expected was for WCHB to have very low failure rates and for STATS and STFB to have relatively high failure rates.

Both of these predictions came true, with the exception of STFB performing well in the TSV case, which previously was not the case. The PCHB links are significantly more fragile than the WCHB links, although both are technically QDI.

Planar Links

Examining the STATS, STFB, and WCHB links across all three technologies, we can see that STFB still outperforms the other links, including our two PCHB offerings. STATS is again relatively unattractive for general planar link usage. The PCHB links do not perform very well at all, and this is due to the extreme round-trip latency inherent to the link archetype. In the PCHB evMx1ofN, the sending process generates a token, the validity of which is checked via NOR gates and a C-element tree in the receiving process. This right-side validity signal, which is used as part of the handshaking in the sending process must then traverse link wiring back to the sending process. Thus there are two full wire latencies, one for data, and one for validity in addition to a tree completion in the critical path for the channel.

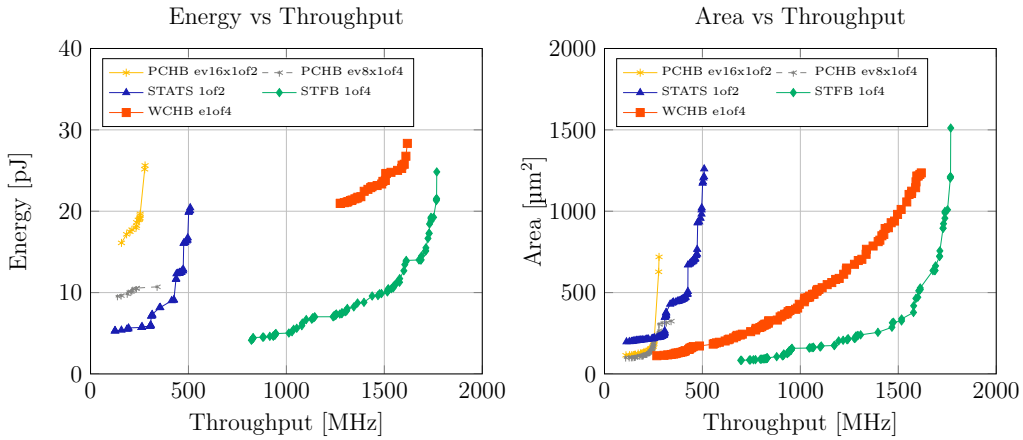


Figure 8.10: 90 nm 1000 µm 16-Bit Link

Figure 8.11 show a point at which STATS actually outperforms WCHB around

500 MHz. However, even in this edge case, it is not worth switching to STATS from WCHB. A few fractions of a picojoule are not worth the penalty in link robustness. It is worth noting that WCHB matches the performance of STFB at the higher throughputs in 65 nm in both energy and area. However, the area cost of these buffers are quite high. In practice, most designs would likely operate in the 1 GHz range, so spending more than a factor of 4× in energy and area for only a 2× improvement in frequency is simply not worth it.

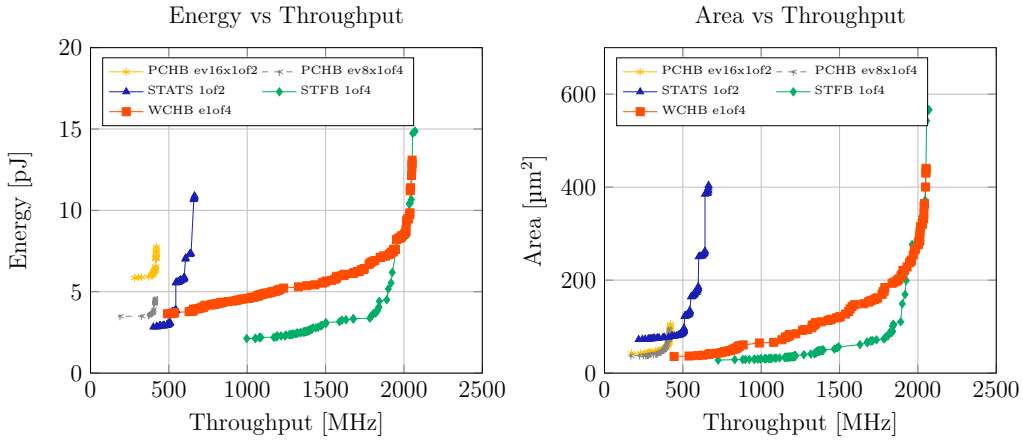


Figure 8.11: 65 nm 1000 µm 16-Bit Link

The comparison between 28 nm between WCHB and STFB is much the same. While the absolute difference between the energy and area of WCHB and STFB is less, the relative difference is comparable. STATS and the two PCHB configurations remain unattractive design options, although STATS barely outperforms the ev8x1of4 version of PCHB in energy. STATS still loses in area due to the amount of additional support circuitry it needs to function.

The story for 16-bit links remains the same as that of 2-bit and single-bit links. STFB still holds the title as the most energy and area efficient link, although it cannot match WCHB for robustness. In practice, to minimize the amount of design, test, and designer headache, I would recommend using any WCHB-based

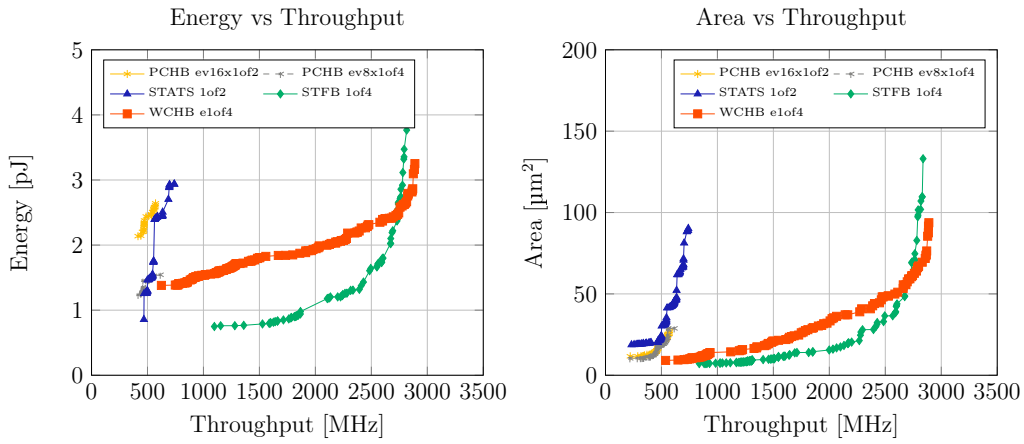


Figure 8.12: 28 nm 300 µm 16-Bit Link

link.

TSV Links

The designer takeaway for the TSV links below is much the same as all of the other data presented before. STFB and WCHB dominate all scenarios, energy and area cost are less than that of planar, but frequencies are similar. STATS and the PCHB configurations remain unattractive.

Table 8.3: 16-bit TSV Costs

Link	TSV Count	Switching TSVs
PCHB ev8x1of4	34	10
PCHB ev16x1of2	34	18
STATS 1of2	16	8
STFB 1of4	32	8
WCHB e1of4	40	16

Table 8.3 shows the TSV counts per link. STATS, as advertised, has the least in both overall count as well as TSVs that are switching per token. STFB has twice as many TSVs but switches the same number of TSVs as STATS. WCHB is one of the most expensive links in terms of TSVs, next to PCHB ev16x1of2. If a

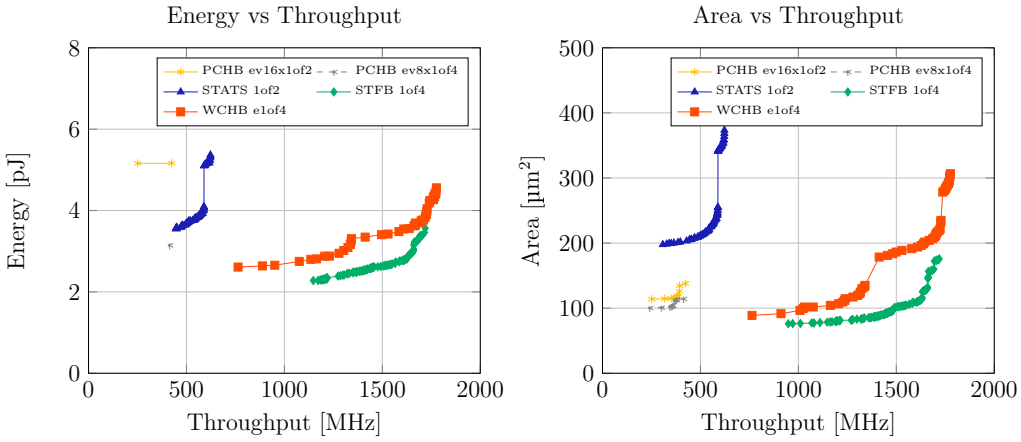


Figure 8.13: 90 nm TSV 16-Bit Link

design is particularly TSV-starved STATS may still be attractive, but studying the numbers in Table 8.2 suggests that a designer will likely be spending a great deal of time trying to get a STATS link working. At that point, that time might be better spent re-organizing the design structure to reduce pressure on TSV resources to allow designers to use STFB or WCHB links.

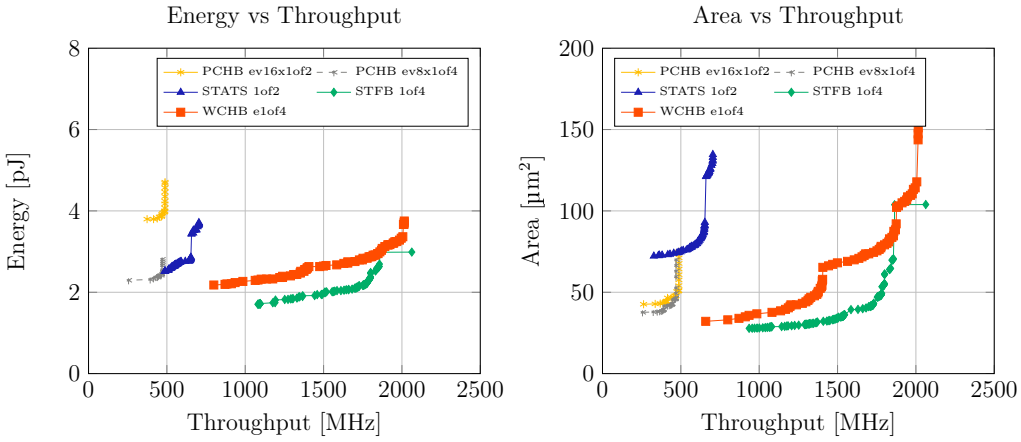


Figure 8.14: 65 nm TSV 16-Bit Link

Area cost is quite important for TSV links as well. As of this writing, TSVs dominate substrate area, often creating keepout areas for transistors. In practice, links exist to carry data around. Presumably that data has been generated by some structure that takes up space on die. Links, logic, and now TSVs must compete

for a limited resource, all under the somewhat artificial pressure of wanting to maximize spatial locality. In other words, having a long planar link drive a TSV link is a counterproductive misuse of resources. While STATS has the fewest number of TSVs it also takes the most area for the least performance benefit, which is yet another reason not to use it.

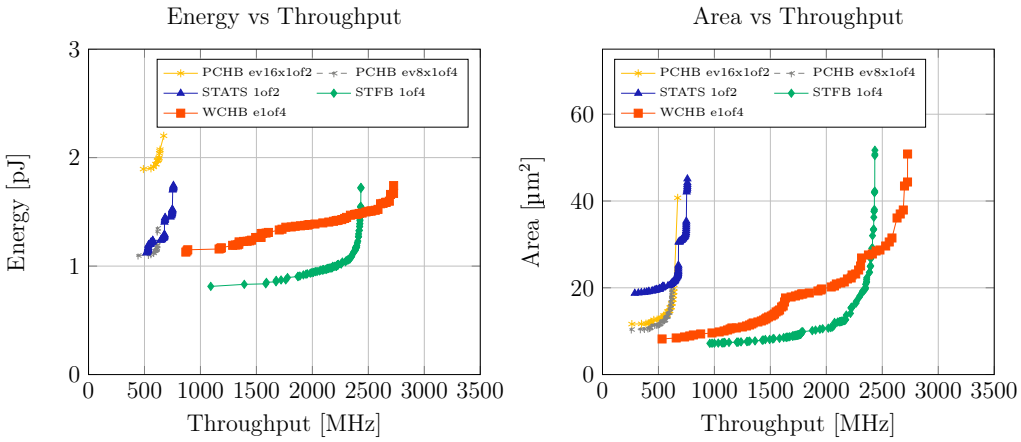


Figure 8.15: 28 nm TSV 16-Bit Link

Of all the technologies, 28 nm offers the most favorable comparison between WCHB and STFB. Unlike the planar case, the area does not grow prohibitively much to achieve high throughput targets, making WCHB a useable alternative to STFB. As before, use STFB if you are willing to put in the design and verification time, and WCHB if you would prefer a most robust link.

8.6 Criticisms

STATS really does not work, then.

Yes, I agree. There might be a situation where you have truly run out of TSVs and you need to recover some. If you are willing to pay in design and verification time, area, energy, and performance to reduce the number of

TSVs, then use STATS. In practice you are better off with WCHB or STFB.

Where is my designer intuition? All I see are graphs.

Hopefully the graphs have provided a reasonable picture of the design space, or at least enough of one for you to make an informed decision on what link to choose. I have also elaborated on some of the reasoning behind the trends in the graphs. Unfortunately, part of the experience of this methodology is in the act of performing the optimization and pouring over the data yourself. The methodology does allow you to examine the configuration parameters for each point in each Pareto front, which should allow you build intuition as you work on your design. Rather that provide a dump of that data, I have distilled the salient points for your consumption above.

8.7 Executive Summary

In practice, wide datapaths are much more common than single-bit datapaths. Single-bit links are not terribly well-suited to wide datapaths, especially links with delay insensitive encodings where the wire or TSV count per bit can be up to three. When dealing with delay insensitive encodings, we can generally classify them as Mx1ofN links, where N represents a bit grouping or codeword, i.e. $N = 2$ is a codeword with a true rail and a false rail. $N = 4$ represents a codeword where we have a wire or rail for the values 0 through 3. M represents the overall number of codewords we use to implement our datapath. Thus, a 8x1of4 link can carry the same number of bits as a 16x1of2 link, i.e. 16-bits.

The number of wires used by each link is simply MN, but it is important to consider the number of wires which transition when sending each token. Only one

of N wires in each codeword will transition, meaning the total number of transitioning wires is M. While we would like to minimize the number of transitioning wires, N grows much faster than we can reduce M. For example, a 1x1of65536 link has the least number of transitioning wires but a truly ridiculous number of wires for a 16-bit link. In practice, 1of2 and 1of4 codewords are optimal, and actually use the name number of overall wires to represent the same amount of bits. However, 1of4 implementations transition half as many wires, resulting in a potential energy savings. This savings comes at the cost of increased complexity of logic and sequencing circuitry, but this may be acceptable for long links or highly capacitive TSVs.

We considered several different multi-bit links. The easiest to describe are STFB 1of4 and WCHB e1of4, which are simply the 1of4 codeword variants of their single-bit siblings. We also implemented a STATS 1of2 variant, which uses two side by side single-bit STATS links, but transitions only one of the links when sending tokens, resulting in additional energy savings. Finally, we considered PCHB ev8x1of4 and ev16x1of2 links, which share enable signals across all 16-bits for additional amortization. We simply instantiated multiple copies of the STATS, STFB, and WCHB links to build 16-bit links.

WCHB and STFB performed very well in comparison to our other link candidates, offering affordable energy and area costs for acceptable performance. The same could not be said for STATS and PCHB. STATS has traditionally suffered in planar use cases, so this is not a surprise. STATS still offers the lowest cost in terms of TSV count, so if that is a concern it may be worth a cursory look before ultimate dismissal as a curiosity. The PCHB evMx1ofN style links are unattractive because of the long critical loop, described in detail above. In short, as with

single-bit links, STFB offers the best performance with WCHB lagging slightly behind. However, WCHB dwarfs all other competitors in terms of sheer number of functional link configurations, implying its superior robustness

CHAPTER 9

LOGIC STUDY

> Advice is what we ask for when we already know the answer but wish we didn't.
>
> Erica Jong

9.1 Overview

In the previous chapters, we have discussed on-chip links. The other major class of circuits is logic—of course there is memory as well. There are many different logic implementation styles and many different types of logic to implement, so I have limited the combinatorics to two styles of logic and two types of logic. The styles I have chosen to evaluate are Null Convention Logic (NCL) [21,22], and Pre-Charge Half Buffer (PCHB) [20,44].

NCL is touted as a low cost of entry asynchronous circuit family. It is designed as the asynchronous variant of the industry standard cell flow. Designs are mapped to a library of approximately 40 cells, which are then placed and routed in the usual way. This has the benefit of being able to leverage industry tools, directly addressing the issue of lack of asynchronous tool support. Of course, nothing is free. By virtue of having to map functions to a limited set of general purpose cells, the resulting design often has more series stages or other inefficiencies than a custom design would have.

PCHB, on the other hand, is a template-based logic family, as discussed in subsection 8.3.5. The sequencing/handshaking logic introduces minimal overhead,

especially when compared against NCL, and all logic is custom-tailored to the application. One common criticism of the PCHB design style is that it has limited tool support, takes a significant amount of full-custom designer effort, and does not adapt well to an industry-standard toolflow. Recent advances in research tools have married the flexibility of full-custom design to the productivity of industry place and route tools [40]. These new tools have allowed designers rapidly prototype silicon using PCHB templates [33].

To evaluate the difference between NCL and PCHB, I have chosen two of the most common logic structures, the full adder and the half adder. Since NCL and PCHB are different design styles, I have provided a brief implementation sketch for all four circuits in section 9.2.

9.2 Test Circuits

9.2.1 NCL Full Adder

The NCL full adder is based on the threshold gate. A threshold gate is typically denoted THMN, which is a N input, 1 output gate. The initial output is low, and when M of the N inputs are high, the output will transition high. The gate will reset once all inputs are low again. A TH1N gate is simply an N-way OR gate, and an THNN gate is a N-input C-element.

Figure 9.1 shows the implementation of an NCL adder. A, B, and C are the A/B operands and the carry in. S and D represent the sum and carry out, respectively. Note how the carry out signals are wire copied to the TH35 gate twice. There is

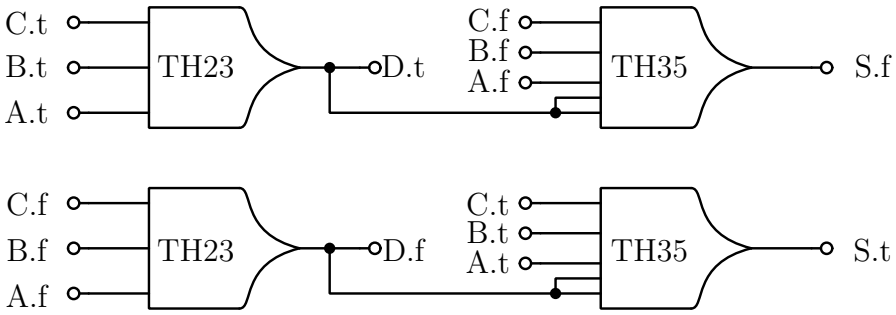


Figure 9.1: NCL Full Adder

another type of gate that weights a particular input more than one, i.e. it will count a single input more than once for the purposes of transitioning high. We can implement this particular TH35 gate as a TH34W2111 gate, which means the first input of four is doubly weighted for the purposes of transitioning high. I do allow hopTK to choose between the TH35 and the TH34W2111 implementations.

The gate as shown in Figure 9.1 does not include an implementation of handshaking, so there is an additional layer of C-elements configured like a WCHB on the input side of the full adder for that purpose.

9.2.2 NCL Half Adder

Like the full adder, Figure 9.2 does not include handshaking, but that is implemented by an additional set of C-elements before the logic. Since NCL is fully-indicating, this is acceptable.

The half adder, unlike the full adder, is only one stage of logic deep. The full adder serializes the carry computation with the sum computation. The half adder also uses a THMNcomp gate, abbreviated TMMNcm. If we label the inputs of the TH24cm gate A through D, the boolean algebra computation is as follows:

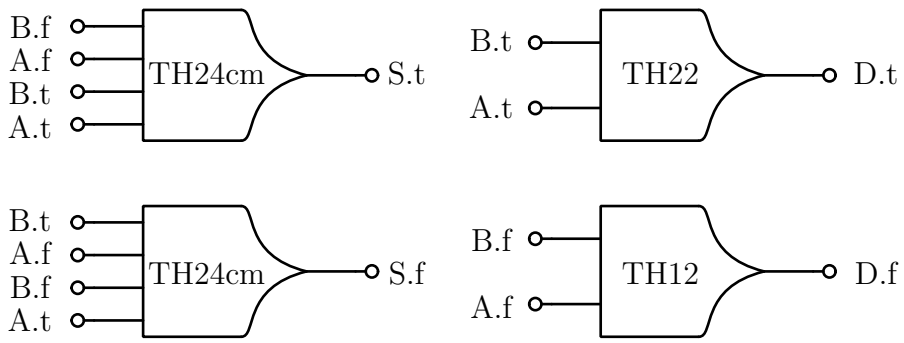


Figure 9.2: NCL Half Adder

$AC + BC + AD + BD$.

9.2.3 PCHB Full Adder

As described in subsection 8.3.5, the Pre-Charge Half Buffer template is ideal for implementing logic. Figure 9.3 is simply Figure 8.3 where the pull-down network has been replaced with a transistor network implementing the desired logic function. In this case, it implements a full adder. Of course, I did not develop this particular implementation [44].

The PCHB full adder takes three e1of2 channels as input, representing the A, B, and carry in (C) operands. It produces two e1of2 channels as output, representing the sum (S) and carry out (D) results. Thus, there are four total output data rails, as can be seen in Figure 9.3. The L.e signal is wire copied to A.e, B.e, and C.e to acknowledge the input signals. I included a singular error in the figure to check to see if you are reading this paragraph. The sum true and false signals are actually switched, but everything else should be correct. Note that the sum and carry out channel enables are treated individually. The circuits outlined in Figure 9.3 are actually simplified for clarity. The input validity signals are actually null signals

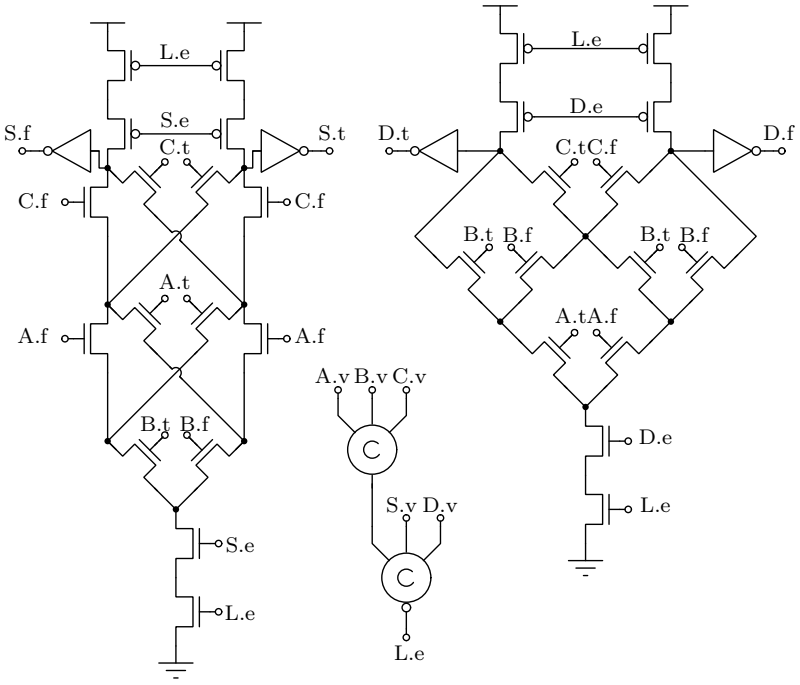


Figure 9.3: PCHB Full Adder

calculated by NOR2 gates, for example.

9.2.4 PCHB Half Adder

The half adder shown in Figure 9.4 is simply a pruned version of the full adder shown in Figure 9.3. It has the same reading check error in the figure. Do note that a half adder takes one less input channel, i.e. there is no carry-in.

The carry out rails for the half adder are unbalanced, but this is only really a factor due to the shared pulldown network. This is just a simple trick which reduces the overall area. It does significantly complicate the expression in our HDL, however, and also makes staticizer expression more complicated as well.

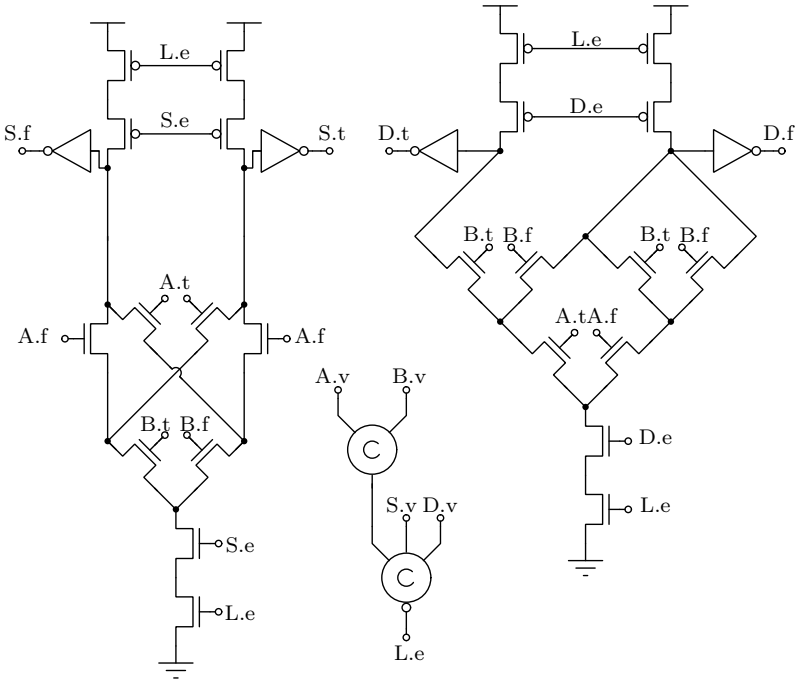
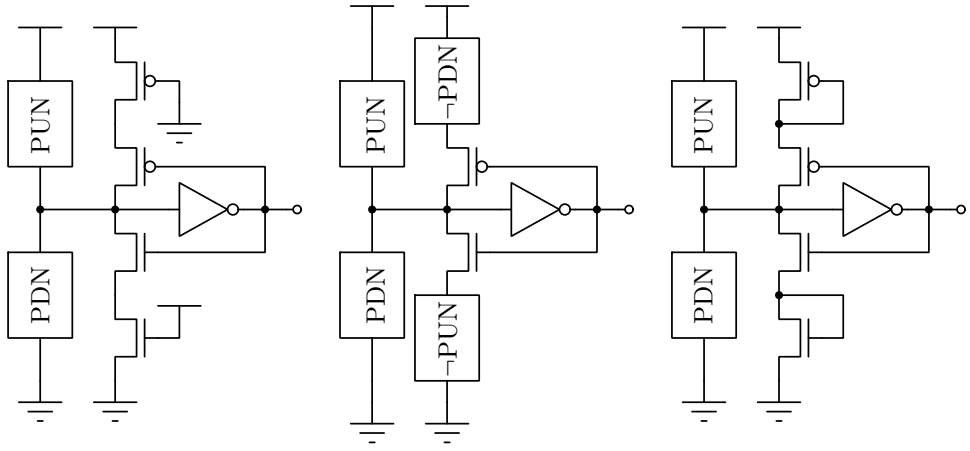


Figure 9.4: PCHB Half Adder

9.3 Methodology

The methodology for the logic study is practically identical to that described in section 6.3 and section 8.4. The only major difference is that there is now more than one input and output channel. Each adder accepts an A and B operand and produces a sum and carry out result. The full adder will also accept a carry in operand. Each Device Under Test (DUT) thus has up to 5 data ports. While the input streams are pseudorandom, the output tokens checked to ensure that the circuits are actually adding properly. Incorrect values result in the DUT being flagged as a failure and discarded from consideration.

While some link structures can be quite complex, e.g. STATS (chapter 6), most are relatively simple. Much of the complexity in the decision-making comes from buffer insertion and sizing considerations. Logic, however, can have much more complicated transistor structures. Many asynchronous CMOS gates are not



Weakened Inverter Combinational Feedback Diode Feedback

Figure 9.5: Staticizer Types

complimentary, i.e. the boolean logic expression for the pullup network is not the negation of the pulldown network. As such, we often require some sort of staticizer structure to hold the state of the output node of a gate against charge sharing and leakage. Typically that is accomplished with a weak feedback inverter, but a combinational feedback design is also possible, as shown in Figure 9.5. Combinational feedback is often much more expensive than the weak-feedback staticizer, depending on the complexity of the gate itself. As an addition degree of freedom, I allow hopTK to choose between several different staticizer options, namely no staticizer, weak feedback, combinational, and diode feedback.

9.4 Evaluation and Discussion

As in section 8.5, we have results for 90 nm, 65 nm, and 28 nm. Half adders are simpler than full adders, so they will always outperform full adders in energy and area for the same frequency. It is not realistic to compare a full versus a half adder, as they implement different functions, but I have included all four adder types on

the same set of plots to reduce space.

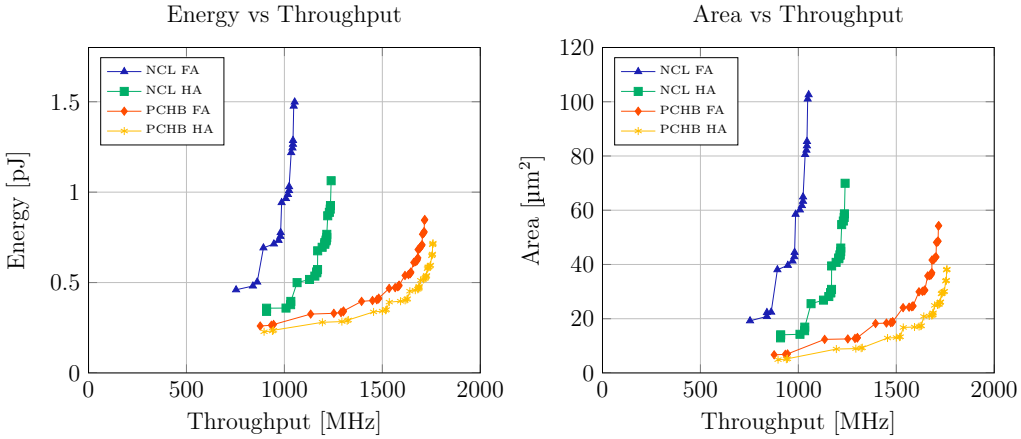


Figure 9.6: 90 nm Adder Stage

The first obvious conclusion is that the PCHB-based adders significantly outperform the NCL adders. At the low end of the frequency range, around 1 GHz, it appears that the PCHB adders are only slightly outperforming the NCL adders. However, as the frequency increases, the relative cost in energy and area for a frequency gain is much higher for NCL. This is largely due to the number of gate transitions involved in an NCL implementation.

A single NCL gate has two transitions, as does a PCHB stage. However, a single PCHB stage implements the desired functionality, it requires several series NCL gates to attain the same functionality. As discussed earlier, the NCL gates depicted in section 9.2 do not include the handshaking circuitry, which is implemented as upstream C-elements. At bare minimum, you already have two transitions due to the C-element. Each NCL half adder is thus four transitions and the full adder 6. You can see effects of the increased number of transitions reflected in the frequency disparity of the NCL half and full adders.

Regarding area, you can see the costs of sizing in the stair-step-like shape of the area plots for NCL—multiple gates are being upsized at the same time, resulting

a discrete step as opposed to the more gradual shape of the PCHB curves.

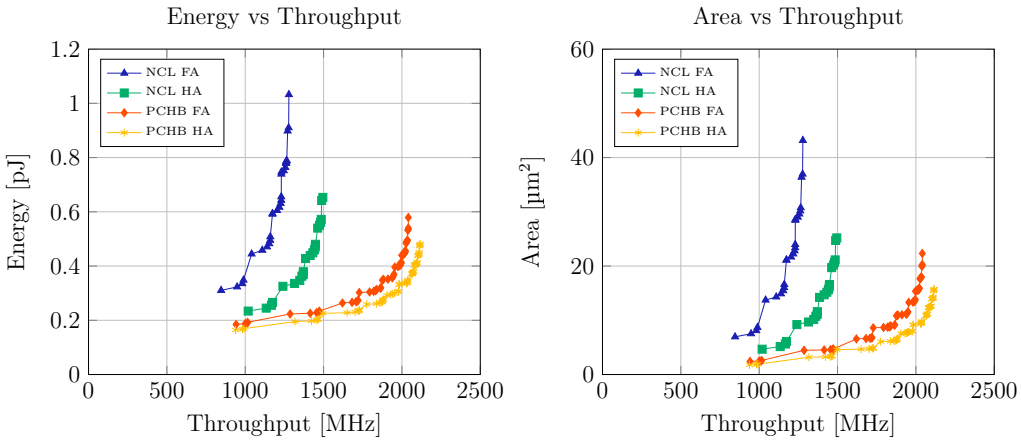


Figure 9.7: 65 nm Adder Stage

Though the PCHB half and full adders are technically different circuits, comparing Figure 9.3 and Figure 9.4 shows very clearly that the PCHB half adder is simply a pruned version of the PCHB full adder. We see this reflected in how the Pareto fronts track closely for both frequency-energy and frequency-area.

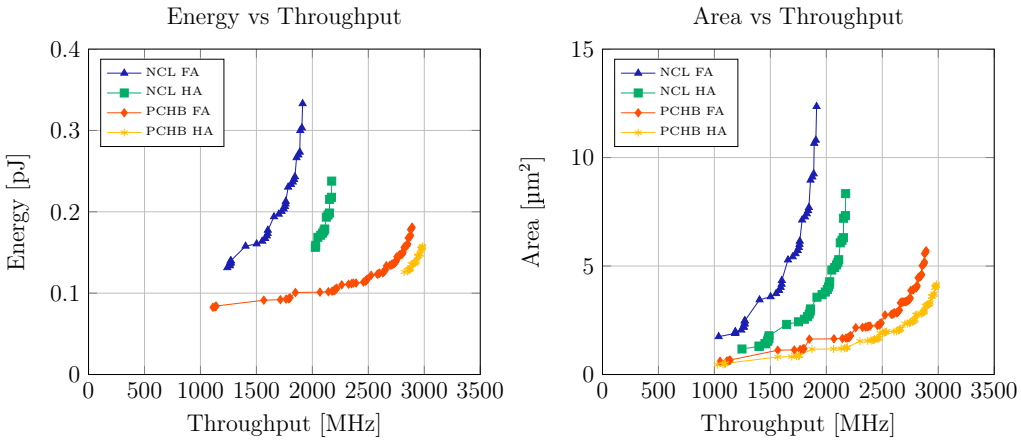


Figure 9.8: 28 nm Adder Stage

As for the choice of staticizer, most if not all results presented in Figure 9.6, Figure 9.7, and Figure 9.8 are either not using a staticizer or using the diode-connected staticizer as shown in Figure 9.5. This is certainly not surprising, as the

feedback element of staticzers act in direct opposition to a valid gate transition, which costs energy. Additional transistors cost area as well.

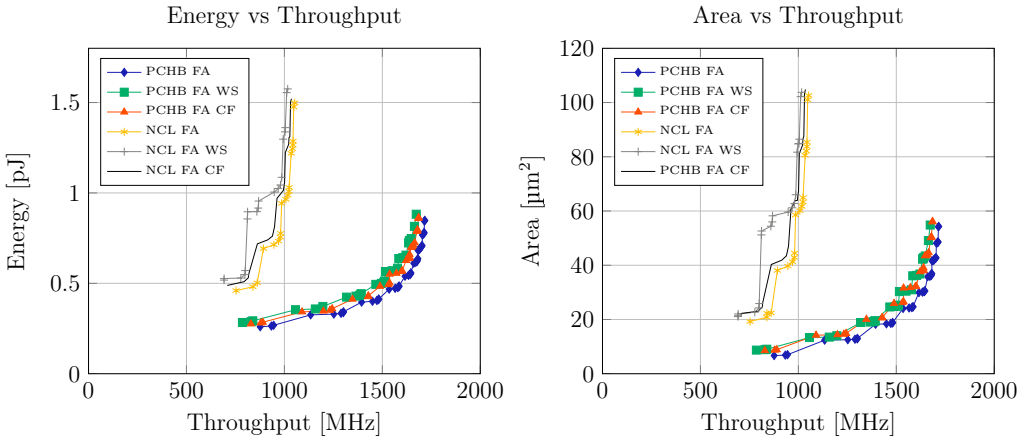


Figure 9.9: 90 nm Adder Staticizer Study

Figure 9.9 shows different staticizer configurations for NCL and PCHB full adders. The traces without a suffix are the original traces from Figure 9.6, and the other two traces with WS and CF appended show the appropriate pareto front for Weakened Staticizer and Combinational Feedback respectively. Ranking in terms of efficiency, no or diode staticizer is the most efficient in terms of energy and area cost, followed by combinational feedback and finally weakened staticizer.

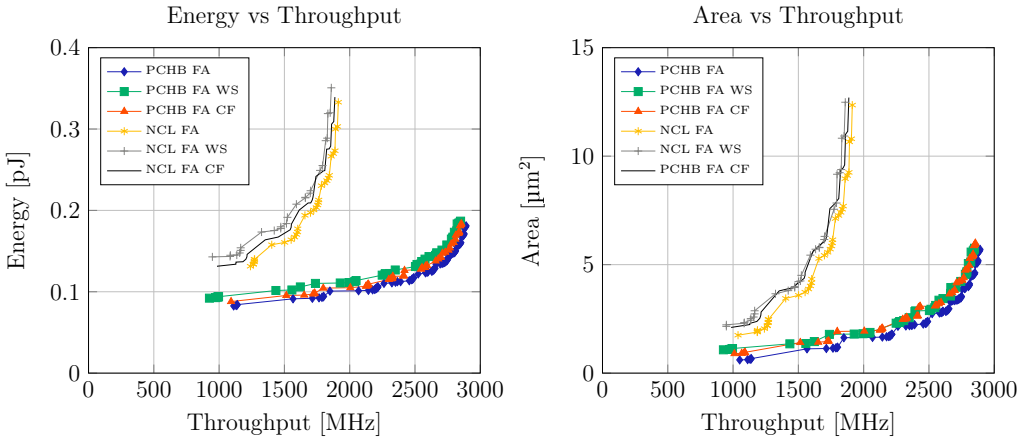


Figure 9.10: 28 nm Adder Staticizer Study

The actual area cost is likely to be much higher for weakened staticizer, as

typically the weakening transistors are long-gate transistors. While long transistors are possible, most deep submicron processes such as our 28 nm process, have fixed gate lengths. The end result is a series chain of minimum length transistors and the associated area bloat due to design rules. The methodology as it stands does not capture the effects of this bloat, as we estimate a lower-bound on area by the $W \times L$ product of transistor dimensions.

Figure 9.10 shows the different staticizer configuations in 28 nm, just to illustrate that the trends remain the same for a more advanced technology node. In general, all three staticizer options are largely equivalent based on this data. This may be true in practice for the energy consideration, but the area will increase dramatically once placement and routing effects are considered.

9.5 Criticisms

So why bother with NCL?

I am not suggesting that you do. That said, if you are in a situation where you have a ready-built, validated, and tested NCL library, it may make sense from a productivity standpoint to use NCL. What the above results show is the approximate cost of moving to NCL from PCHB, so you can make an informed decision.

9.6 Executive Summary

Logic is a critical part of any chip design. To show that our methodology is capable of providing useful information about logic circuits, we implemented half and full

adders in two different logic families, PCHB and NCL. The NCL logic family is intended to provide an asynchronous analogue to the synchronous standard cell toolflow. Designs are synthesized targeting a library of 40 some odd NCL cells, which are then place and routed. The primary argument behind NCL is that it offers the traditional benefits of asynchronous design—robustness to variation, delay insensitivity, etc—while retaining the productivity of a typical toolflow.

PCHB, by contrast, is much more full-custom. The PCHB itself is merely a template, providing sequencing and handshaking control structures for a designer to attach to logic. Logic is implemented as the pulldown stack only, as PCHB stands for Pre-Charge Half Buffer. Evaluation is accomplished in a domino logic style, wherein the handshaking sequence precharges the node high, and the appropriate pulldown path resolves the desired circuit state. The PCHB template does allow a much greater degree of flexibility as well as greater opportunity for optimization when implementing logic, but this has tradtionally come at the cost of design time. Fortunately, through tools such as cellTK [40], we have access to similar levels of automation and tool support for the PCHB template, making PCHB an attractive design style for architects.

Evaluating the half and full adders in a frequency-energy and frequency-area design space, we see that PCHB-template adders outperform the NCL adders. Half adders are also more energy and area efficient than full adders, although this not a surprise nor is it a fair comparison since they implement different functionality.

The methodology also allows us to examine different staticizer topologies. Staticizers are state-holding structures to combat the effects of charge sharing and leakage on domino and non-combinational gates. Unsurprisingly, adding more transistors costs more energy and area, so the best performing configurations did not

include any staticizers at all. Unfortunately, this is not typically recommended for asynchronous circuit design, so we must pay a small penalty in energy and area to retain the desired functionality.

The bottom line for logic is that NCL adders are inferior to PCHB adders. The traditional argument that an NCL-based toolflow is more productive than a PCHB-based one is no longer entirely valid, due to the advent to tools such as cellTK. Unless there is an extremely compelling reason to use NCL, I recommend PCHB.

CHAPTER 10

CONCLUSION

> Getting a PhD doesn't mean you're smart. It just means you have a high pain tolerance.
>
> Sean Pieper

This thesis proposes a methodology using heuristic optimization techniques to obtain a well-populated, evenly distributed Pareto front of points in a multi-objective design space for a design of interest. The intent of the methodology is not to replace the designer but rather to allow a designer to focus their efforts on the higher levels of design abstraction, namely the microarchitecture. The methodology automates lower-level design space explorations and reports results in a way that builds designer intuition and definitely answers the question "Which is better, A or B?" by providing a graphical representation of the design space.

hopTK, the shoddy Python implementation of my proposed methodology, was used to evaluate a variety of single- and multi-bit links as well as logic. For links, the Single-Track Full Buffer (STFB) design of Ferretti and Beerel [24] was a clear winner in most situations, although the standard Weak-Conditioned Half Buffer was a very strong contender as well.

While I did not cover logic styles or gates in exhaustive detail, the Pre-Charge Half Buffer (PCHB) template performed better than Null Convention Logic (NCL) in the usage case of half and full adders. NCL traditionally has been touted as a high-productivity asynchronous logic family, as it allows you to leverage standard cell toolflows. However, recent advances in asynchronous tools such as cellTK [40] have brought similar levels of productivity to PCHB and related logic families as well.

One additional contribution of this thesis is a description of the Single-Track Asynchronous Ternary Signaling (STATS) protocol, which directly addresses the high wiring cost of delay insensitive data encodings. It offers a 3× reduction in wiring cost when compared against WCHB, and a 2× improvement over STFB for single-bit links. Unfortunately, that wiring cost reduction comes at a steep energy and area cost, rendering STATS generally unattractive except in the most wire-starved cases such as Thru-Silicon Vias (TSVs).

This thesis presents three case studies: single-bit links, multi-bit links, and adder units. While individually, none of these case studies are particularly compelling or impressive, they do illustrate the potential of the methodology. The methodology and its hopTK implementation did allow a single individual to understand and evaluate a reasonably large family of circuits. The practicality of the methodology in industry may be a subject for further debate, but as an educational tool in the graduate arena it has merit. Thank you for your time and attention, dear reader.

BIBLIOGRAPHY

[1] E Afacan, G Berkol, A E Pusane, G Dundar, and F Baskaya. Adaptive sized Quasi-Monte Carlo based yield aware analog circuit optimization tool. In *CMOS Variability (VARI), 2014 5th European Workshop on*, pages 1–6. IEEE, 2014.

[2] F Akopyan, C Otero, D Fang, S Jackson, and R Manohar. Variability in 3-D integrated circuits. *IEEE CICC*, pages 659–662, September 2008.

[3] C.J Alpert, A Devgan, J P Fishburn, and S T Quay. Interconnect synthesis without wire tapering. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 20(1):90–104, 2001.

[4] T Back, F Hoffmeister, and H P Schwefel. A survey of evolution strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[5] 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 *IEEE ASYNC*, pages 132–140, 2003.

[6] K Banerjee, S.J Souri, P Kapur, and K Saraswat. 3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. In *Proc. IEEE*, 2001.

[7] S Borkar. 3D integration for energy efficient system design. In *IEEE DAC*, pages 214–219, 2011.

[8] N Chase, M Rademacher, E Goodman, R Averill, and R Sidhu. A Benchmark Study of Multi-Objective Optimization Methods. Technical Report BMK-3021, Red Cedar Technology.

[9] Chunhong Chen, Xiaojian Yang, and M Sarrafzadeh. Potential slack: an effective metric of combinational circuit performance. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 198–201. IEEE, 2000.

[10] A R Conn, P K Coulman, R A Haring, G L Morrill, C Visweswariah, and C W Wu. JiffyTune: circuit optimization using time-domain sensitivities. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 17(12):1292–1309, 1998.

[11] B De Smedt and GGE Gielen. Watson: Design Space Boundary Exploration and Model Generation for Analog and RF IC Design. *... -Aided Design of Integrated Circuits ...*, 2003.

[12] Mark E Dean, Ted E Williams, and David L Dill. Efficient self-timing with level-encoded 2-phase dual-rail (LEDR). In *Advanced Research IN VLSI*, pages 55–70. IEEE ICCD, April 1991.

[13] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Kan-GAL report*, 200001, 2002.

[14] A Deutsch, P.W Coteus, G.V Kopcsay, H.H Smith, C.W Surovic, B.L Krauter, D.C Edelstein, and P.L Restle. On-chip wiring design challenges for gigahertz operation. In *Proc. IEEE*, pages 529–555, 2001.

[15] Li Ding and P Mazumder. Dynamic noise margin: definitions and model. In *IEEE VLSI DESIGN*, pages 1001–1006, 2004.

[16] Alex Doboli, Adrian Nunez-Aldana, Nagu Dhanwada, Sree Ganesan, and Ranga Vemuri. Behavioral synthesis of analog systems using two-layered design space exploration. In *DAC '99*, pages 951–957, New York, New York, USA, 1999. ACM Press.

[17] J Ebergen, B Coates, and A Lee. Long-Distance On-chip Communication Using GasP. *IEEE ASYNC*, pages 109–116, 2011.

[18] Magdy A El-Moursy and Eby G Friedman. Optimum wire sizing of RLC interconnect with repeaters. *Integration, the VLSI Journal*, 38(2), December 2004.

[19] H Esbensen and E S Kuh. Design space exploration using the genetic algorithm. In *ISCAS-96*, pages 500–503. IEEE, 1996.

[20] David Fang. *Width-adaptive and non-uniform access asynchronous register files*. PhD thesis, Cornell University, January 2004.

[21] K M Fant and S A Brandt. NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. In *Application Specific Systems, Architectures and Processors, 1996. ASAP 96*, 1996.

[22] Karl M Fant and Scott A Brandt. NULL Convention LogicTM. Technical report, Theseus Logic, Inc, 1997.

[23] T Felicijan and S.B Furber. An asynchronous ternary logic signaling system. *IEEE VLSI*, 11(6):1114–1119, 2003.

[24] M Ferretti and P.A Beerel. Single-track asynchronous pipeline templates using 1-of-N encoding. In *IEEE DATE*, pages 1008–1015, 2002.

[25] M Ferretti and P.A Beerel. High performance asynchronous design using single-track full-buffer standard cells. *IEEE SSC*, 41(6):1444–1454, 2006.

[26] J P Fishburn and A E Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing - Springer. *The Best of ICCAD*, 2003.

[27] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 2012.

[28] S.M Gilla, M Roncken, and I Sutherland. Long-Range GasP with Charge Relaxation. *IEEE ASYNC*, pages 185–195, 2010.

[29] P Golani and P.A Beerel. High-performance noise-robust asynchronous circuits. *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, 00, 2006.

[30] R Gonzalez, B M Gordon, and M A Horowitz. Supply and threshold voltage scaling for low power CMOS. *IEEE JSSC*, 32(8):1210–1216, 1997.

[31] D Harris, R.F Sproull, and I.E Sutherland. *Logical Effort: Designing Fast CMOS Circuits*. M. Kaufmann, 1999.

[32] T R Harris, S Priyadarshi, S Melamed, C Ortega, R Manohar, S R Dooley, N M Kriplani, W R Davis, P D Franzon, and M B Steer. A Transient Electrothermal Analysis of Three-Dimensional Integrated Circuits. *IEEE CPMT*, 2(4), 2012.

[33] B Hill, R Karmazin, C T O Otero, J Tse, and R Manohar. A split-foundry asynchronous FPGA. In *Custom Integrated Circuits Conference (CICC), 2013 IEEE*, pages 1–4, 2013.

[34] Ron Ho, Kenneth W Mai, and Mark A Horowitz. The future of wires. In *Proc. IEEE*, pages 490–504, 2001.

[35] M Horowitz and W Dally. How scaling will change processor architecture. In *IEEE SSCC*, pages 132–133, 2004.

[36] M Horowitz, T Indermaur, and R Gonzalez. Low-power digital design. In *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, pages 8–11. IEEE, 1994.

[37] Y I Ismail, E G Friedman, and J L Neves. Figures of merit to characterize the importance of on-chip inductance. *IEEE VLSI*, 7(4):442–449, 1999.

[38] Seobin Jung, Jiho Lee, and Jaeha Kim. Yield-Aware Pareto Front Extraction for Discrete Hierarchical Optimization of Analog Circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 33(10):1437–1449, 2014.

[39] S K Karandikar and S.S Sapatnekar. Fast comparisons of circuit implementations. *IEEE VLSI*, 13(12):1329–1339, 2005.

[40] Robert Karmazin, Carlos Tadeo Ortega Otero, and Rajit Manohar. cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells. In *IEEE ASYNC*, 2013.

[41] Stephen W Keckler, William J Dally, Brucek Khailany, Michael Garland, and David Glasco. GPUs and the Future of Parallel Computing. *IEEE Micro*, 31(5):7–17, 2011.

[42] Christopher LaFrieda and Rajit Manohar. Reducing Power Consumption with Relaxed Quasi Delay-Insensitive Circuits. *IEEE ASYNC*, pages 217–226, May 2009.

[43] How-Rern Lin and Ting Ting Hwang. On determining sensitization criterion in an iterative gate sizing process. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 18(2):231–238, 1999.

[44] A.M Lines. Pipelined Asynchronous Circuits. Master's thesis, California Institute of Technology, June 1995.

[45] Dake Liu and C Svensson. Power consumption estimation in CMOS VLSI chips. *IEEE SSC*, 29(6), June 1994.

[46] Hung-Yi Liu, I Diakonikolas, M Petracca, and L Carloni. Supervised design space exploration by compositional approximation of Pareto sets. In *IEEE DAC*, pages 399–404. IEEE, 2011.

[47] R Mariani, R Roncella, R Saletti, and P Terreni. On the realisation of delay-insensitive asynchronous circuits with CMOS ternary logic. In *Advanced Research in Asynchronous Circuits and Systems*, pages 54–62, 1997.

[48] AJ Martin. Compiling Communicating Processes for Delay-Insensitive VLSI Circuits. *Distributed Computing*, 1986.

[49] AJ Martin. The Limitations to Delay-Insensitivity in Asynchronous Circuits. In *6th MIT Conference on Advanced Research in VLSI*. Proceedings of the 6th MIT Conference on Advanced Research in VLSI, 1990.

[50] A.J. Martin. Asynchronous logic for high variability nano-CMOS. In *IEEE ICECS*, pages 69–72, 2009.

[51] A.J. Martin and M Nystrom. Asynchronous Techniques for System-on-Chip Design. *Proc. IEEE*, 94(6):1089–1120, June 2006.

[52] P B McGee, M Y Agyekum, M A Mohamed, S M Asynchronous Circuits Nowick, and Systems 2008 ASYNC '08 14th IEEE International Symposium on. A Level-Encoded Transition Signaling Protocol for High-Throughput Asynchronous Global Communication. *IEEE ASYNC*, pages 116–127, 2008.

[53] W F McLaughlin, A Mitra, and S M Nowick. Asynchronous Protocol Converters for Two-Phase Delay-Insensitive Global Communication. *IEEE VLSI*, 17(7):923–928, July 2009.

[54] R Murgai. On the global fanout optimization problem. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 511–515. IEEE, 1999.

[55] Omesh Mutukuda. *Unidirectional Multi-Bit FPGA Architecture For Area Efficient Implementation of Datapath Circuits*. PhD thesis, Ryerson University, Toronto, 2010.

[56] Ravi Sankar Parameswaran Nair, Scott C Smith, and Jia Di. Delay-Insensitive Ternary Logic. *IEEE ICCD*, January 2009.

[57] J L Neves and E G Friedman. Optimal clock skew scheduling tolerant to process variations. In *IEEE DAC*, 1996.

[58] M Nystroem. *Asynchronous Pulse Logic*. PhD thesis, California Institute of Technology, California Institute of Technology, 2001.

[59] Naoya Onizawa, Atsushi Matsumoto, and Takahiro Hanyu. Long-Range Asynchronous On-Chip Link Based on Multiple-Valued Single-Track Signaling. *IEICE Trans. Fundamentals*, E95.A(6):1018–1029, 2012.

[60] Paul I. Pénzes and Alain J Martin. Energy-delay efficiency of VLSI computations. *... 12th ACM Great Lakes symposium on VLSI*, 2002.

[61] Jean-Mark Philippe, Ekue Kinvi-Boh, Sebastien Pillement, and Oliver Sentieys. An energy-efficient ternary interconnection link for asynchronous systems. *IEEE ISCAS*, pages 4 pp.–1014, 2006.

[62] A Somani, P P Chakrabarti, and A Patra. A model-based hybrid evolutionary algorithm for fast yield-inclusive design space exploration of analog circuits. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4 pp. IEEE, 2006.

[63] K.S Stevens. Energy and performance models for clocked and asynchronous communication. In *IEEE ASYNC*, pages 56–66, 2003.

[64] K.S Stevens, P Golani, and P.A Beerel. Energy and Performance Models for Synchronous and Asynchronous Communication. *IEEE VLSI*, 19(3):369–382, 2011.

[65] V Sundararajan and S.S Sapatnekar. Fast and Exact Transistor Sizing Based on Iterative Relaxation. *Computer-Aided Design ...*, 2002.

[66] I Sutherland. Micropipelines. *CACM*, 32(6), June 1989.

[67] I Sutherland and S Fairbanks. GasP: a minimal FIFO control. *IEEE ASYNC*, pages 46–53, 2001.

[68] Christer H Svensson and Ji-Ren Yuan. A 3-level asynchronous protocol for a differential two-wire communication link. *IEEE JSSC*, 29(9), September 1994.

[69] Hiran Tennakoon and Carl Sechen. Gate sizing using Lagrangian relaxation

combined with a fast gradient-based pre-processing step. In *ICCAD '02*, pages 395–402, New York, New York, USA, 2002. ACM Press.

[70] S K Tiwary, P K Tiwary, and R A Rutenbar. Generation of yield-aware Pareto surfaces for hierarchical circuit design space exploration. In *IEEE DAC*, pages 31–36. IEEE, 2006.

[71] J Tse, B Hill, and R Manohar. A Bit of Analysis on Self-Timed Single-Bit On-Chip Links. *Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on*, pages 124–133, 2013.

[72] K van Berkel and A Bink. Single-track handshake signaling with application to micropipelines and handshake circuits. In *Advanced Research in Asynchronous Circuits and Systems*, pages 122–133, 1996.

[73] R Weerasekera, M Grange, D Pamunuwa, and H Design Automation Test in Europe Conference Exhibition DATE 2010 Tenhunen. On signalling over Through-Silicon Via (TSV) interconnects in 3-D Integrated Circuits. In *IEEE DATE*, pages 1325–1328, 2010.

[74] Guo Yu and Peng Li. Yield-aware hierarchical optimization of large analog integrated circuits. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 79–84. IEEE, 2008.