A Power/Area Optimal Approach to VLSI Signal Processing

Dejan Marko Markovic

May 18, 2006

A Power/Area Optimal Approach to VLSI Signal Processing

by

Dejan Marko Marković

Eng. (University of Belgrade, Yugoslavia) 1998
M.S. (University of California, Berkeley) 2000

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Spring 2006

The dissertation of Dejan Marko Marković is approved:

Co-chair	Date
Co-chair	Date
	Date
	Date

University of California, Berkeley

Spring 2006

A Power/Area Optimal Approach to VLSI Signal Processing

Abstract

A Power/Area Optimal Approach to VLSI Signal Processing

by

Dejan Marko Marković

Doctor of Philosophy in Engineering – Electrical Engineering

and Computer Sciences

University of California, Berkeley

Professor Robert W. Brodersen, Co-chair

Professor Borivoje Nikolić, Co-chair

The complexity of integrated circuits (ICs) in wireless communication devices has been steadily increasing to support more functionality and new ideas from information theory. Computational requirements can be quite drastic, especially in multi-antenna (MIMO) communication systems which use multi-dimensional signal processing algorithms. The required increase in computational efficiency in MIMO systems can be far greater than the improvements provided by scaling of IC technology alone.

This work will present a methodology for power/area efficient ASIC realization of signal processing algorithms for wireless communications, taking into account unique features of scaled technology such as leakage power and process variation. A sensitivity based optimization framework will be applied to multiple layers of design abstraction: circuits, micro-architecture, and macro-architecture. The proposed approach enables power and area optimizations across the boundary of algorithm, architecture, and circuits, which is essential for creating globally optimal designs.

In wireless baseband chip realization, the design cycle traditionally requires reentering data at various abstraction levels, thus constraining the implementation choices and increasing time-to-market. An approach using a unified design description for algorithm verification and architecture exploration is also presented. The proposed graphical block-based design entry and retargetable design flow provide the ability to track technology features in the process of architectural selection.

As a proof of concept, the design methodology will be demonstrated on a wideband 4×4 MIMO channel decoupling through singular value decomposition. The computational throughput of 70GOPS was implemented with 0.5 million gates at a 100MHz clock and 385mV supply, dissipating 34mW of power. The chip achieves a power efficiency of 2.1GOPS/mW in just $3.5mm^2$ in a standard $90nm$ CMOS process.

Professor Robert W. Brodersen
Dissertation Committee Co-chair

Professor Borivoje Nikolić
Dissertation Committee Co-chair

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

My sincere gratitude goes to my advisors, Professor Bob Brodersen and Professor Bora Nikolić. Their stimulating guidance has sparked many ideas that carried my research. Bob's constant drive for his students to succeed is truly exceptional, and I would not be typing this if it were not for him. The rightness and enthusiasm have been with me at every step along the way. Bora's understanding and technical support has greatly transcended my knowledge about circuit design. I am indebted to both of them for their faith in my research and their tolerance of my weaknesses. Their patient, trustful, and encouraging support has made me a better scholar and a better person. Professor Jan Rabaey and Professor Paul Wright provided thoughtful evaluation of my research proposal and review of this dissertation.

My early work in energy-performance optimization started as a hobby project together with my friend Vlada Stojanović, who was at Stanford University at that time. I am grateful to him and his advisor Professor Mark Horowitz for their contributions in this work. I am especially grateful to professor Jan Rabaey for giving me a unique opportunity to experience teaching as a graduate student and for many inspiring technical discussions. Professor Seth Sanders gave me the opportunity to pursue my Ph.D. at Berkeley. I thank Professor Vojin Oklobdžija for starting my research during my undergraduate diploma project, and his selfless guidance ever since.

The multi-disciplinary, open minded setting of the BWRC created many opportunities to further develop my work in collaboration with others. Special gratitude goes to Danijela Čabrić. She and Vlada have been peers for endless technical discussions and also great friends since college. Ada Poon developed the SVD algorithm that was being used as a demonstration example for my work. Changchun Shi provided the FFC tool for word-length optimization. Rhett Davis did early development of an automated Simulink-to-Silicon design flow, which was later revised by Kimmo Kuusilinna and constantly updated by Brian Richards. Brian helped me understand intricacies of many IC design tools. Engling Yeo, Mike Sheets, Farhana Sheihk and Tufan Karalar shared the burden in setting up the IC design flow. Chen Chang and Pierre-Yves Droz created the BEE, an FPGA based emulation platform, which turned out to be also applicable to my work for ASIC test and debug. Zhengya Zhang shared experience in using the FPGA boards.

Pam Atkinson gave me the opportunity to master my teaching skills by facilitating UC Berkeley distance learning program to working professionals and students.

The support of many fellow students has helped me get through pressures of graduate school. Chris Rudell, Jeff Weldon, Tom Burd, and Ian O'Donnell have been very supportive by sharing their wisdom and experience. I am also thankful to my classmates Fred Chen, Johan Vanderhaegen, Yun Chiu, Radu Zlatanovici, David Sobel, Chinh Doan, Brian Limketkai, Dragan Petrović, Kevin Camera, Hayden So, and Ning Zhang for stimulating discussions and friendship. Nathan Chan has been the resident guru for electronic gadgets and continuous source of various software updates. Kostas Sarrigeorgidis added his unique character to the BWRC group.

Members of the DCDG group engaged in many hard-core discussions about research.

This research was supported by the MARCO/C2S2 program and BWRC member companies. ST Microelectronics provided fabrication support. Special gratitude goes to Gary Kelson, Tom Boot, Brenda Vanoni, and Elise Mills for doing a wonderful job in organizing and administering many activities of the center.

Personal gratitude goes to Siniša for being a great roommate and for his support over the years. Srdjan, Nenad, Bora, Voja, Filip, Dragan, Richard, and Nate have kept my passion for various sports alive. Thanks to Daniel Bukumirović from B92 sports studio in Belgrade, I have also experienced other side of sports, as a reporter. Special thanks go to Vlade, Pedja, and Žarko for making it a truly unique experience.

I am truly indebted to Tomašević-Wakeman family for providing the best housing in Berkeley, which greatly eased my transition into the new world. I thank čika Nikola for lessons about virtually everything. Talking to him is like communicating with a live encyclopedia, apart from his unusual talent to speak almost every language on the planet. Teta Desa has also provided enormous support with her personal advice and encouragement. I have been blessed to meet these people.

I would like to thank Aleksandra for her loving support and understanding for my hectic schedule, especially during chip tapeout. She has been a constant source of inspiration and encouragement.

Finally, and most importantly, I would like to express gratitude to my parents and my brother. This work is a product of their endless love and continued support.

Chapter 1

Introduction

...It is difficult to predict at the present time just how extensive the invasion of the microwave area by integrated electronics will be. The successful realization of such items as ***phased-array antennas****, for example, using a multiplicity of integrated microwave power sources, could completely revolutionize radar.*

G. Moore, 1965

The nature of integrated circuit design has experienced a major change in recent years due to continued scaling of the underlying technology. In the past, the amount of functionality that could be integrated on chip was limited by area; today, power dissipation is the primary limiting factor. Technology scaling over the past four decades has provided about one million times higher integration complexity, and this trend is continuing. Smaller transistors consume less power, but higher transistor density and speed result in overall higher chip power consumption and more generated heat. This is not the only challenge. As technology scales into the deep sub-micron regime (beyond $0.25\mu m$ node) designers are starting to see some unique technological features that they did not have to think about before. Power leakage has been increasing exponentially up to $90nm$ node, and the impact of process parameter variations on power and performance has been increasing with each technology generation. As a result, technology has been degrading, in a way, while design complexity kept increasing. To address the issue of increased variability and leakage, modern technologies offer devices with several thresholds. The multiplicity of transistor thresholds from which to choose gives the designer even more flexibility and makes design more challenging.

1.1 Power-Limited Design

The characteristics of power constraints are different for desktop processors and mobile devices, but in both cases the maximum achievable performance depends on the degree of energy-efficiency in performing a computation. Focusing primarily on performance for high-speed circuits results in too much power dissipation. Focusing only on energy for mobile applications is equally inadequate, since this approach rarely achieves the required performance. The correct optimization either minimizes energy consumption, subject to a throughput constraint, or maximizes the amount of computation for a given energy budget.

At the same time, the complexity of chip implementations has been steadily growing, both in desktop and portable devices. To meet the power challenge, microprocessor designs now use a multi-core approaches to increase overall performance, without increasing the baseline frequency of the operation [2]. Growing demand for mobile, multi-functional devices stresses the issue of power efficiency perhaps even more due to limited energy capacity of portable batteries. Finally, new ideas from communication theory such as using multiple antennas also add to this complexity quite substantially. This dissertation will present general concepts related to power and area minimization while focusing on wireless communication designs.

The scaling of the semiconductor technology has provided an exponential increase in integration complexity, with the number of transistors that can be integrated on a chip doubling every 18 to 24 months. This trend is called “Moore’s Law” after Gordon Moore of Intel who made an observation in 1965 that the number of transistors per integrated circuit doubled about every year. The pace wasn’t able to sustain quite this level, but Moore made a downward revision in 1975, saying that they doubled about every two years. Such a trend has been steadily in place, resulting in integration capability exceeding a billion transistors on chip today. The question is what could a designer do with so many components? The dual-core Itanium® processor is a good example of successfully tracking the historical trend, with 1.72B transistors integrated on a single die, [3]. It is hard to predict what kind of designs will be needed in the future, but the very last sentence of the famous article by Gordon Moore quoted in the beginning of this chapter [4] is quite indicative. The last visionary statement in the article predicted the use of scaled technology for realization of phased-array antennas for radar applications. This dissertation addresses a similar problem in principle: realization of multiple-antenna (MIMO) wireless communication.

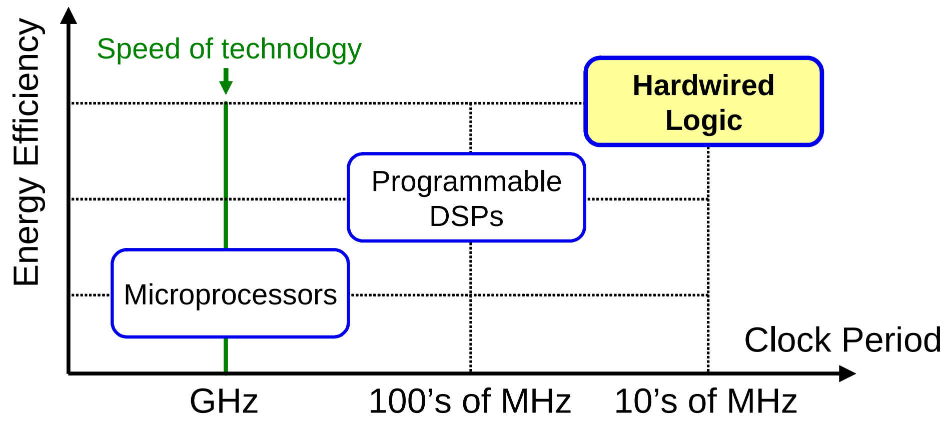


Figure 1.1: Architectural choices for wireless baseband chip design: Microprocessors, Programmable DSPs, Direct-mapped logic.

1.1.1 Architectural Considerations

Before investigating realization of MIMO technology, it is interesting to review basic architectural solutions available for the realization of signal processing algorithms in wireless baseband chips, Fig. 1.1. Digital baseband functionality in wireless local area network (WLAN) chips is typically provided with direct-mapped, hardwired logic[1]. This approach offers the highest energy-efficiency, which is a prime concern in battery operated devices. Another extreme is the heavily time-multiplexed microprocessor architecture, which has the highest flexibility, but it is also the least energy-efficient because of overhead to support the time multiplexing. Between these two extremes are programmable DSP solutions, such as those found in cellular phones.

Another look at the three architectural choices along the horizontal axis reveals that the clock speed required from direct-mapped parallel architectures is significantly

[1]Hardwired logic can achieve a much higher speed in general. Clock rate shown in this plot is for the typical case of a wireless local area network baseband.

lower than the speed of technology tailored for microprocessor designs. The difference in clock rate also hints to architectural differences, for example a faster clock is required for time-multiplexing. This discrepancy can be viewed as an opportunity for further energy and area reduction in chip implementations. It is thus interesting to study an example built on the concept of a direct-mapped architecture to see what kind of power and area improvements can be achieved.

Case Study: A 802.11a WLAN Transceiver

Figure 1.2 illustrates an example of a commercial transceiver chip [5], for the 802.11a wireless LAN standard, [6]. The digital baseband operates with a 80 MHz clock, [7]. In fully parallel direct-mapped transceiver architecture, the clock rate is consistent with an 80 MHz sampling rate of the ADC. Baseband processor executes 40 billion operations per second (GOPS) while consuming 200mW of power in a $0.25\mu m$ CMOS. The total die area that includes digital core, ADCs and DACs, I/Os, and PLL is $46.2mm^2$. The design as such is not the most power/area efficient because the speed of a $0.25\mu m$ technology is greater than the required 80 MHz clock rate. The excess speed could be used to reduce power and area. Power could be reduced by lowering core supply voltage, for example. Area could be reduced by introducing time-multiplexing.

In a commercial environment, design architecture is typically ported to two or three successive technology generations by simply shrinking all device dimensions. Such an approach works well in terms of power and area due to benefits of scal-

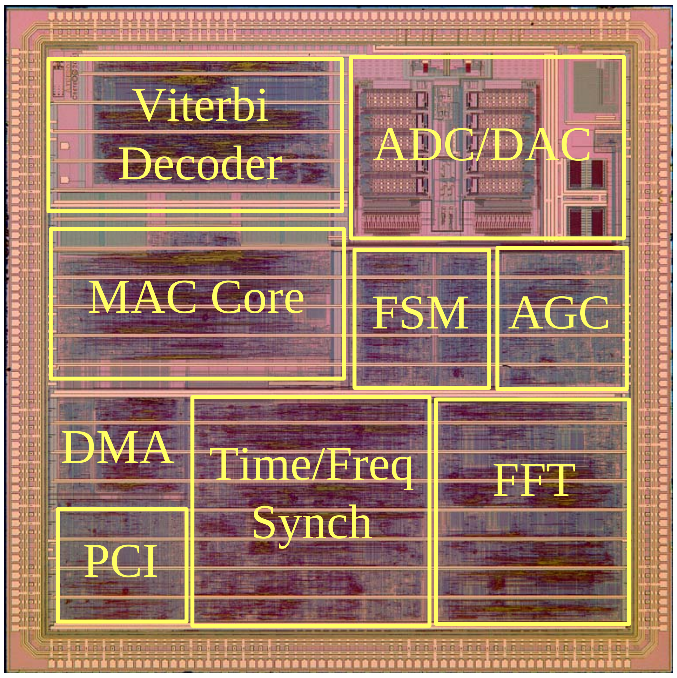


Figure 1.2: Wireless technology today: 802.11a WLAN transceiver chip.

ing, but the original architecture, often called “lead architecture”, would not be the most power/area efficient in the new technology. With scaling of technology, speed-area-power characteristics of the underlying devices change, rendering a previously optimal architecture sub-optimal in the new technology. In order to achieve the best power/area efficiency, the architecture has to track the energy-performance characteristics of the underlying circuits.

1.1.2 Design Productivity

The process of re-designing at the architectural level requires significant amount of engineering resources since it typically involves the close interaction of multiple engineering teams spanning algorithms, architecture, and circuits. The situation is further complicated by the fact that these various teams use different design tools to verify their ideas. Algorithm designers use Matlab or C, for example, while hardware designers like to work with synthesis tools from Synopsys or Cadence. This creates the scenario in which the same design needs to be re-entered multiple times. The ability to re-target an algorithm to a different technology in a power/area efficient way using a unified design description is crucial to improving design productivity. It also enables an efficient way to add flexibility required by future wireless devices.

1.2 MIMO Technology

Future devices are going to be much more complex. To satisfy the growing need for higher capacity and extended range, WLAN devices are moving to using multiple-input multiple-output (MIMO) algorithms as being defined in the 802.11n standard, [6]. Particular interest of this work is to find out how this impacts the digital baseband in terms of power and area. Figure 1.3 shows conceptual diagram of a multi-antenna communication system, requiring multiple transmit and receive antennas. This system will be described in more detail in Chapters 8, 9. For now, observe that the 802.11a chip works with scalar information to estimate the wireless channel. In the MIMO case, the channel information is in a matrix form, where elements of the matrix represent transfer function between various antenna pairs. The complexity of matrix-based signal processing for an $N \times N$ MIMO channel would be approximately

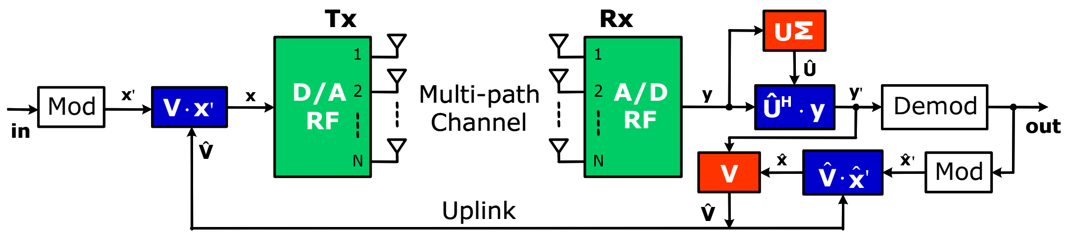


Figure 1.3: Future wireless technology: Multi-antenna (MIMO) communication.

N^2 higher than for the single antenna case, while scaling of technology can provide only linear increase (2×) in computational complexity.

The power and area of a simple direct-mapped parallel implementation would increase quadratically with the number of antenna-pairs N, for a fixed process. Such complexity outpaces Moore's law in terms of area. A simple way to reduce area, demonstrated in Chapter 4, is to use time-multiplexing, but this increases power dissipation. On the other hand, supply voltage scaling with parallelism increases the area. Simultaneous optimization for power and area is therefore needed.

1.3 Overview of Previous Work

The methods for achieving optimum performance are well explored, [8]. Establishing the balance between performance and power consumption has been a popular research topic in the past as well. An optimum in the energy-delay space has been searched for through minimization of objective functions that combine energy and delay. Today, designers work at each level of design abstraction, Fig. 1.4 separately, which leads to sub-optimal designs.

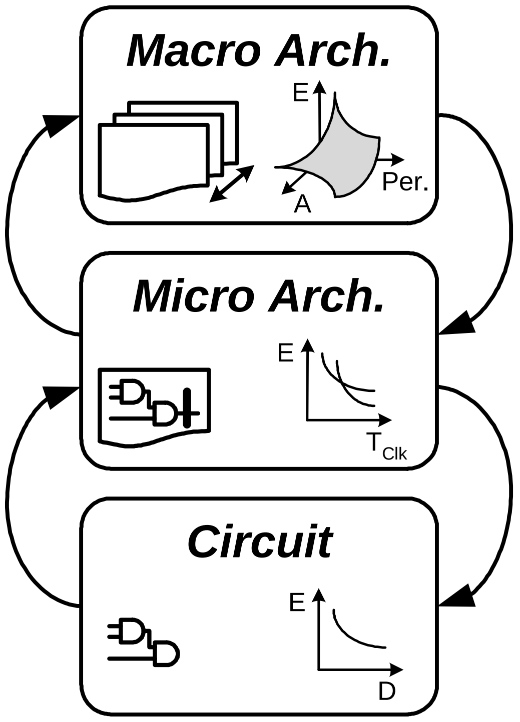


Figure 1.4: Layered optimization approach.

Circuit Techniques

Minimizing the energy-delay product (EDP) [9], [10] of a circuit results in a particular design point in the energy-delay space where 1% of the energy can be traded for 1% of the delay. Although the EDP metric is useful for comparison of different implementations of a design, the design optimization points targeting EDP may not correspond to an optimum under desired operating conditions. Metrics in a general form of $E{\cdot}D^n$ [11] or energy-performance ratio by Hofstee [12] have been used instead. For example, the $E \cdot D^2$ metric [13] puts more weight on the delay than the energy, and is a V_{dd}-invariant metric. Minimizing $E \cdot D^n$, however, has limited applicability since it gives only one (E, D) point in the energy-delay space at which the energy

E is minimized for a fixed delay D. A complete understanding of the energy-delay trade-off for a design is obtained by minimizing the energy subject to an arbitrary delay constraint.

Sizing optimization of digital circuits has been explored extensively resulting in several optimization tools such as TILOS [14], JiffyTune [15] and EinsTuner [16]. Most such tools can at least approximate energy-constrained sizing by constraining the total transistor width available for the circuit. In addition, a number of researchers have derived analytical solutions for area and energy optimization through gate sizing. The analysis is typically restricted to simple logic gates and inverter chains [17], [18]. Like TILOS, a simple analytical timing model is used to guarantee a convex optimization problem, but the delay dependence on V_{dd} and V_{th} is explicitly used to allow multi-variable optimization to be done.

Architectural Techniques

Supply voltage scaling is another common technique that is used to minimize energy under performance constraints. It was one of the key techniques in the low-power DSP work of Chandrakasan et al. [19] and has been practically demonstrated in [20], [21]. With the emerging importance of leakage power consumption, the threshold voltage becomes a critical tuning variable and is generally considered together with supply voltage. Liu and Svensson hinted about the existence of an optimal supply and threshold for a given design [22]. Gonzalez et al. [10] investigated joint supply and threshold voltage scaling for energy-delay product minimization. Kuroda et al. [20] and Nose and Sakurai [23] extended this work and proposed closed-form expressions for the optimum supply, threshold, and leakage-to-switching power ratio.

Various architectural techniques in the energy-area-performance space are evaluated in order to minimize power and area. Prior work applied similar techniques to simple building blocks such as FIR filters [24] in standard VLSI design environments. The methodology presented in this dissertation is scalable to large degrees of complexity. This work uses the graphical Matlab/Simulink environment, familiar to both algorithm developers and chip designers, thus giving practical insight to algorithm developers as well providing a better understanding of the algorithms to VLSI architects.

1.4 Dissertation Outline

Chapter 2 introduces a general sensitivity-based optimization framework. Sensitivity is defined as the absolute gradient of a cost function to a constraint when changing some design variable. In a multi-variable optimization, the optimization procedure exploits the tuning variable with the largest marginal return. A fixed point in the optimization is reached when marginal costs of tuning all tuning variables are equal. In this work sensitivities are used to formalize the trade-off between energy and performance.

In order to create more globally optimal designs, the optimization is applied to several design abstraction layers as shown in Fig. 1.4. Prior work is expanded by applying a sensitivity-based optimization framework to multi-variable optimizations across several abstraction layers, allowing for a comparison of power, area and performance for different solutions. Optimization is performed at three layers of abstraction: system architecture, micro-architecture, and fixed circuit topology optimization.

Chapter 3 presents circuit-level optimization that minimizes energy subject to delay constraint using gate size (W), supply (V_{dd}) and threshold voltage (V_{th}) as variables. All variables are jointly considered to yield the highest energy-efficiency. The sensitivity framework is applied to circuit logic blocks with significantly different topologies: an inverter chain, a memory decoder, and a 64-bit adder. The effectiveness and scope of all variables in the optimization are analyzed. Since circuit-level optimization is only effective in a very narrow performance range, micro-architectural optimization is considered.

Chapter 4 takes results from the circuit-level optimization and provides insight for the micro-architectural optimization. The best choice of circuit topology and optimal level of parallelism is made by investigating a set of optimal energy-delay trade-off curves corresponding to various circuits. Design issues such as determination of the optimal balance between leakage and switching power, the optimal V_{dd} and V_{th}, and the investigation of energy-area trade-offs are addressed.

Chapter 5 extends the lessons from micro-architectural optimization and reviews signal processing techniques such as data-stream interleaving and folding. Implementation cost in terms of cycle latency is also considered. Since different arithmetic blocks vary in complexity, this also raises a question how to optimally distribute pipeline registers, both in feed-forward and feedback based computations. To address this, the methodology for loop retiming is presented, followed by analysis of iterative square rooting and division techniques.

Chapter 6 provides general classification of MIMO algorithms starting from fundamental diversity versus spatial multiplexing trade-off. An overview of diversity methods such as BLAST and spatial-multiplexing methods such as QR-decomposition and SVD is provided. The goal is to classify basic concepts used in MIMO detection and investigate the computational complexity involved. The chapter concludes with a brief survey of published ASIC implementations related to MIMO baseband signal processing.

Chapter 7 presents the use of the design optimization methodology within a commercial CAD environment. The procedure starts with design entry in Matlab/Simulink, followed by word-length optimization. Technology specific block characterization is used to determine the optimal micro-architecture and the operating supply voltage. Finally, the design is taken through highly automated back-end synthesis tools that incorporate gate sizing and supply voltage optimization. Design procedure is illustrated on the inverse square rooting example.

Chapter 8 applies concepts developed in previous chapters on design and optimization of a 4×4 adaptive MIMO algorithm, to demonstrate the level of complexity that can be supported using proposed methodology. An ASIC chip is fabricated to verify the validity of the methodology.

Chapter 9 discusses measured results from the ASIC chip. An FPGA-based test setup for efficient ASIC verification is also described.

Chapter 10 concludes this work and proposes some steps for future research.

Chapter 2

Sensitivity-Based Optimization

This chapter briefly introduces the general sensitivity-based optimization framework that will be used throughout this dissertation. The framework quantifies the cost of tuning each variable in the design and balances sensitivities to all design variables in order to minimize the cost function subject to design constraints.

2.1 Optimization Principle

Sensitivity analysis is based on simple gradient expression as given by Eq. (2.1),

$$S_A(A_0) = \left.\frac{\partial E/\partial A}{\partial D/\partial A}\right|_{A=A_0} \tag{2.1}$$

where E is a cost function and D is a design constraint. For example, E could be the energy dissipation and D could be the delay in a digital circuit. Both E and D

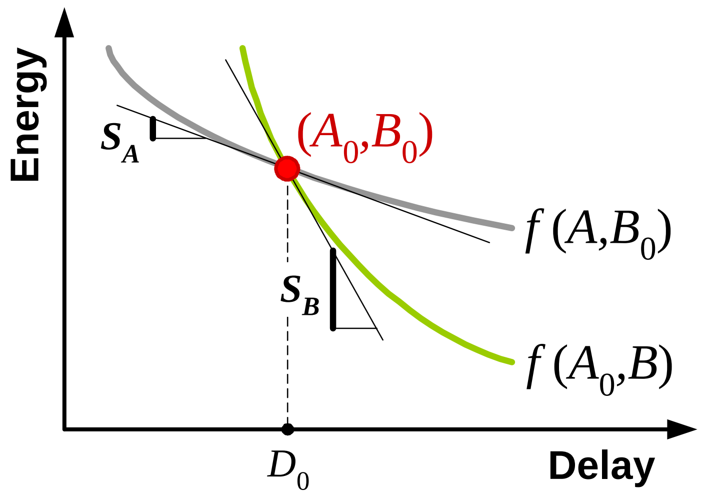


Figure 2.1: Illustration of Energy-Delay sensitivity.

are functions of some general design parameter A, for example supply voltage. The sensitivity S_A to variable A provides intuition about the profitability of optimization, i.e. how much change in E and D is achieved by tuning design variable A around operating point A_0.

The concept of sensitivity is illustrated in Fig. 2.1 for an example of energy-delay trade-off, which is described with function $f(A, B)$. Assuming that the design is positioned at the operating point (A_0, B_0), the sensitivity to variable A, for example, is simply defined as the slope of the line tangent to function $f(A, B_0)$. The function $f(A, B_0)$ is obtained by varying variable A around (A_0, B_0). In this example, variable B has higher sensitivity than variable A resulting in a sub-optimal design.

The key concept to realize is that at the optimal solution point, sensitivities should be equal. An optimization mechanism based on this principle is illustrated for the general energy-delay trade-off shown in Fig. 2.2. The goal is to minimize energy

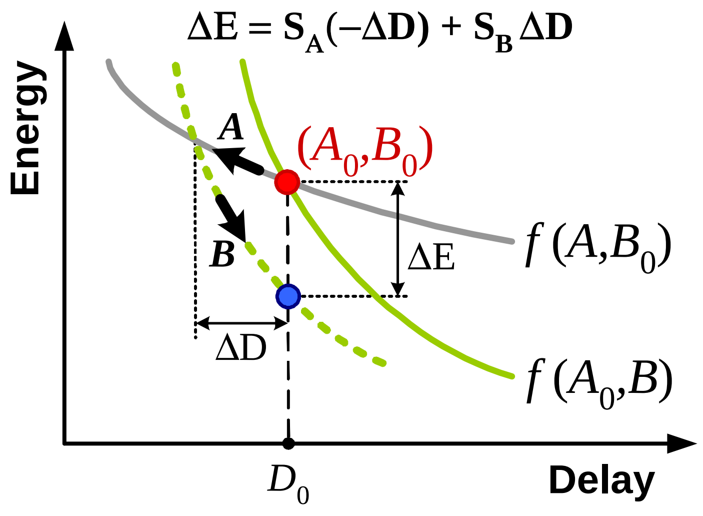


Figure 2.2: Process of balancing sensitivities.

subject to a delay constraint D_0. Assume that the operating point (A_0, B_0) is the initial design point. Starting from this point, delay can be reduced ΔD by lowering the value of variable A. This results in an energy increase[1] proportional to sensitivity S_A, $\Delta E = S_A \cdot (-\Delta D)$. The delay slack ΔD provided by tuning variable A can be utilized by tuning a more sensitive variable B to return to the starting delay point D_0. Overall, an energy reduction of $\Delta E = (S_B - S_A) \cdot \Delta D$ is achieved because energy cost of tuning variable B is higher than the energy cost of tuning variable A. In other words, the energy reduction is possible because sensitivities S_A and S_B are not equal at the initial operating point (A_0, B_0). A fixed point in the optimization, corresponding to $\Delta E = 0$, is therefore achieved when all sensitivities are equal. During the process of balancing sensitivities, some variables may reach the upper or lower limit, resulting in a constrained solution.

[1]Note that S_A is negative. The minus sign represents decrease in delay.

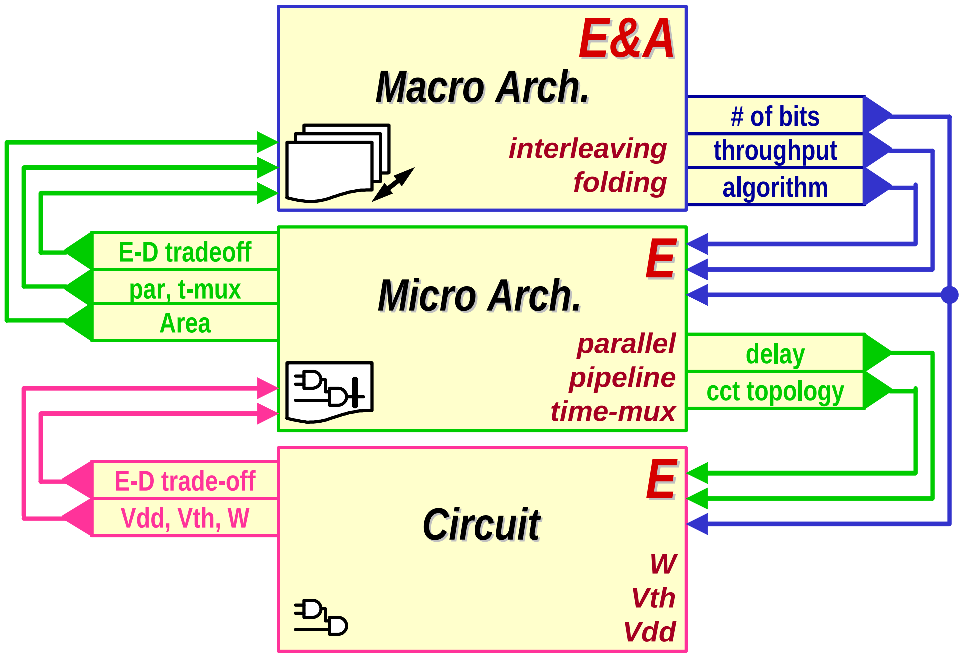


Figure 2.3: Block diagram illustrating various abstraction layers in the optimization. Energy is the objective function at the circuit and micro-architectural layers, while achieving proper energy-area trade-off is the objective at the macro-architectural layer.

2.2 Layered Optimization Approach

The general sensitivity based framework is applied to multiple layers of abstraction as illustrated in Fig. 2.3. The goal is to minimize the overall cost function of a design, such as energy or area, subject to some constraint, such as performance.

The nature of performance constraints is different at various abstraction layers in the optimization. For instance, the performance of a circuit is measured by the circuit delay, while the performance of a micro- or macro-architecture is related to the cycle time or the number of instructions per cycle. Each new layer in the optimization introduces more degrees of freedom, such as the level of parallelism at the micro-architectural layer or the area of the macro-architecture. However, designs of higher complexity can be still optimized based on the optimal energy-performance trade-offs of their building blocks. This is computationally much more efficient than performing large-scale optimization at the gate level.

The procedure is illustrated in Fig. 2.3, where the goal is to be able to obtain trade-offs at individual layers and understand the interaction among the different layers. Also, if the desired operating point in E-D space is known, which corresponds to some sensitivity, then the optimization problem can be decoupled into a sequential procedure where independent variables are optimized individually. Decoupling the problem into independent spaces provides good intuition about various trade-offs comprising the overall design optimization.

At the circuit level, the goal is to minimize the energy subject to the delay constraint in data-path logic and flip-flops, using the gate sizing, supply and threshold voltage as variables. The result is the values of optimization variables for the optimal energy-delay (E-D) trade-off. Traversing upward through the design abstraction stack, additional variables in the optimization can be considered, such as the amount of parallelism or time-multiplexing at the micro-architecture level. (Figure 2.3 is by no means complete, it illustrates some of the most commonly used variables.)

At the micro-architecture level, the optimal energy-delay trade-off curves from the circuit level are used to hierarchically extend the optimization. Information about the optimal E-D trade-off in individual circuit blocks is used, along with top-level specifications such as throughput, word-size, algorithm, etc. These trade-off curves coupled with optimal W, V_{dd}, and V_{th} are strategically combined to obtain the optimal energy-performance trade-off for circuit macros. With this optimal energy-performance curve, the designer can select the appropriate circuit topology for each pipeline stage or choose an optimal level of parallelism based on the circuit optimization results. However, the optimal V_{dd} and V_{th} in the individual circuits are rarely the same for all pipeline stages. In order to achieve the most energy-efficient solution under a single V_{dd} and V_{th} combination, several iterations may be required to optimize all the circuits under that particular V_{dd} and V_{th}. This includes finding the optimal V_{dd} and V_{th} for a given architecture based on balancing the leakage and switching components of energy under some performance constraint. The result is an optimal E-D curve, along with area information, that becomes important when parallelism and time-multiplexing are applied to large blocks. Finally, at the top-level the goal is to find the optimal compromise between the energy and area of the implementation.

The general framework presented in this chapter will be applied to multiple design abstraction layers. In the next two chapters, individual optimizations at the circuit-level (Chapter 3) and the micro-architectural level (Chapter 4) are analyzed.

Chapter 3

Circuit Optimization

This chapter presents methods for efficient energy-performance optimization at the circuit level. Optimizations are performed based on the alpha-power law gate delay model and switching and leakage components of energy. The sensitivity-based approach is applied to energy minimization subject to a delay constraint, using gate size, supply and threshold voltage as variables. General energy-delay sensitivity to each of the variables is derived to provide an understanding of the effectiveness of various variables in multi-variable optimization. The impact of basic topological characteristics found in logic gates such as branching and path reconvergence is also analyzed and illustrated by examples of an inverter chain, a memory decoder, and an adder. It is demonstrated that energy savings of about 65% can be achieved without delay penalty with equalization of sensitivities to sizing, supply, and threshold voltage in a 64-bit adder, compared to the reference design sized for minimum delay.

3.1 Technology Calibration

The energy and the delay of a logic gate are functions of its size, supply voltage, and transistor threshold voltage. In order to calculate the sensitivities of larger logic blocks comprised of simple logic gates, it is necessary to develop simple and accurate models of the energy and delay of the gate. For the optimization purposes, the models work with continuous variables. General conclusions derived from continuous optimizations are applied to synthesis environment that works with discrete variables, such as standard-cell sizes (Chapter 7).

3.1.1 Delay Model

While there are many different models that can be used, our prior work [25], [26] suggests the use of the alpha-power law model of [27] as a baseline for derivation of the gate delay formula:

$$\begin{aligned} t_p &= \frac{K_d \cdot V_{dd}}{(V_{dd} - V_{on} - \Delta V_{th})^{\alpha_d}} \cdot \left(\frac{w_{out}}{w_{in}} + \frac{w_{par}}{w_{in}} \right) \\ &= \tau_{ref} \cdot g \cdot \left(h + \frac{p}{g} \right) \\ &= \tau_{ref} \cdot d \end{aligned} \tag{3.1}$$

This is a curve-fitted expression and parameters V_{on} and α_d are intrinsically related, yet not necessarily equal, to the transistor threshold voltage and velocity saturation index. ΔV_{th} is the change from the standard threshold voltage given by technology;

K_d is a fitting parameter; $h = w_{out}/w_{in}$ is the electrical fan-out of a gate; and w_{par}/w_{in} is a measure of its intrinsic delay. Gate and parasitic capacitances are both linearized and the effects of transistor capacitance non-linearities are lumped into the fitting parameters V_{on}, α_d, and K_d.

The simple delay model fits SPICE simulated data within 5% over a range of supply voltages from $0.4V_{dd}^{max}$ to V_{dd}^{max} and fanout factors from 2 to 10, assuming equal input and output rise and fall times [18]. Using the linear delay model from the method of logical effort [21], the delay formula can be expressed simply as a product of the process-dependent time constant τ_{ref} and the unit-less delay d. This delay d consists of the intrinsic delay p due to the self-loading of the gate, and the fanout delay $g \cdot h$ which is the product of the logical effort g and the fanout h. The logical effort g represents the relative ability of a gate to deliver current for a given input capacitance. The fanout h is the ratio of the total output to input capacitance. The simple linear delay model naturally extends to logic paths and multiple-supply voltages [25]. The delay of a logic path is simply $D = d_{path} \cdot \tau_{ref}$, where d_{path} is the sum of the normalized gate delays along the path.[1]

When there are multiple supply voltages, the perception of a "process constant" to which delays are normalized changes. A gate that operates at a reduced supply voltage has a smaller device current for the same input capacitance. Thus, if V_{dd} is scaled down, the logical effort and the parasitic delay increase, as modeled by

[1]Small letters label gate parameters, capital letters label circuit and system parameters

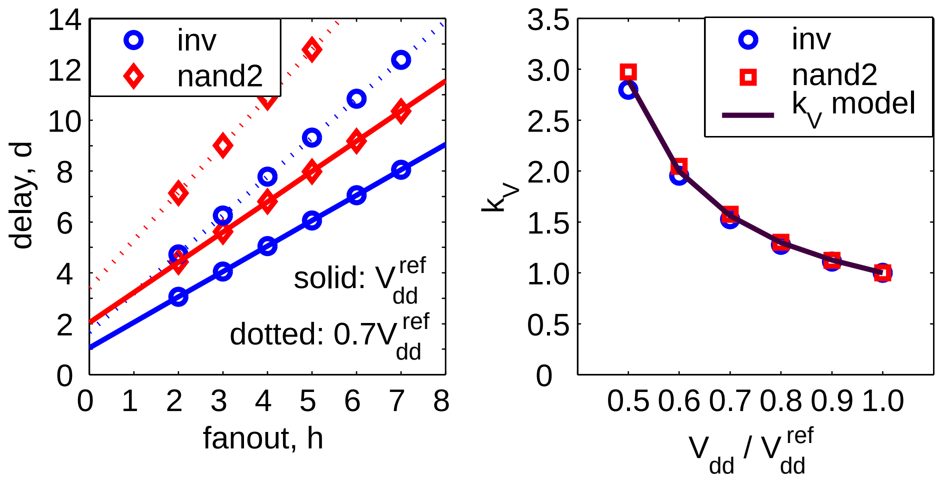


Figure 3.1: Delay vs. fanout and supply.

the voltage-dependent factor k_v in Eq. (3.2) and demonstrated in Fig. 3.1. When a regular gate is placed at the interface between the high and low supply domains, the pull-up path operates at reduced supply, thus with a higher logical effort. The size of the pull-up must increase in order to equalize the logical effort of each path. This is modeled through voltage and gate topology dependent scaling factors.

$$k_v = \frac{V_{dd}^{low}}{V_{dd}^{ref}} \cdot \left(\frac{V_{dd}^{ref} - V_{on}}{V_{dd}^{low} - V_{on}} \right)^{\alpha_d} \tag{3.2}$$

3.1.2 Energy Model

Switching and leakage energy are considered as the two dominant components in this analysis. The analysis based on the gate-level model is expanded to data-path logic, taking into account the impact of switching activity and state-dependent leakage.

Switching Energy

The switching component is the standard dynamic energy term given by Eq. (3.3),

$$e_{Sw} = \alpha \cdot K_e \cdot (w_{out} + w_{par}) \cdot V_{dd}^2 \tag{3.3}$$

where $K_e \cdot w_{out}$ is the load capacitance, $K_e \cdot w_{par}$ is the self-loading of the gate, and α is the probability of an energy-consuming transition at the output of the gate. The main optimization parameters are supply voltage V_{dd} and capacitances which are directly proportional to gate size.

Leakage Energy

Static leakage of a logic gate at $V_{gs} = 0$ is modeled as:

$$e_{Lk} = D \cdot w_{in} \cdot I_0(S_{in}) \cdot \Big(V_{th}^{ref} + \Delta V_{th} - \gamma \cdot V_{dd}\Big) \cdot V_{dd} \tag{3.4}$$

where $D = d_{critical-path} \cdot \tau_{ref}$ is the cycle time, $I_0(S_{in})$ is the normalized leakage current of the gate with inputs in state S_{in}, V_{th}^{ref} is the standard threshold voltage provided by the technology, and $V_0 = n \cdot kT/q$ and γ account for the sub-threshold slope and DIBL factor, respectively. The model in Eq. (3.4) is calibrated in HSPICE over the full range (defined by lower and upper limits) of design parameters V_{dd}, V_{th}, W, and also over the entire set of states S_{in} for each of the gates. The model neglects gate leakage that occurs due to direct tunneling of carriers through a thin gate oxide.

Gate leakage is expected to become a problem with oxides thinner than $2nm$ [28], or below the $90nm$ node. (For reference, the diameter of an atom is about $0.1nm$.) With further shrinking of the gate oxide, gate leakage is predicted to increase at a rate of $500\times$ per technology generation while sub-threshold leakage is predicted to increase $5\times$ for each technology generation [29].

In large circuit blocks, the logic state and the switching probability of the internal gates are obtained through logic simulation. This way, the gate-level models given by Eqs. (3.3), (3.4) are extended to compute the total circuit energy.

3.2 Optimization Approach

Optimization procedure starts by finding the reference point for a design and then minimizing energy subject to a delay constraint normalized to the reference delay.

3.2.1 Reference Point

Reference point for design optimization is the minimum delay D_{min} design obtained through gate size optimization, under the standard supply V_{dd}^{ref} and threshold voltage V_{th}^{ref} for a technology. The minimum delay point is one of the points on the energy-efficient curve and is convenient reference, because it is well defined.

Reference supply voltage corresponds to the maximum supply V_{dd}^{max} specified by the technology reliability limit. In this chapter, the nominal threshold voltage is the low-V_{th} from a dual-V_{th} $0.13\mu m$ technology, and this value is labeled as reference V_{th}^{ref}, corresponding to $\Delta V_{th} = 0$ in the model, Eqs. (3.1), (3.3), (3.4). In this particular $0.13\mu m$ technology, for example, $V_{on}^{ref} = 0.34V$, $V_{dd}^{max} = 1.2V$. The minimum delay D_{min} is achieved for some specified output load C_L and a fixed input capacitance C_{in}. All capacitances are normalized to the input capacitance of a unit inverter.

3.2.2 Optimization Procedure

In the optimization procedure, some percentage incremental change in delay, d_{inc}, relative to the reference point is specified. The energy is minimized for the new target delay $D = D_{min}(1 + d_{inc}/100)$, by using supply voltage, threshold voltage, gate size, and optional buffering as optimization variables. Supply optimizations investigated include global supply reduction, two discrete supplies, and per-stage supplies. In the discrete supply optimizations, supply voltage can only decrease from input to output of a block assuming that low-to-high level conversion is done in registers, while sizing is allowed to change continuously.

3.3 Sensitivity Analysis

In energy-delay optimization, the objective is to utilize available timing slack for maximal energy reduction. There are usually several tuning variables that can be used to trade timing slack for energy at various levels in the design hierarchy. As pointed out in Chapter 2, the energy-efficient design is achieved when the marginal costs of all the tuning variables are balanced. Each design variable x carries a certain energy reduction potential per delay cost, at each point in the energy-delay space, as given by Eq. (3.5).

$$S_x(X) = \left.\frac{\partial E/\partial x}{\partial D/\partial x}\right|_{x=X} \tag{3.5}$$

Analysis using relative sensitivity, which was termed hardware intensity, was proposed in [30]. The relative sensitivity simply represents percent power per percent performance for an energy-efficient design as given by Eq. (3.6). However, formulation in Eq. (3.5) is preferred since relative contributions of energy and delay from various blocks are unknown and also may change during the optimization.

$$\eta_x(X) = S_x(X) \cdot \frac{D}{E} \tag{3.6}$$

The true power minimization method always exploits the tuning variable with the largest capability for energy reduction. This ultimately leads to the point where the energy reduction potentials of all tuning variables are equalized. In order to further develop the understanding of these relative gradients, practical expressions (sensitivities) are derived for different tuning variables. The variables considered include gate size W, supply voltage V_{dd}, and change in threshold voltage ΔV_{th} as knobs in the optimization. By analyzing sensitivities, the efficiency of W, V_{dd}, and ΔV_{th} optimizations can be estimated from the energy profile of the logic block. Further, understanding the relationship between the logic block topology and the energy profile is necessary in order to identify the most efficient tuning variables without an exhaustive search.

3.3.1 Sensitivity to Gate Size, W

The sensitivity of circuit energy $E = E_{Sw} + E_{Lk}$ to delay due to a change in size of a gate in stage i is given by Eqs. (3.7) and (3.8), where $ec_i = \alpha \cdot K_e \cdot (w_{in,i} + w_{par,i}) \cdot V_{dd}^2$ represents the switching energy due to capacitances of stage i (this is not the energy consumed at the output of stage i), $e_{Lk,i}$ is the leakage energy of stage i as defined by Eq. (3.4), E_{Sw} and E_{Lk} are the total switching and the total leakage energy, respectively. Parameter $h_{eff,i} = g_i \cdot h_i$ is the effective fanout of stage i.

$$\frac{\partial E_{Sw}/\partial w_i}{\partial D/\partial w_i} = -\frac{ec_i}{\tau_{ref} \cdot (h_{eff,i} - h_{eff,i-1})} \tag{3.7}$$

$$\frac{\partial E_{Lk}/\partial w_i}{\partial D/\partial w_i} = \frac{E_{Lk}}{D} - \frac{D \cdot e_{Lk,i}}{\tau_{ref} \cdot (h_{eff,i} - h_{eff,i-1})} \tag{3.8}$$

Equations (3.7), (3.8) show that the largest potential for energy savings occurs at the point where the design is sized for minimum delay with equal effective fanouts h_{eff}, resulting in infinite sensitivity. Intuitively, the delay cannot be reduced beyond the minimum achievable delay, regardless of how much energy is spent. While decreasing gate size decreases the leakage current, it also increases the cycle time D, which increases the leakage energy. At the point where the sensitivity in Eq. (3.8) becomes positive, the leakage energy will start increasing with further gate size reduction due to longer cycle time. In order to achieve equal sensitivity in all stages, the difference in the effective fanouts must increase in proportion to the energy of the gate, which closely ties the circuit energy profile with optimal sizing [25]. For example, this matches with the variable taper result of Ma and Franzon [18] for energy minimization of a delay constrained inverter chain.

3.3.2 Sensitivity to Supply Voltage, Vdd

The sensitivity of total circuit energy to delay increase due to global supply reduction is given by Eqs. (3.9), (3.10). Similar to the sizing approach, the design sized for the minimum delay at maximal supply voltage offers the greatest potential for energy reduction. This potential diminishes with the reduction in supply voltage since the energy $E = E_{Sw} + E_{Lk}$ decreases, cycle time D increases, and the V_{on}/V_{dd} ratio increases. Supply voltage reduction has a two-fold impact on the leakage energy in Eq. (3.10): the leakage energy E_{Lk} increases because of increase in cycle time D, but it also decreases because of the supply reduction on leakage power. The resulting tendency is a decrease in the leakage energy with supply reduction, which results in negative sensitivity of the leakage energy to delay.

In the sub-threshold regime, however, leakage energy will start to increase with further supply voltage reduction. At some point the effect of supply reduction on leakage power saturates, while the delay increases rapidly. This results in an increase in the leakage energy. Since the switching energy also saturates with the saturation of voltage, a minimum energy point exists in the sub-threshold regime [31].

$$\frac{\partial E_{Sw}/\partial V_{dd}}{\partial D/\partial V_{dd}} = -\frac{2E_{Sw}\cdot(1-V_{on}/V_{dd})}{D\cdot(\alpha_d - 1 + V_{on}/V_{dd})} \tag{3.9}$$

$$\frac{\partial E_{Lk}/\partial V_{dd}}{\partial D/\partial V_{dd}} = -\frac{E_{Lk}}{D}\cdot\left(\frac{(1-V_{on}/V_{dd})\cdot(1+\gamma\cdot V_{dd}/V_0)}{\alpha_d - 1 + V_{on}/V_{dd}} - 1\right) \tag{3.10}$$

In dual-supply voltage optimization, the same formula holds, where E_{Sw}, E_{Lk}, and D represent the total switching energy, the total leakage energy, and the delay of the stages under the reduced supply voltage, respectively.

3.3.3 Sensitivity to Threshold Voltage, Vth

The sensitivity of energy to delay due to the change in threshold voltage is given by Eq. (3.11). In the proposed energy model, switching energy is not affected by the change in threshold voltage. The energy-delay sensitivity decays exponentially with increasing ΔV_{th} because E_{Lk} is an exponential function of ΔV_{th}, as in Eq. (3.4).

$$\frac{\partial E_{Lk}/\partial(\Delta V_{th})}{\partial D/\partial(\Delta V_{th})} = -\frac{E_{Lk}}{D}\cdot\left(\frac{V_{dd} - V_{on} - \Delta V_{th}}{\alpha_d\cdot V_0} - 1\right) \tag{3.11}$$

The exponential dependence of the leakage energy on ΔV_{th} limits the optimization range. Lowering ΔV_{th} while maintaining circuit speed for designs with very low leakage allows for a reduced V_{dd} and therefore reduced switching energy. The total energy is minimized when the leakage and switching components are comparable [32].

3.4 Optimization Examples

In re-examining circuit examples representing common topologies, it can be seen that they differ in the amount of off-path loading and path reconvergence. By analyzing how these parameters affect a circuit energy profile, general principles in energy reduction relating to logic blocks can be better defined. This is illustrated using an inverter chain, a memory decoder, and an adder as examples.

3.4.1 Inverter Chain

Inverter chain is the example of a single path topology without branching. At the reference point, energy profile grows geometrically toward the output.

Gate Size Optimization

The use of gate sizing to minimize the energy of a fixed length inverter chain is shown in Fig. 3.2. Initially, when the circuit is sized for minimum delay, all stages have the same delay. Due to the geometric progression in size, most of the energy is dissipated in the last few stages, with the largest energy stored in the final load. Starting from the minimal delay point where all of the sensitivities are infinite, the gate sizes along the chain are adjusted so that all of the sensitivities decrease equally. This, in turn, leads to an increase in effective fanout toward the output where most of the energy

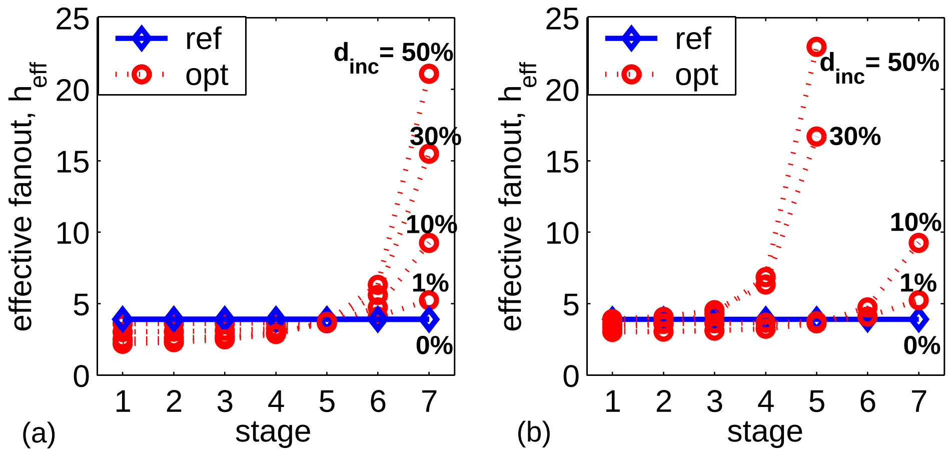


Figure 3.2: Delay profile in an inverter chain after sizing optimization: a) fixed, b) variable number of stages.

is consumed, as shown in Eq. (3.7). Therefore, the biggest energy savings for a fixed delay increase are achieved by downsizing the largest gates in the chain first. The optimal size of stage i is derived in Eq. (3.12). The expression is similar to that in [18] and directly follows from Eq. (3.7).

$$W_i = \frac{W_i^{ref}}{\sqrt{1 + \frac{ec_{i-1}}{S_W \cdot \tau_{ref}}}} \tag{3.12}$$

In the above formula, W_i^{ref} is the size of stage i that results in the minimum delay of the chain [8]. In an energy-efficient design, the sizing sensitivity of all stages S_W is equal and also a function of the delay constraint.

If the number of stages can be varied, the delay constraint may be met with a fewer number of stages leading to a greater energy reduction. Intuitively, as the final stage is downsized to gain the biggest energy savings for a given delay increase, the size and number of the remaining stages adjust to meet the delay constraint, Fig. 3.2(b) [25]. It is important to realize that, due to geometric progression in size in an inverter chain, most of the energy is consumed in driving the fixed final load, and the maximum energy saving from sizing is limited to about 30%.

Supply Voltage Optimization

Unlike sizing, scaling the supply directly affects the energy needed to charge the final load capacitance, and therefore can have a larger effect on the total energy. For illustration, supply optimization on a per-stage basis in the inverter chain is shown. The assumption is that the supply voltage can only decrease from the input toward the output to avoid level conversion inside the block. In the nominal case, in which the delay of each stage is equal, the supply sensitivity of each stage depends only on the energy of that stage, as indicated in Eq. (3.9). As in sizing, supply voltage optimization adds incremental delays, first to the stages with the highest energy consumption (stages toward the end of the chain), while increasing the effective fanout of these stages by lowering their supply voltage. Figure 3.3 shows the optimized per-stage supply and the resulting effective fanout.

Compared to sizing, the supply optimization requires less change in the effective fanout for the same energy reduction. In practical designs, the effective fanout of the gate is bounded by the signal slope constraints to around 10-15. This is also considered in the optimization.

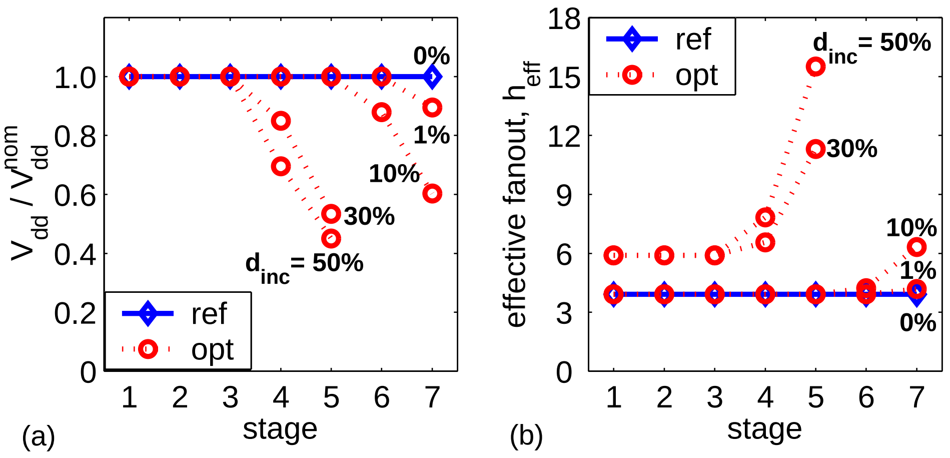


Figure 3.3: Per-stage supply optimization of an inverter chain with variable number of stages: a) optimal stage supply, b) delay profile.

Comparison: Gate Size vs. Supply Voltage Optimization

Figure 3.4 is the result of various optimizations performed on the inverter chain: sizing (cW), global V_{dd} (gV_{dd}), two discrete supplies ($2V_{dd}$), and per-stage V_{dd} (cV_{dd}). These graphs show energy reduction and sensitivity versus delay increment. The key concept to realize is that the parameter with the largest sensitivity has the largest potential for energy reduction. For example, at small delay increments sizing has the largest sensitivity, so it offers the largest energy reduction, but the potential for energy reduction from sizing quickly falls off. At large delay increments, it pays to scale the supply voltage of the entire circuit, achieving the sensitivity equal to that of sizing at around 25% excess delay.

The results in Fig. 3.4(a) also suggest that dual-V_{dd} closely approximates the optimal per-stage supply reduction, meaning that there is almost no additional benefit of having more than two discrete supplies for improving energy in this topology.

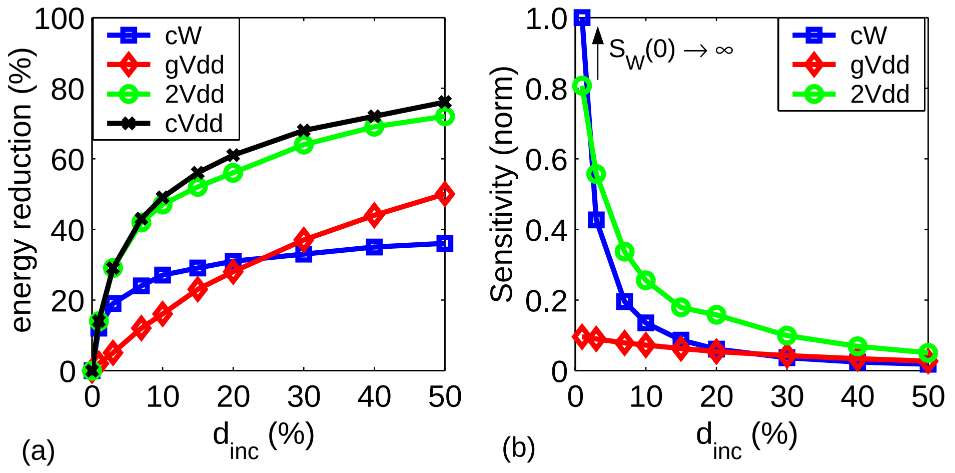


Figure 3.4: Sizing and supply optimization of an inverter chain: a) energy reduction vs. delay increment, b) energy-delay sensitivity.

An inverter chain has a particularly simple energy distribution, one which increases geometrically until the final stage. This type of profile drives the optimization over sizing and supply to focus on the final stages first. However, most practical circuits have a more complex energy profile. An example of such a circuit is a memory decoder.

3.4.2 Memory Decoder

The decoder shares some characteristics with a simple inverter chain; the total capacitance at each stage grows geometrically, but the number of active paths decreases geometrically, as well. As a result of this, the peak of the energy distribution is often in the middle of the structure. For example, the 256 wordline SRAM decoder shown in Fig. 3.5 has the energy peak at the output of the predecoder because of the path properties shown in Table 3.1.

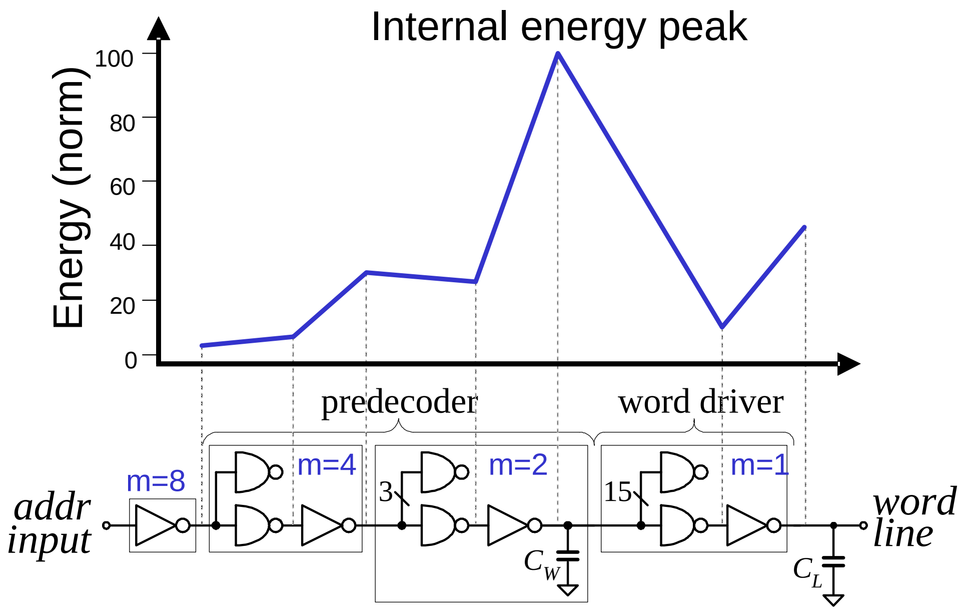


Figure 3.5: Critical path of 8/256 wordline SRAM decoder. Energy profile is also shown.

Table 3.1: Activity map of 8/256 wordline SRAM decoder.

Decoder gates	predriver	predecoder		word driver
	Inv	Nand-Inv	Nand-Inv	Nand-Inv-Buf
Active	16	4	2	1
Total	16	16	32	256

Figure 3.5 shows the critical path of this SRAM decoder. The multiplication factor m denotes the number of active gates at each stage. Branching occurs at the input of each NAND gate and the number of active gates per stage decreases in a geometric fashion to select only one wordline at the output. In addition to a large branching factor at the output of the predecoder, there is an extra capacitance C_W from the wire spanning word drivers. The resulting energy peak, on a per-stage basis, is thus at the output of the predecoder.

Gate Size Optimization

Gate size optimization effectively reduces the internal energy peaks through direct gate sizing or buffer insertion, as shown in Fig. 3.6. The initial sizing for minimum delay does not require an extra buffer at the output of the decoder, thus the total number of stages is seven. Inserting a buffer stage at the output reduces the effective load presented by the 256 decoder/word driver cells. Alternatively, optimization by direct gate sizing minimizes the size of the word driver input and produces the same effect, as shown in Fig. 3.6(b). This essentially divides the sizing problem into two sub-problems: a) sizing of the predecoder logic to drive the minimum word driver input, and b) sizing of the word driver to drive the wordline. This is readily seen from the per-stage sensitivity expression with branching:

$$\frac{\partial E_{Sw}/\partial w_i}{\partial D/\partial w_i} = -\frac{b_{i-1} \cdot ec_i}{\tau_{ref} \cdot (h_{eff,i} - h_{eff,i-1})} \tag{3.13}$$

in which b_{i-1} is the branching factor of stage $i-1$. Downsizing the gates driven by the stage with the highest branching factor yields the biggest energy savings for the given delay cost. In the decoder example this situation occurs at the output of the predecoder, as shown in Fig. 3.6. While the peak of switching energy is inside the block, the peak of leakage energy occurs at the output, due to the activity profile of the decoder. This is illustrated in Fig. 3.7, where the energy profile in a min-delay sized decoder is shown for cases with seven and nine stages. Output gates are the

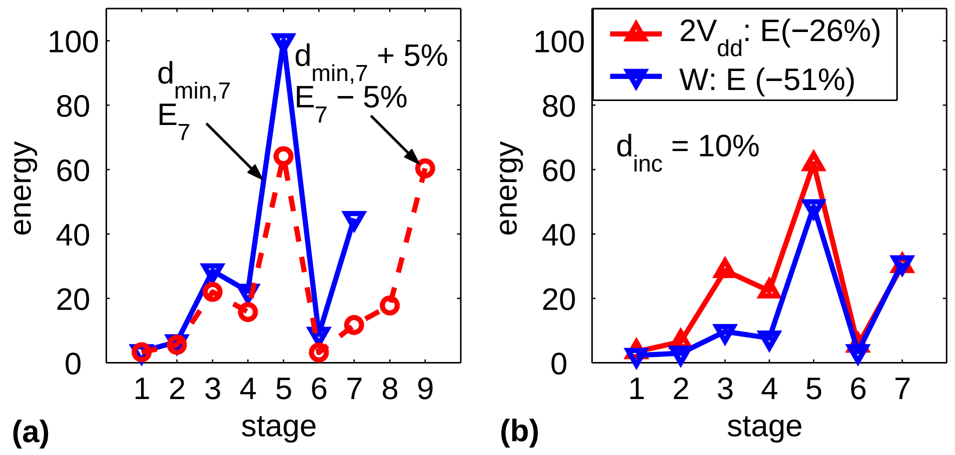


Figure 3.6: Energy profile in SRAM decoder: a) reference design, b) design with 10% incremental delay. ($W_L = 128$, $\alpha = 15\%$)

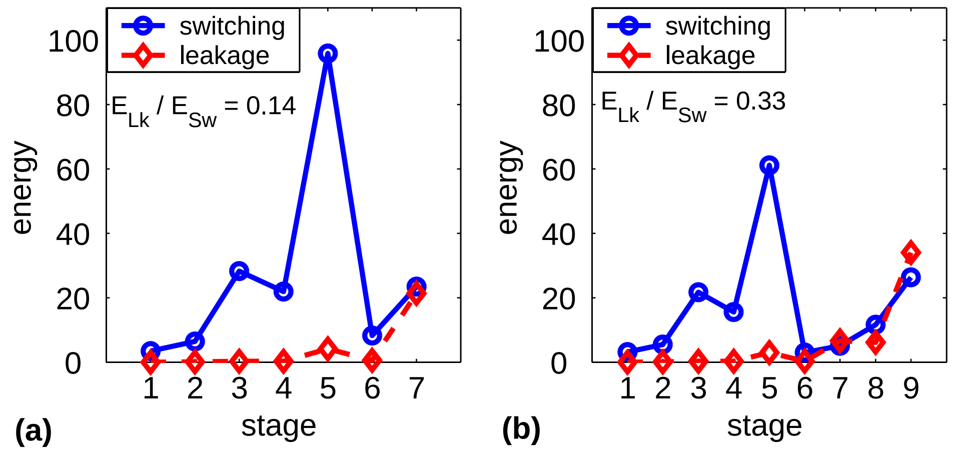


Figure 3.7: Energy profile in SRAM decoder with: a) 7 stages, b) 9 stages. Leakage energy increases with the number of inactive gates. ($d = d_{min}$, $W_L = 128$, $\alpha = 15\%$)

largest and the majority of the gates are inactive, resulting in the largest leakage at the output. Although inserting a buffer stage reduces the size of the predecoder, the leakage energy of the word driver increases relative to the switching energy as shown in Fig. 3.7, because only one output buffer is active at a time.

Supply Voltage Optimization

The supply optimization is less effective in designs where the peak of energy consumption occurs inside the block. Because of the assumption that the supply voltage can only decrease from input to output, the delay of all stages after the peak needs to increase in order for the supply to affect the energy peak. This reduces the marginal return, as shown in Fig. 3.6(b). Sizing optimization has more marginal return than discrete supply optimization, because sizing can selectively reduce dominant energy peaks inside the block at the expense of increased delay in stages immediately after the energy peak. On the contrary, the supply optimization starts from the output of the block and works backwards.

3.4.3 Tree Adder

More complex designs may have reconvergent fanouts and multiple active outputs generated by paths with varying logic depths. The circuit energy profile is crucial to choose the tuning variable that is most effective in reducing the total energy of the circuit, as illustrated on a 64-bit Kogge-Stone carry-lookahead tree adder [33]. For

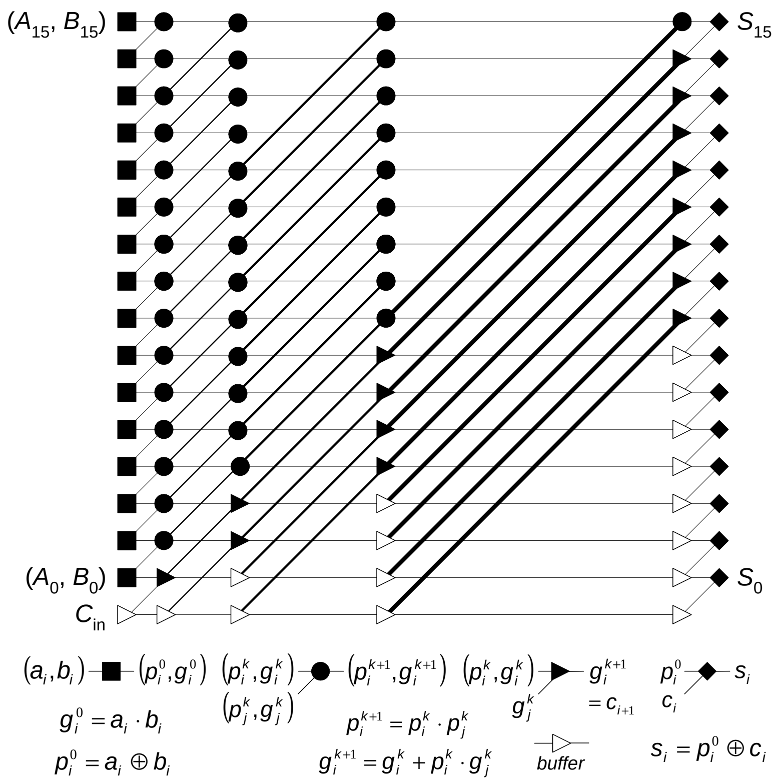
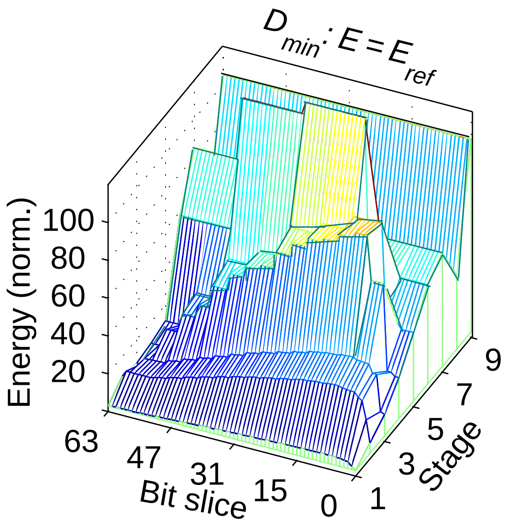

(A_{15}, B_{15})
S_{15}
(A_0, B_0)
S_0
C_{in}
(a_i,b_i) (p_i^0,g_i^0)
$g_i^0 = a_i \cdot b_i$
$p_i^0 = a_i \oplus b_i$
(p_i^k,g_i^k) (p_j^k,g_j^k) (p_i^{k+1},g_i^{k+1})
$p_i^{k+1} = p_i^k \cdot p_j^k$
$g_i^{k+1} = g_i^k + p_i^k \cdot g_j^k$
(p_i^k,g_i^k) g_j^k g_i^{k+1} $= c_{i+1}$
buffer
p_i^0 c_i s_i
$s_i = p_i^0 \oplus c_i$

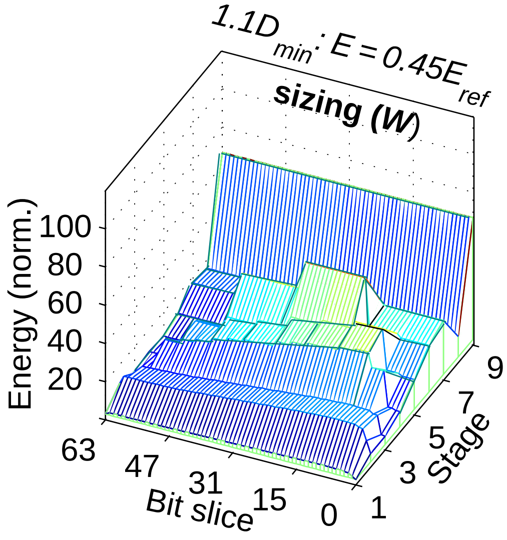
Figure 3.8: Schematic of a 16-bit Kogge-Stone tree adder.

brevity, Fig. 3.8 shows a 16-bit tree as an example. Various symbols in the figure correspond to different logic operations [34], as indicated in the figure. Dot operators compute propagate and generate signals in a parallel-prefix tree. Significant features of this adder topology include reconvergent fanouts inside propagate-generate blocks, long wires, and multiple active outputs.

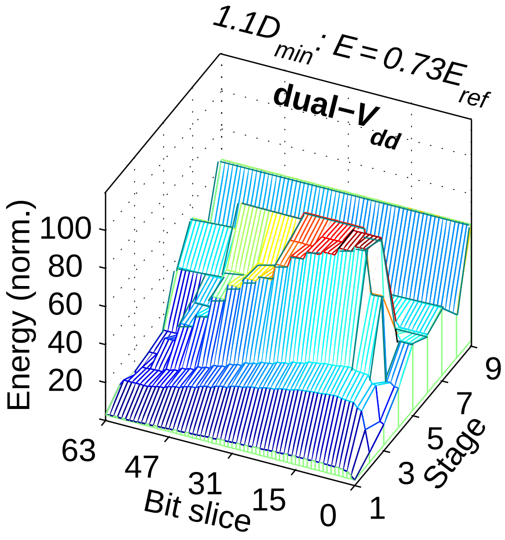
The initial sizing of the reference adder attempts to make all the paths in the adder equal to the critical path for a fair comparison. Each gate in the adder is allocated to a bit slice, which is the natural partitioning for tree adders. Figure 3.9 shows the resulting energy map for minimum delay, as well as the case when a 10%



(a)



(b)



(c)

Figure 3.9: Energy distribution in a 64-b adder: a) reference design, b) design with 10% additional delay after optimal sizing, c) design with 10% additional delay, after dual-V_{dd} optimization. Each sum output is loaded with $C_L = 32$, input data activity is 10%.

delay increase is allowed. In this type of adder, the switching activity of propagate logic diminishes rapidly with the number of stages, and most of the switching energy is consumed by the generate logic in the later stages. The internal energy peak in Fig. 3.9(a) occurs due to the large activity of the propagate logic that is comparable to that of generate logic close to the input of the adder, and also due to the large load by the gates which drive long wires in the final stages of the adder.

Comparison: Gate Size vs. Supply Voltage Optimization

The adder energy map of Fig. 3.9(b) shows that the gate size optimization is very effective in circuit topologies in which energy peaks occur inside the block. This statement holds as long as gate sizes do not reach their limits, which happens for small incremental delays, up to about 20%. For a 10% excess delay, a 55% decrease in energy is possible using transistor sizing under V_{dd}^{max}, while only 27% is saved by using two supplies without resizing, as shown in Fig. 3.9(c). Reducing the supply over the whole block yields even lower energy reduction at only 17%. Using multiple supplies is therefore less effective than sizing in designs where the peak of energy consumption occurs inside the block. In order for the supply optimization to affect the energy peak, the delay of all stages following the peak needs to increase, thus reducing the marginal return. On the other hand, sizing can selectively target energy peaks, by focusing on downsizing of the selected internal gates first, yielding higher energy returns than the discrete supply optimization.

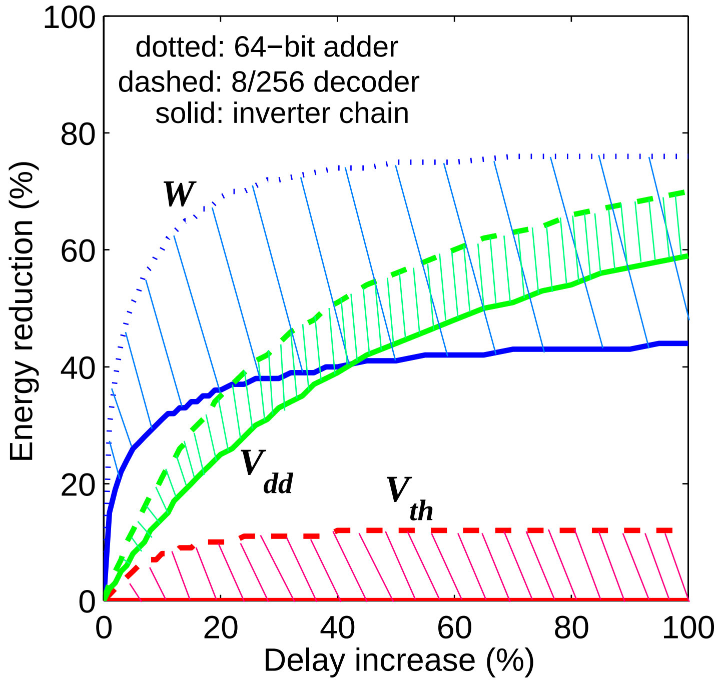


Figure 3.10: Energy reduction due to W, V_{dd} and V_{th} in inverter chain, 8/256 memory decoder, and 64-bit adder. Only cases with min and max energy reduction are shown.

3.4.4 Takeaway Points

The analysis of energy reduction potentials in the three topologically different circuit examples reveals some general conclusions. Plot of energy reduction in Fig. 3.10 shows regions of the potential energy reduction for an inverter chain, memory decoder, and an adder due to W, V_{dd}, and ΔV_{th}. The results provides some insight into which parameters are more effective in different regions of delay.

- Sizing optimization is the most effective at small incremental delays. Sensitivity to sizing is infinite at the reference point, but the benefits of sizing get quickly utilized at about 20% excess delay. This can be observed from Fig. 3.11 which

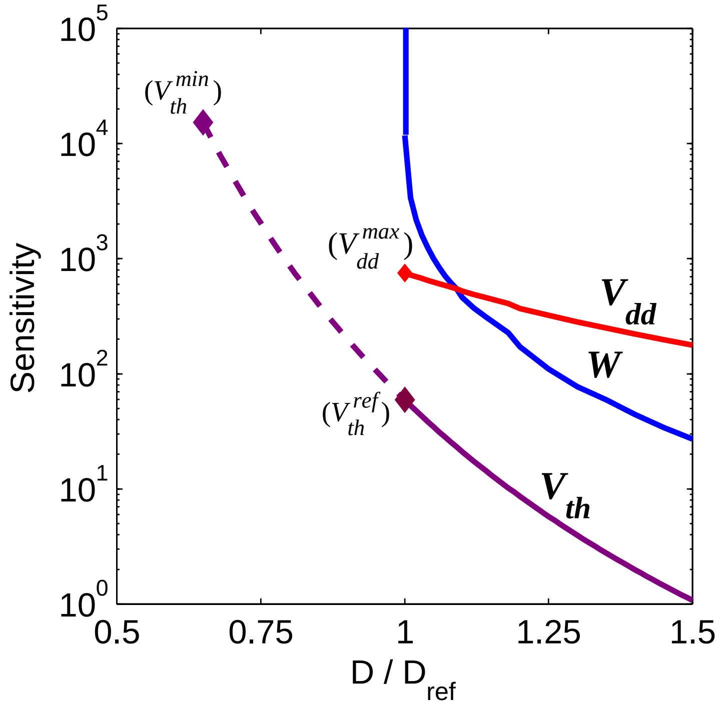


Figure 3.11: Sensitivity to V_{dd}, V_{th}, and W in the adder example. V_{dd}^{max} and V_{th}^{min} are limits on V_{dd} and V_{th}.

plots the energy-delay sensitivity to tuning variables in the adder example.

- At larger delays, sensitivity to sizing diminishes and supply voltage becomes more effective for energy savings.

- The threshold voltage primarily affects leakage energy which is significant in designs with lots of inactive gates, such as memory decoders [26]. In fact, V_{th}^{ref} is too high in most circuits, which could be exploited advantageously to reduce V_{th} to speed up the circuit even beyond the reference delay. This in turn creates the opportunity for other variables such as V_{dd} or W to exploit the timing slack for overall energy reduction.

- The sensitivity gap between variables can be exploited to effectively perform multi-variable optimization, leading to the most energy-efficient solution.

3.4.5 Multi-Variable Optimization

The highest energy-efficiency is achieved by balancing energy reduction potentials of all tuning variables, otherwise one would tune the variable with low energy cost rather than the variable with high energy cost.

The simultaneous use of gate size, supply and threshold voltage is investigated for the adder example. Figure 3.12 shows the optimal energy-delay trade-off curve obtained by jointly optimizing gate size, supply and threshold voltage. Energy and delay are normalized to the reference design (D_{ref}, E_{ref}). As seen in the plot, there is still significant room for improvement for the reference design, because the sensitivities differ at that point. In the reference design, the sizing sensitivity is infinite, the supply sensitivity is fifty percent higher, and the threshold sensitivity is five times smaller than the sensitivity at the optimal point (D_{ref}, E_{min}), which is used as the baseline case in Fig. 3.12.

After balancing the sensitivities by downsizing the gates and decreasing supply and threshold, about 65% of energy is saved without any delay penalty. This is illustrated in Fig. 3.12, where the reference design (D_{ref}, E_{ref}) moves down on the y-axis to the optimal design point (D_{ref}, E_{min}) on the energy-efficient curve. Alternatively, speedup of about 25% can be achieved while maintaining the same level of energy.

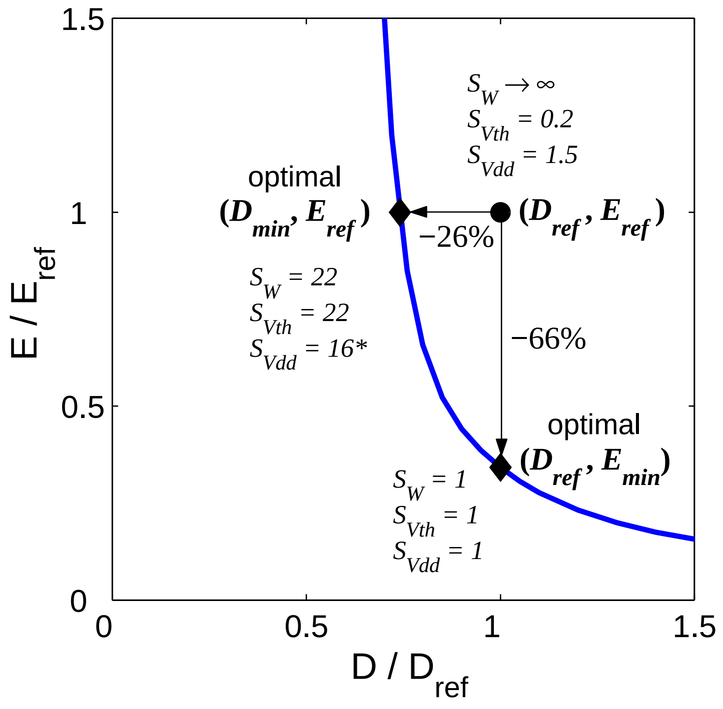


Figure 3.12: Optimal energy-delay trade-off in a 64-bit adder after performing $V_{dd} - V_{th} - W$ optimization. Reference is the design sized for minimum delay under V_{dd}^{max} and V_{th}^{ref}. Sensitivity to each of the tuning variables is marked on the graph.

Although (D_{min}, E_{ref}) is still on the energy-efficient curve, the data in Fig. 3.12 shows that the sensitivities are not the same in this case, because V_{dd} reached its upper limit, V_{dd}^{max}, resulting in a constrained optimum.

Typically, only a subset of tuning variables is selected for optimization. A general heuristic for the overall design optimization is as follows: 1) choose two variables with extreme sensitivities (i.e. variable with the lowest and variable with the highest sensitivity) and balance sensitivity to these two variables, 2) repeat the procedure until all variables are balanced. In the adder case, for delays close to D_{ref}, for example, sizing and threshold voltage should be balanced first since there is the largest gap between the sizing and threshold sensitivities around the nominal delay point, as illustrated in Fig. 3.11. This way, the designer can obtain nearly optimal energy-delay trade-off.

Although energy savings at the circuit-level are quite significant, the performance range of effective circuit-level optimization is very narrow. The data in Fig. 3.12 shows that circuit optimization is really effective only in the region of about ±30 percent around the reference delay, D_{ref}. Outside of this region, optimization becomes costly either in terms of delay or energy, and a more efficient variable must to be introduced at another level in the design hierarchy. This naturally expands the optimization to the micro-architectural level.

Chapter 4

Micro-Architectural Optimization

Energy savings of about 65% at the circuit level in the adder example are possible without any delay penalty by simply choosing appropriate values of supply, threshold, and circuit size. However, individual circuit examples may be misleading. For example, if the energy of the adder, or some other functional-unit block, is a much smaller fraction of the total processor energy than that of registers and clocking, then it might be more beneficial to lower the power of the registers (making the latches slower) and increase the power of the adder (making the adder faster).

This chapter is focused on micro-architectural optimization demonstrating that the scope of energy-efficient optimization can be extended by the choice of circuit topology, the level of parallelism and/or time-multiplexing. Since parallel and time-multiplexed solutions significantly affect the area of their respective designs, the overall design cost is minimized when optimal energy-area trade-off is achieved.

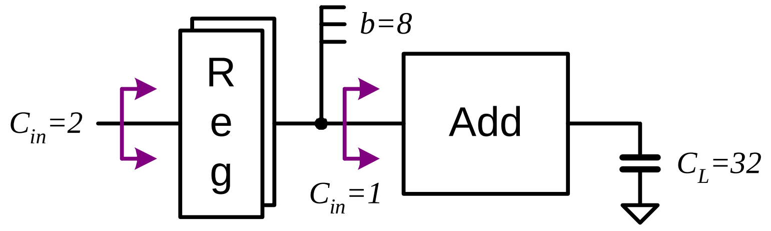


Figure 4.1: Simplified model of one bit-slice of a 64-bit ALU.

4.1 Choosing Optimal Circuit Topology

The optimization of a pipeline that jointly optimizes registers and logic is demonstrated using a modular approach. When cascading heterogeneous circuit blocks, such as registers and logic, the total available delay has to be optimally divided among the circuit blocks to achieve minimal energy. As an example, an ALU shown in Fig. 4.1 is analyzed. The ALU consists of two registers that drive a 64-bit Kogge-Stone tree adder. Each register can be built with simple cycle-latches (CL) [35] or static master-slave latch-pairs (SMS) [36] shown in Fig. 4.2. The output load C_L is due to registers, wire, and bus capacitances; term b is the branching effort [37] at the output of the registers.

The input capacitance of the adder $C_{in}(Add) = 1$ is fixed in order to reduce search space in the global optimization. Without a fixed C_{in} constraint, the optimal $C_{in}(Add)$ would be larger than minimum only in a very narrow range around the minimum delay. For delays farther away from the minimum delay, $C_{in}(Add)$ would quickly reach the lower limit due to a large branching at the output of the register. Therefore, fixing $C_{in}(Add)$ is a good heuristic which also allows for a modular design.

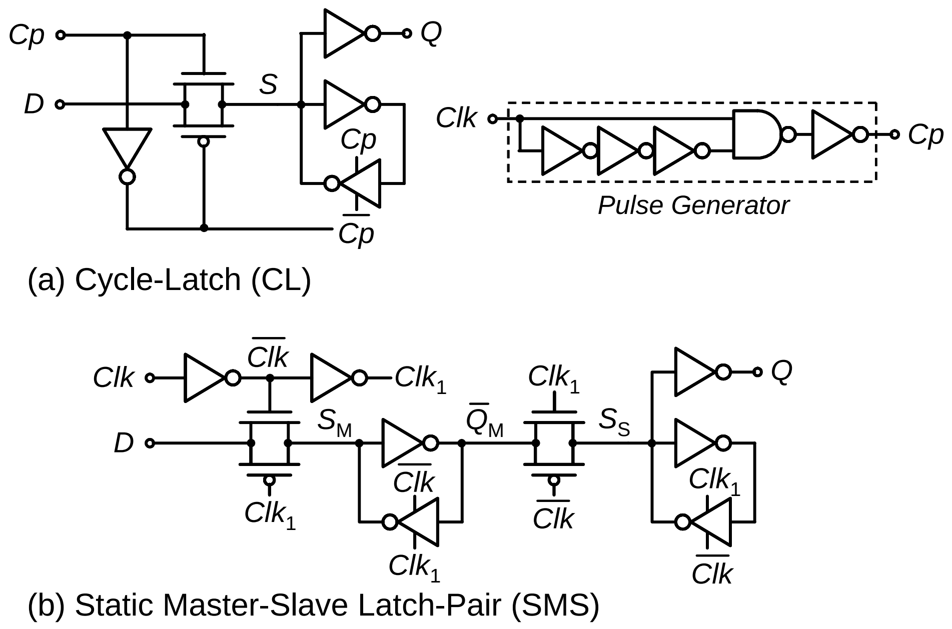


Figure 4.2: Flip-flops used in implementation of the ALU register in Fig. 4.1: (a) high-performance cycle-latch (CL), (b) low-energy static master-slave latch-pair (SMS).

The register and adder significantly differ in their switching activity, which results in a lower initial E_{Lk}/E_{Sw} ratio in the registers than in the adder. The register has higher average activity primarily due to a large activity factor of the clocked nodes. For this reason, the optimal value of V_{dd} and V_{th} tends to be lower in the registers than in the adder. In reality, however, designs are usually constrained to one core-level V_{dd} and V_{th}, making them global variables. Hence extra effort will be spent in sizing the register during the V_{dd}-V_{th}-W optimization of the ALU. The goal is to equalize sensitivities to W for both the *Reg* and *Add* blocks to obtain the most energy-efficient solution. With V_{dd} and V_{th} fixed, sizing the gates inside each of the blocks simply compensates for the intrinsic mismatch in sensitivity due to logic topology and activity.

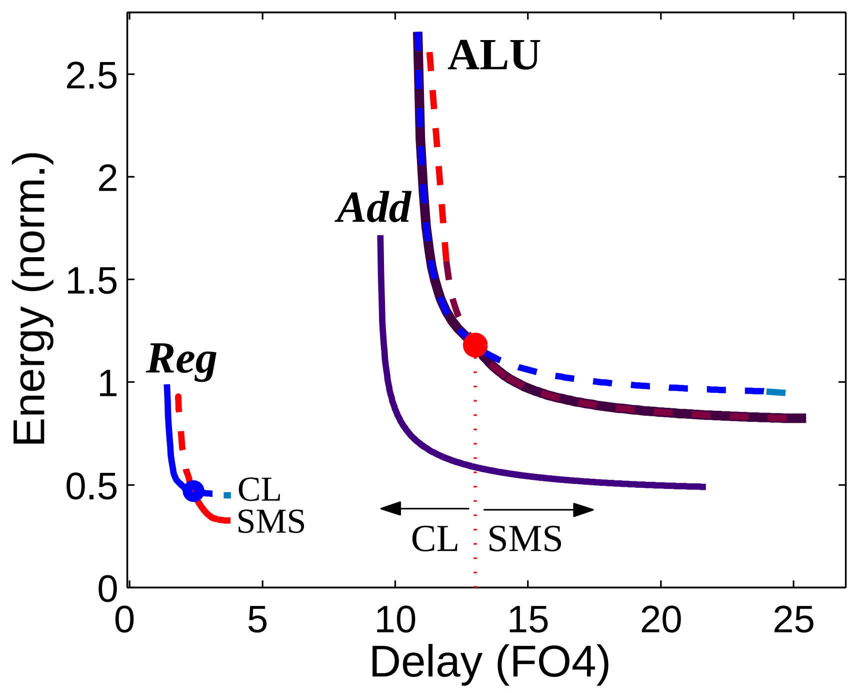


Figure 4.3: Energy-efficient curves in register, adder, and ALU after gate size optimization. The dots indicate transition between CL- and SMS-based register.

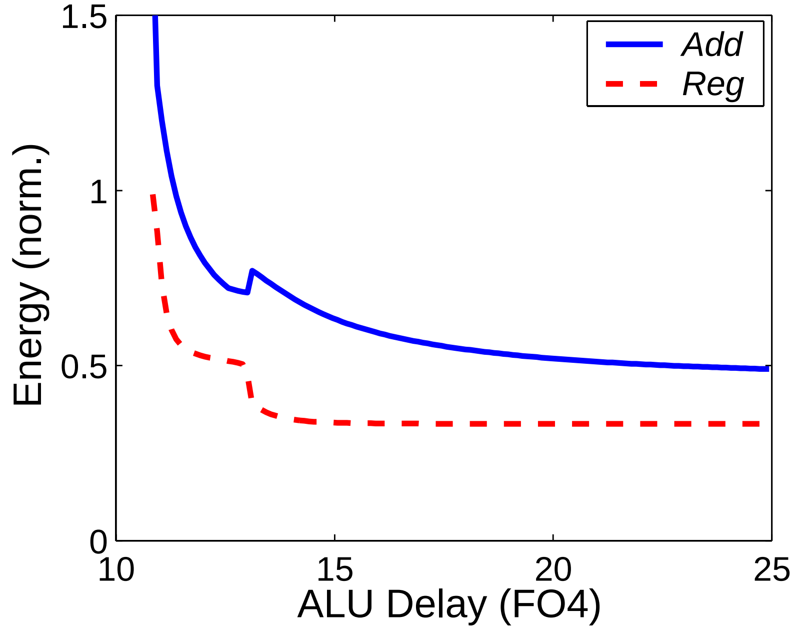
Combining the circuit-level results of the *Reg* and *Add* optimizations, the total energy of the ALU is minimized subject to a cycle time constraint. Figure 4.3 shows the energy of the ALU after optimal sizing under V_{dd}^{max} and V_{th}^{ref}. Dashed lines show sub-optimal designs using an incorrect choice of the register topology. Energy-efficient curves for registers comprised of CL or SMS latches combine to define a composite energy-efficient curve for the *Reg* block, as shown in Fig. 4.3 by the solid line. For each target ALU delay, points from the optimal *Add* and *Reg* curves (solid lines) are chosen to minimize the overall energy of the ALU. The optimal solution confirms that high-performance designs naturally use fast cycle-latches, while simple SMS latch-pairs are suitable for low-energy designs.

The scope of energy-efficient ALU optimization is extended through the selection of the optimal register topology. This is best illustrated through observation of the energy-delay sensitivity in the register and in the adder. The goal of equalizing sensitivities during sizing optimization at the circuit level applies here as well. Thus, the sensitivity of the adder block has to be the same as the sensitivity of the register. Because of the fixed interface between the blocks, the sensitivity of each block simply reduces to the ratio:

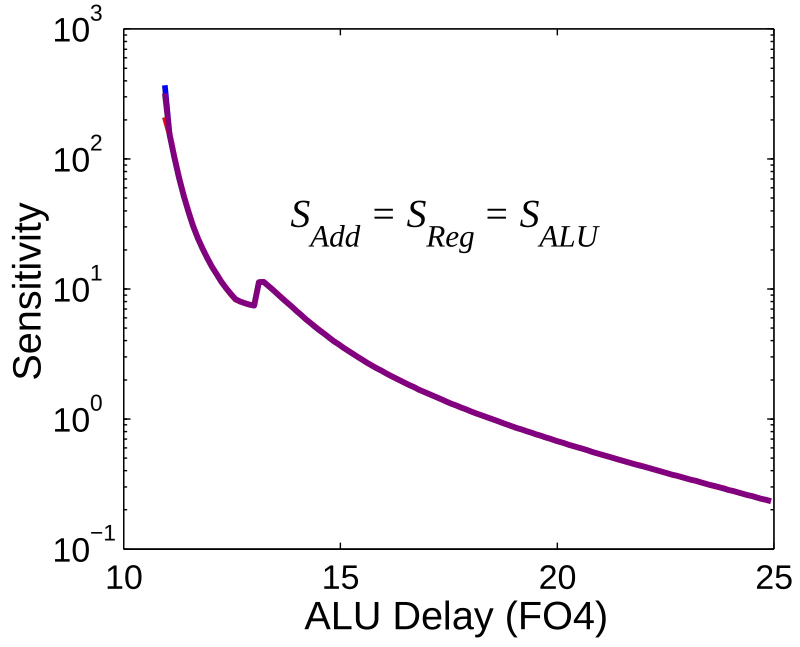
$$S_{block}(W) = \frac{\Delta E_{block}(W)}{\Delta D_{block}(W)} \tag{4.1}$$

where the change in energy ΔE_{block} and the change in delay ΔD_{block} are both functions of the size W. For ALU delays greater than 13FO4 inverter delays, the solution with an SMS-based register becomes more energy-efficient because the benefits from sizing of the CL-based register are utilized. Timing slack created by a faster adder can be exploited in the optimization of the register. This leads to an overall energy reduction of the ALU due to higher register sensitivity to sizing.

The process described above is illustrated in Fig. 4.4(a), which plots energy of the adder and register when they are combined within an ALU. The design is optimal, because the sensitivities of the adder, the register, and the ALU are equal, as illustrated in Fig. 4.4(b). Higher energy-efficiency of the ALU due to a change in register topology means higher sensitivity around the point where the change in register topology occurs (the "twiddle" in the graph), extending the performance range of energy-efficient optimization. Circuit topology and intrinsic node activity have a



(a)



(b)

Figure 4.4: Plots after optimal sizing and change in register topology. (a) Energy of adder and register when they are combined as the ALU, (b) Corresponding sensitivity of register, adder, and ALU.

large effect on the optimization result. At the optimum trade-off point, high-activity gates with large numbers of transistors per stage, such as registers, are downsized, while the slack is consumed by upsized and lower-activity units, such as adders. This agrees with the result from [30], which states that the hardware intensity of the register should be smaller than that of the adder, i.e. the percentage energy per percentage delay in the register is smaller than that in the adder. Therefore, the operating point of the register is pushed out toward longer delays.

4.1.1 Combinational Logic

The discussion so far assumed adder realization using static CMOS gates with continuous sizes. Synthesized designs using discrete standard-cells typically result in about 2× worse performance. This still works well for designs with moderate delay requirements where energy consumption has to be low. Fastest adders are built using dynamic gates [38], resulting in another factor of 2× better delay than a full-custom static CMOS, but consume more energy.

Figure 4.5 shows the optimal energy-delay trade-off for several 64-bit adder implementations. The optimal trade-off curves are obtained through sizing optimization [39] of representative carry look-ahead (CLA) topologies. Solid lines represent actual adder points, while the dashed lines represent just additional available slack. In between domino and static design styles is a compound domino. Rather than having a simple inverter after a dynamic gate, as in domino, more complex static gates such

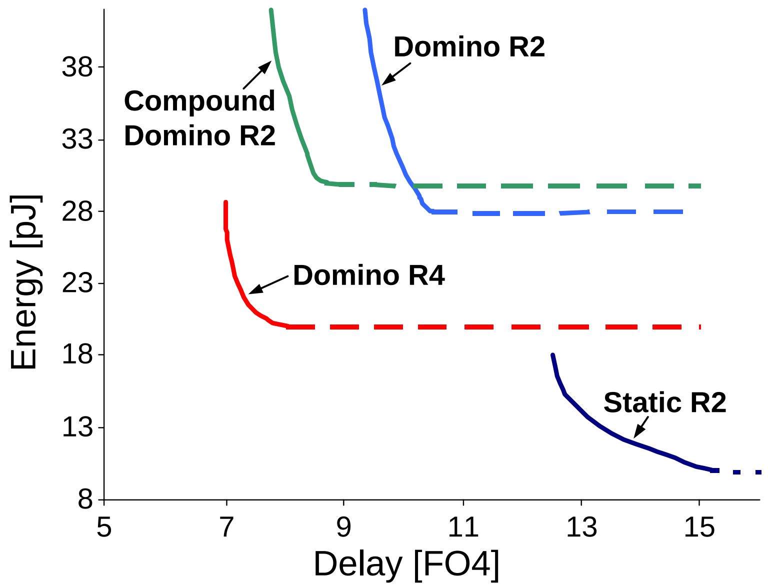


Figure 4.5: Energy-Delay space in 64-bit CLA adder implemented using domino and static CMOS logic styles (courtesy of R. Zlatanovici).

as and-or-inverts are inserted in compound domino. These gates actually perform a radix-2 carry merge operation. Although fairly fast, they are also quite power-hungry. Adders implemented with domino gates consume more energy than static counterparts and are typically used for performance critical units, such as microprocessor ALU cores. Detailed analysis of energy-delay trade-offs in CLA adders can be found in [39], [40]. Zlatanovici et al [41] demonstrated that the fastest 64-bit adder topology is a radix-4, sparseness-2 CLA tree based on Ling's equations [42] achieving a 7.5FO4 delay in a 90nm CMOS process.

This work focuses on ASIC realizations of wireless communication algorithms, so the implementations are based on standard-cell static CMOS gates, which are suitable for synthesis environment and also for aggressive supply voltage scaling [43].

4.1.2 Low-Swing Clocking

Due to intrinsically higher switching activity of the clock nodes, it is attractive to distribute clock signal with reduced swing. The questions are how much performance degradation this incurs and what are the optimal topologies of clocked storage elements suitable for reduced swing clocking.

The analysis considers flip-flops, logic, and the clock distribution network. The preferred flip-flop topologies for low-swing clocking (low-V_{Clk}) are different from those normally used in a full-swing (high-V_{Clk}) design. Clock buffers are redesigned to accommodate reduced drive current under low-V_{Clk}. In order to maintain performance, logic and flip-flops must absorb the increase in clock skew due to low-V_{Clk}.

Low-swing clocking cannot be implemented by simply scaling down the supply voltage of the clock subsystem. To maintain the cycle time and overall performance of a system, the clocked storage elements, logic blocks, and clock distribution network must be optimized. Extra timing overhead, namely the delay increase in flip-flops Δt_{FF} and the clock skew increase in the clock distribution network Δt_{Skew} has to be distributed among clock tree, flip-flops, and logic blocks, Fig. 4.6, in order to keep the performance equal to that of the traditional full-swing clock design. This section elaborates design considerations when making all these optimizations and determines a design point at which low-swing clocking yields the largest energy returns without performance loss. Results are based on simulation data of a simple 64-bit ALU in a 130nm dual-V_T CMOS technology.

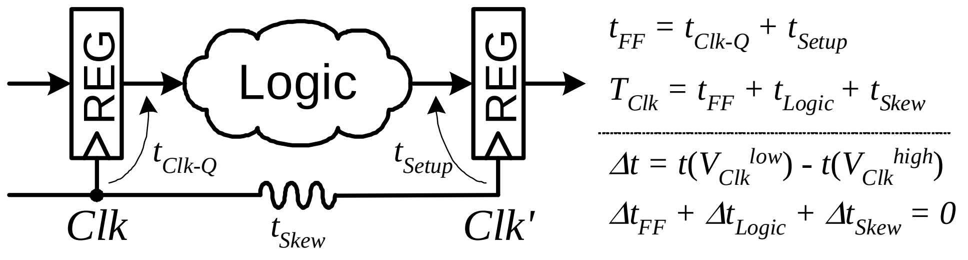

REG
Logic
REG
$t_{Clk\text{-}Q}$
t_{Setup}
Clk
t_{Skew}
Clk'
$t_{FF} = t_{Clk\text{-}Q} + t_{Setup}$
$T_{Clk} = t_{FF} + t_{Logic} + t_{Skew}$
$\Delta t = t(V_{Clk}^{low}) - t(V_{Clk}^{high})$
$\Delta t_{FF} + \Delta t_{Logic} + \Delta t_{Skew} = 0$

Figure 4.6: To maintain cycle time T_{Clk} at low-V_{Clk}, logic must absorb increase in flip-flop delay, Δt_{FF}, and clock skew, Δt_{Skew}.

A clock voltage of $0.7V_{dd}$ is used, which is a commonly used ratio resulting in a good compromise between power reduction and increase in delay [44]. The analysis can be extended to the optimization of V_{Clk} as well, but this is rather application-dependent. A fixed $V_{Clk} = 0.7V_{dd}$ is assumed to illustrate the general energy-delay trade-offs involved in the design of low-V_{Clk} systems. Flip-flops suitable for low-swing clocking are compared in terms of performance and energy consumption, relative to their high-V_{Clk} counterparts. Then the impact of low-swing clocking on the 64-bit ALU example is analyzed, along with the effects of low-V_{Clk} on the clock distribution network. Finally, the overall energy reduction of the low-V_{Clk} design is estimated, for the case where performance equals that of the reference high-V_{Clk} design.

Flip-Flops with Low-Swing Clock

Standard flip-flops for traditional full-swing clocking cannot be used with a low-V_{Clk}, since any clocked PMOS transistor will not fully turn off and cause static current consumption and reduced noise robustness. To combat this problem, a separate well-bias may be used for PMOS transistors to reduce static currents [45], but this does

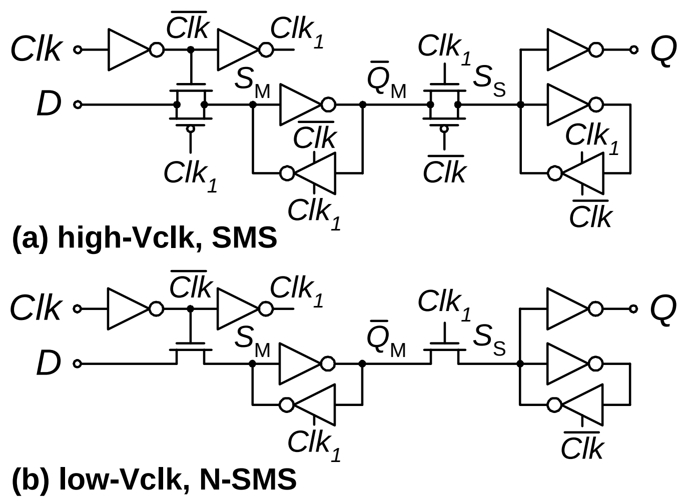


Figure 4.7: Flip-flops for non-critical paths: (a) high-V_{Clk}, static master-slave latch-pair (SMS), (b) low-V_{Clk}, NMOS-only clocked SMS (N-SMS).

not fully eliminate the leakage of the PMOS devices. In order to eliminate the PMOS leakage, low-V_{Clk} is applied to NMOS transistors only. Therefore, flip-flops with NMOS-only clocked transistors are analyzed. These flip-flops are derived either from their high-V_{Clk} counterparts or constructed using NMOS-only circuits. Generally, two types of flip-flops are needed: one for performance non-critical paths and another for performance critical paths.

Non-critical paths: Low-power master-slave (MS) latch-pairs are used. Low-V_{Clk} NMOS-only clocked static MS design (N-SMS) in Fig. 4.7(b) is obtained from the standard high-V_{Clk} SMS of Fig. 4.7(a) by simply removing the clocked PMOS transistors and by gating only the pull-down side in the feedback keepers. In N-SMS, full-swing at state nodes S_M and S_S is provided only after the feedback keepers are enabled. Thus, some additional performance degradation is expected.

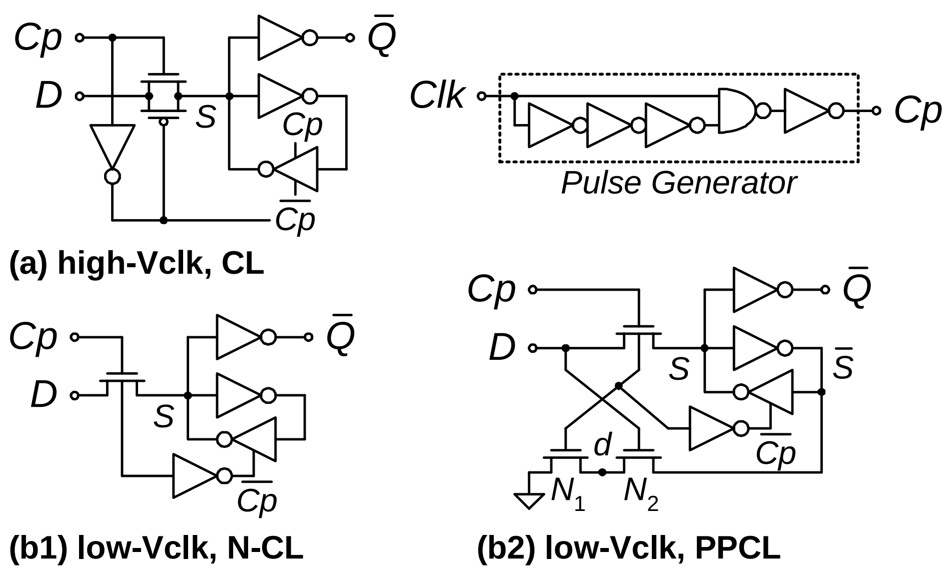


Figure 4.8: Flip-flops for performance-critical paths: (a) high-V_{Clk}, cycle-latch (CL), (b1) low-V_{Clk}, NMOS-only clocked cycle-latch (N-CL), (b2) low-V_{Clk}, push-pull NMOS-only clocked cycle-latch (PPCL).

Critical paths: High-speed explicitly-pulsed latches are used. The N-CL shown in Fig. 4.8(b1) is an NMOS-only clocked cycle-latch derived from the full transmission gate-based cycle-latch (CL) shown in Fig. 4.8(a), [35]. The push-pull NMOS-only clocked cycle-latch (PPCL) in Fig. 4.8(b2) is constructed from the N-CL by adding N_1 and N_2 for faster pull-up of state node S and also for improved robustness to clock noise. The weak pull-up performance of the NMOS pass-gate when $D = 1$ is improved by differentially discharging the opposite side of the keeper through the pull-down path $N_1 - N_2$. Additionally, the gate signals on N_1 and N_2 are ordered in a way that allows time-borrowing. The pulsed structure has an inherent negative setup time, meaning that the clock pulse C_p can arrive before the data D, so C_p "high" discharges node d prior to data arrival, further improving the setup time and delay.

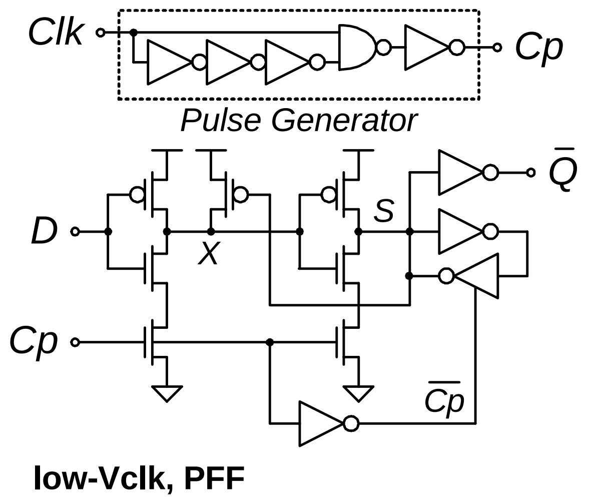


Figure 4.9: Pulsed flip-flop (PFF) suitable for low-V_{Clk} operation.

Straight derivation of low-V_{Clk} flip-flops from their high-V_{Clk} counterparts limits design choices. Flip-flops rarely used with high-V_{Clk} due to low-performance or high-energy should also be considered. An example of such a circuit is an NMOS-only clocked explicitly pulsed flip-flop (PFF) in Fig. 4.9. The idea is to use semi-dynamic or dynamic gates for restoration of logic "1" when low-swing signals drive the pull-down network, [46]. The low-swing clock pulse C_p drives the gates of NMOS transistors and restores nodes X and S to a full-swing.

All flip-flops are gate-size optimized using the framework in [35], which minimizes energy subject to a delay constraint. In each circuit, the output load is fixed at 20fF and the input capacitance restricted to 5fF or less. The simulation results are shown in Figs. 4.10-4.12, where D-Q delay is the total data-to-output delay normalized to a FO4 inverter. Energy-delay comparisons of FF designs with high-V_{Clk} (Fig. 4.10) and low-V_{Clk} (Fig. 4.11) reveal that CL offers the best performance at high-V_{Clk} [2],

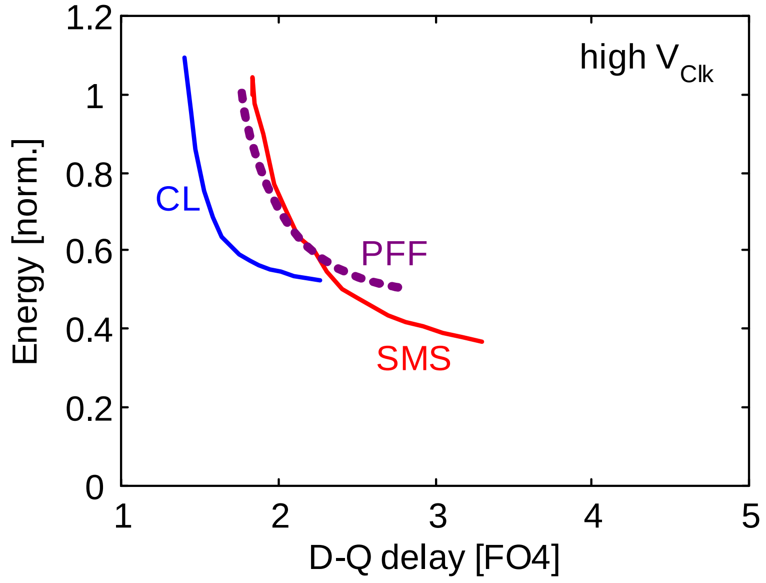


Figure 4.10: Energy-Delay space for flip-flops optimized for high-V_{Clk} operation. Preferred high-V_{Clk} flip-flops are CL and SMS.

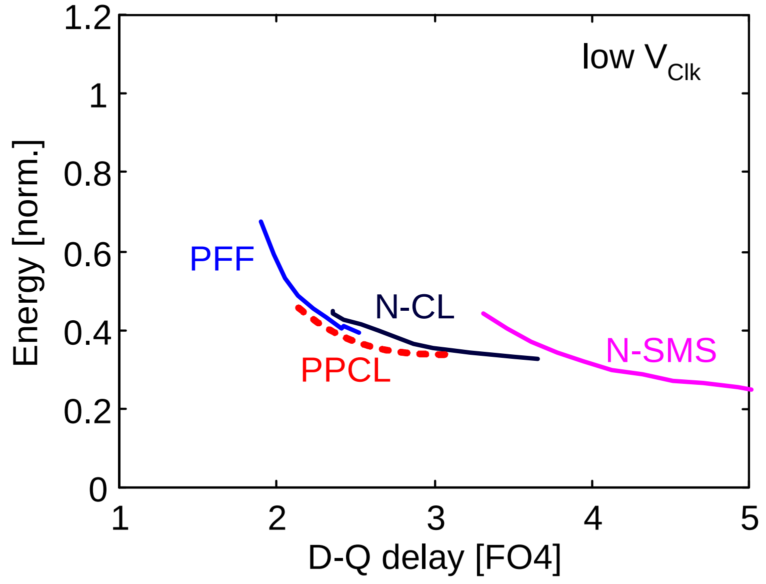


Figure 4.11: Energy-Delay space for flip-flops optimized for low-V_{Clk} operation. Preferred low-V_{Clk} flip-flops are PFF/PPCL and N-SMS. ($V_{Clk} = 0.7V_{dd}$)

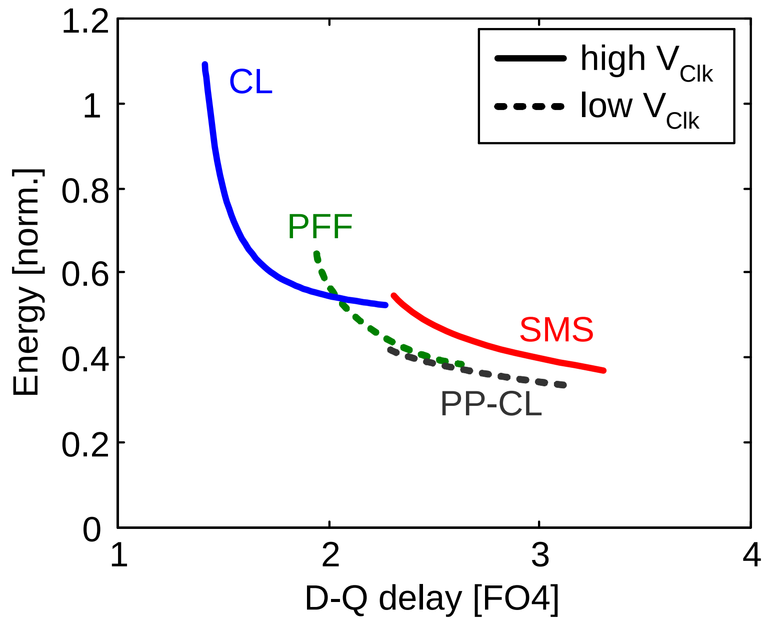


Figure 4.12: Low-V_{Clk} flip-flops cannot achieve the top speed of high-V_{Clk} flip-flops, but have lower energy at less aggressive delays.

while PFF or PPCL are the preferred designs for low-swing clocking. Note that PFF design is sub-optimal with high-V_{Clk}.

Low-V_{Clk} flip-flops (PFF & PPCL) cannot achieve the top speed of the high-V_{Clk} flip-flops (CL), but offer lower energy at delays greater than 2FO4, as illustrated in Fig. 4.12. Any energy reduction in the flip-flop itself is a side benefit; the main energy savings in low-V_{Clk} comes from energy reduction in the clock distribution network. Detailed analysis of flip-flop energy-delay characteristics can be found in [47], including flip-flop derivations and classification.

After balancing the energy-delay sensitivity in a data-path logic consisting of a logic block surrounded by pipeline registers, the next step is to investigate energy-performance characteristics using techniques of parallelism and pipelining.

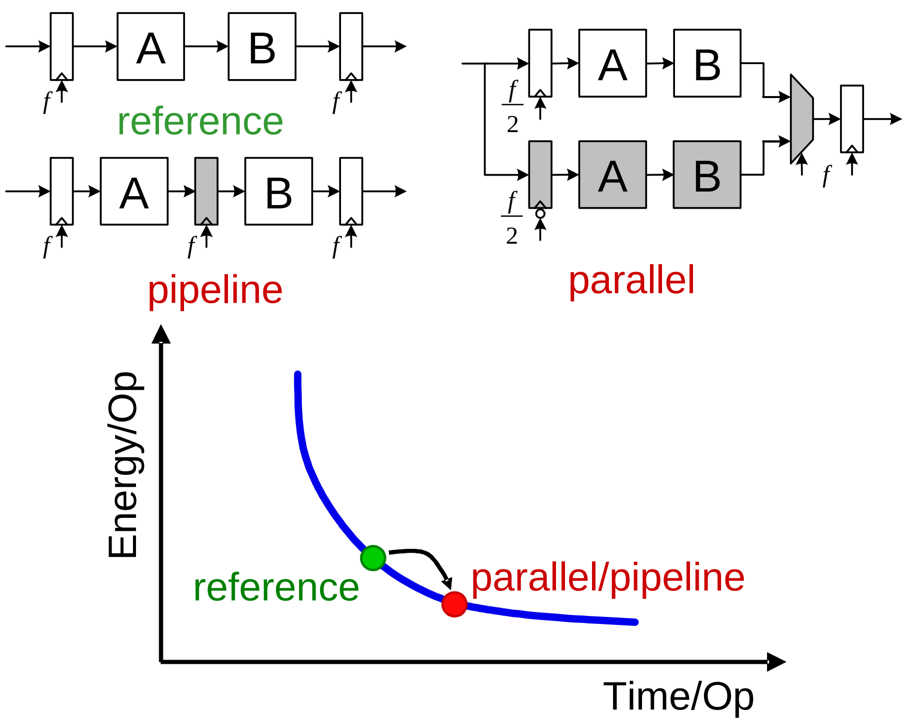


Figure 4.13: Parallelism and pipelining relax timing constraints on logic blocks.

4.2 Parallelism vs. Pipelining

This section revisits the original work of Chandrakasan et al. [19] that evaluated the energy efficiency of a parallel and a pipelined design. It has been shown that both parallelism and pipelining combined with voltage scaling can improve the energy efficiency while maintaining the throughput. Previous work is extended by introducing the threshold voltage as an additional variable in the optimization. The results are evaluated based on the optimal energy-delay trade-off curve for the pipeline logic.

Schematics of the reference, parallel, and pipelined circuits are shown in Fig. 4.13. Parallelism and pipelining are employed to relax timing constraints on the underlying blocks A and B as shown with the energy-delay trade-off in Fig. 4.13. These techniques are particularly useful when the energy of the reference design becomes too costly. In pipelining, an extra register is inserted between blocks A and B, effectively doubling the available computation time for each of the blocks. In a parallel design, the area is doubled by operating the two blocks in parallel. However, the available computation time is also doubled for each block since every other input operand is evaluated in an interleaved fashion.

The nominal design is an add-compare unit which uses the adder described in Sec. 3.4.3 for both the adder (block A) and the comparator (block B). In this example, the SMS latch-pairs of Fig. 4.2(b) are used. The nominal design is first optimized through gate sizing to achieve minimum delay under V_{dd}^{max} and V_{th}^{ref}. Using the throughput of this design as a constraint and energy-delay trade-offs of the adder and comparator blocks from the inner layer, the energy needed for the nominal design and its parallel or pipelined implementation can be estimated.

For all designs in Fig. 4.13, the optimal value of the supply and threshold voltage that results in minimum energy for a given throughput constraint is found, along with the corresponding E_{Lk}/E_{Sw} ratio. The minimal energy is found by V_{dd}-V_{th} optimization, in which ΔV_{th} is swept from 0 to -200mV in steps of 5mV. Each time V_{th} is modified, V_{dd} is adjusted to achieve the target throughput with minimal energy, using the multi-variable sensitivity information from the lower-level blocks. The goal of this sweep is to find the optimal point $(V_{dd}^{opt}, V_{th}^{opt})$ for each micro-architecture and to illustrate the trend around the optimal point, as shown in Fig. 4.14.

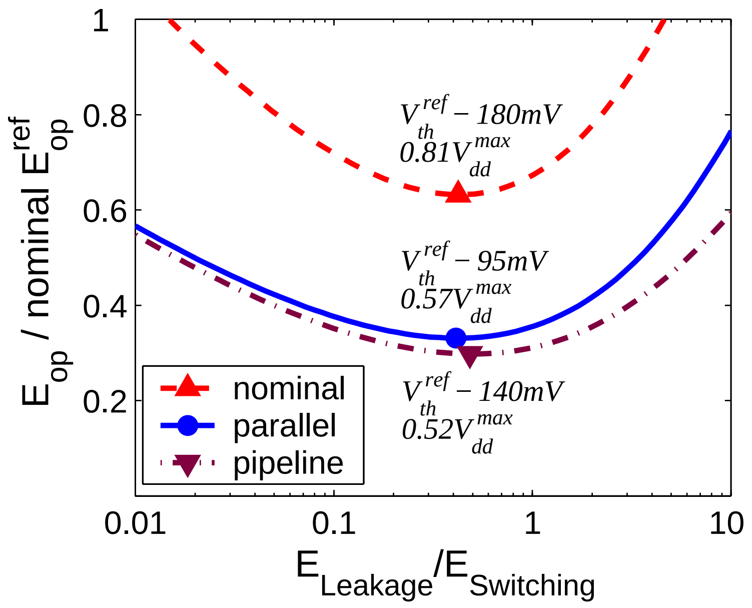


Figure 4.14: Energy-per-operation as a function of the leakage-to-switching energy ratio in nominal, parallel, and pipeline designs. All designs operate at the throughput of the nominal design sized for minimum delay under V_{dd}^{max} and V_{th}^{ref}.

Energy-per-operation in all three designs is compared to the nominal case which operates at V_{dd}^{max} and V_{th}^{ref}. For each design, the optimal $(V_{dd}^{opt}, V_{th}^{opt})$ point is reached when the supply and threshold voltage sensitivities of the underlying blocks are equal.

4.2.1 Optimal Leakage and Switching Energy

It has been shown that parallelism is more energy-efficient than pipelining when the leakage energy is about an order of magnitude smaller than the switching energy [19]. However, as devices become leakier, the larger area of parallel design causes the balance between the switching and the leakage energy to occur at a higher supply voltage than in a pipelined design. This is due to lower effective activity of the parallel design. Equivalently, parallelism decreases the amount of time that a device spends on computations, thereby increasing the ratio of wasted (leakage) to useful (switching) energy. Hence, a parallel implementation achieves smaller energy savings though the difference is very small. A parallel implementation may still be preferable since the energy saving in the pipelined design depends on determining the ideal locations for pipeline latches. In many systems, these points are hard to find.

Observe that the energy-per-operation as a function of the leakage-to-switching energy ratio has a very shallow minimum, as shown in Fig. 4.14. This follows from the logarithmic dependence of the E_{Lk}/E_{Sw} ratio on the logic depth and activity [26]. In this example, the optimal E_{Lk}/E_{Sw} ratio is around 40% for all thee implementations, roughly corresponding to that of its main sub-block, the adder. When considering extreme circuit examples with significantly different switching activities, such as inverter chains and memory decoders [26], it turns out that the optimal E_{Lk}/E_{Sw} ratio of these circuits ranges from 0.2 to 0.8. Since the minimums of the energy curves are very shallow in the range of leakage-to-switching ratio from 0.1 to 1 (Fig. 4.14), the conclusion is that the total energy is minimized at the point where the leakage energy is about half of the active energy.

4.2.2 Optimal Level of Parallelism

Parallelism is most efficient when the target delay is lower than the minimum achievable delay of the underlying blocks. Parallelism of level P implicitly relaxes the delay

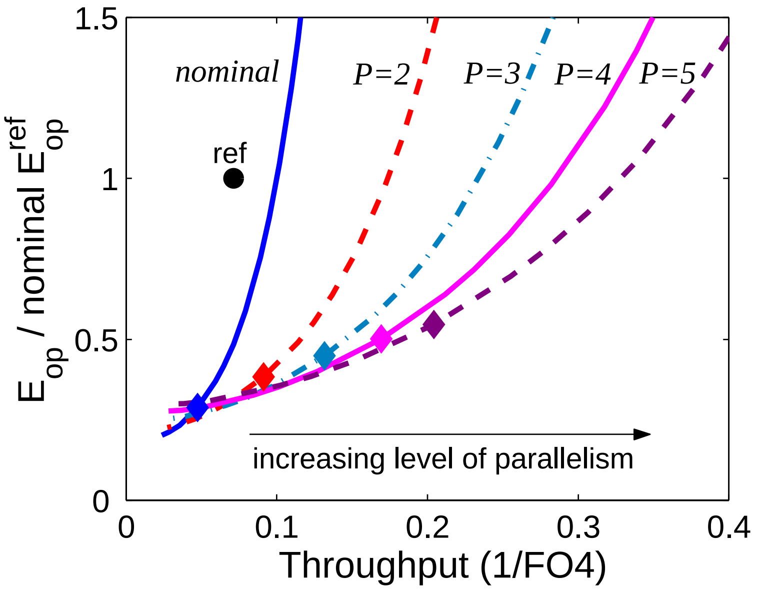


Figure 4.15: Energy-per-operation as a function of the clock cycle in energy-efficient designs with levels of parallelism from $P = 2$ to $P = 5$. Delay and energy penalty due to multiplexers is included. Diamond dots indicate min EDP, circle indicates nominal design initially sized for minimum delay at V_{dd}^{max}, V_{th}^{ref}.

of the underlying logic about P times (minus the flip-flop delay). Graph in Fig. 4.15 shows the energy-performance space for designs exhibiting parallelism from $P = 2$ to $P = 5$, together with the nominal or reference design, all normalized to the reference design point as defined in Chapter 3. The results are obtained from joint supply-threshold-size optimization, with the external load of $C_L = 32$. Delay and energy overhead due to the additional multiplexer at the output is included in the optimization. The data shows that a parallel architecture provides an increase in performance at a very small marginal cost in energy. As a result, addition of more parallel units possibly improves the performance/throughput beyond the maximal throughput of the nominal design. For instance, the parallel architecture $P = 5$ provides about a 3× performance improvement over the nominal design, compared at unit energy, Fig. 4.15. Parallelism is also an option for energy reduction, which is a well known result [19], [48]. The largest energy savings due to parallelism are achieved when the sensitivity to circuit parameters of the reference design become very large.

Minimum EDP is the point at which any given architecture has equal marginal cost in energy and delay, allowing for energy-efficient optimization around that point. At the minimum EDP point in Fig. 4.15, the performance of the design increases with increasing levels of parallelism. This indicates that added parallelism is suitable for boosting performance. Additionally, allowing more levels of parallelism gives a wider range of performance over which energy may be optimized. Relative to the performance level corresponding to the minimum EDP for an architecture, more parallel units are needed to support higher performance. For a reduced performance, the level of parallelism should decrease as well.

While parallelism is a very efficient technique for improving the performance, the area of the parallel design is, to a first order, in linear proportion with the level of parallelism P. Therefore, one must always keep in mind the energy-area trade-off of the architectures which employ parallelism.

4.2.3 Optimal Vdd and Vth

This section presents a procedure for finding the optimal supply and threshold voltage for a given architecture. The procedure is based on tuning the parameters V_{dd} and

V_{th} until the desired ratio of leakage and switching power is achieved, for a given performance level.

The fact that the optimal E_{Lk}/E_{Sw} is around 0.5 (Fig. 4.14) provides a way to quickly estimate the optimal V_{dd} and V_{th} in a function block. Using the dependence of critical-path delay on V_{dd} and V_{th} and the dependence of E_{Lk}/E_{Sw} ratio on V_{dd} and V_{th}, the result in Equations (4.2), (4.3), (4.4) is obtained.

$$\Delta V_{th} = V_0 \cdot ln\left(\frac{(E_{Lk}/E_{Sw})_{init}}{(E_{Lk}/E_{Sw})_{opt}} \cdot \frac{D_{opt}}{D_{init}}\right) \tag{4.2}$$

$$V_{dd}^{opt} = \frac{V_{on}^{init} + \Delta V_{th} + \chi \cdot \frac{\alpha_d - 1}{\alpha_d} \cdot (V_{dd}^{init})^{1/\alpha_d}}{1 - \frac{\chi}{\alpha_d} \cdot (V_{dd}^{init})^{(1/\alpha_d - 1)}} \tag{4.3}$$

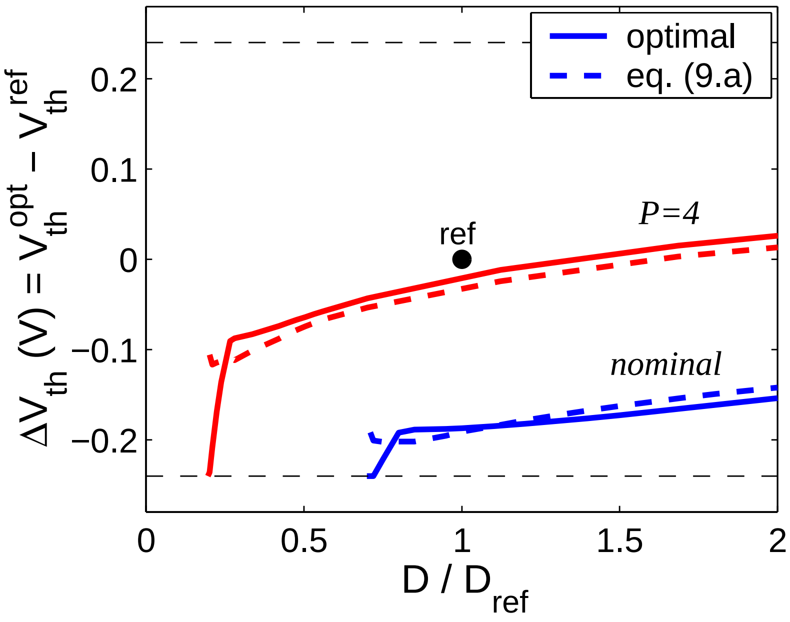
$$\chi = \frac{V_{dd}^{init} - V_{on}^{init}}{(V_{dd}^{init})^{1/\alpha_d}} \cdot (D_{init}/D_{opt})^{1/\alpha_d} \tag{4.4}$$

where indices *init* and *opt* indicate the initial and optimal design points, respectively. The above analysis first changes ΔV_{th} to force the E_{Lk}/E_{Sw} to be about 0.5, and then changes V_{dd} to achieve the desired performance.

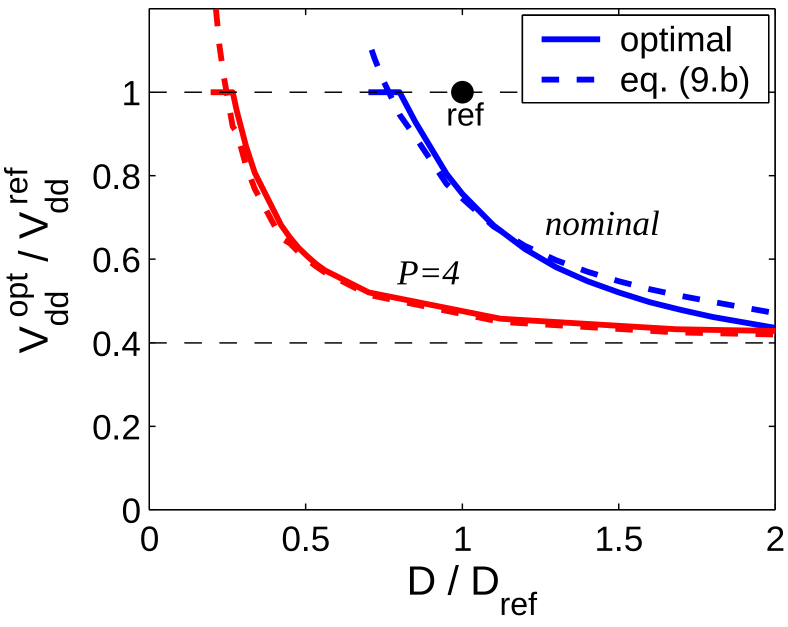
Equation (4.2) finds the change in V_{th} by estimating the required change in leakage current. It can be easily derived by noticing that the leakage current is equal to the leakage energy divided by cycle time, assuming that the change in the switching energy is small. Equations (4.2) and (4.3) follow the analysis in [23] and linearize the alpha-power law equation by taking Taylor expansion about V_{dd}^{init}. This expression relates performance to V_{dd} and V_{th} to derive optimal V_{dd} needed to achieve the desired performance D_{opt} under the new V_{th}.

The optimal design point determined above is then used as an initial point for subsequent iterations, approaching the optimal energy-delay trade-off point. Applying the same procedure to other design points, the full energy-delay trade-off curve for the design can be obtained. As an example, calculation of optimal V_{dd} and ΔV_{th} for the nominal and $P = 4$ topologies across a wide performance range is performed, as shown in Fig. 4.16. The plots are obtained using Taylor expansion about 1V in a $0.13\mu m$ technology. The values obtained from analytical models and by optimization closely match, thus verifying the analysis. The deviation from ideal ΔV_{th} is because the analysis assumes a negligible change in V_{dd} and also due to sub-optimal E_{Lk}/E_{Sw} ratio at V_{dd}^{max}. This analysis also provides insight about the tunable range of supply and threshold voltages.

Among the circuit examples studied in Chapter 3, the memory decoder has E_{Lk}/E_{Sw} ratio of 0.1 under V_{dd}^{max} and V_{th}^{ref} ($E_{Lk}^{ref}/E_{Sw}^{ref} = 10\%$), which is the closest to the optimal ratio of 0.5. The optimal V_{th} in the decoder is therefore close to standard V_{th}^{ref} for the technology. In the adder with $E_{Lk}^{ref}/E_{Sw}^{ref} = 1\%$ and inverter chain with $E_{Lk}^{ref}/E_{Sw}^{ref} = 0.1\%$, V_{th}^{ref} is about 200mV higher than the optimal V_{th} in these circuits, because of the lower E_{Lk}/E_{Sw} in their respective reference designs. These three circuit examples span about three orders of magnitude in the leakage-to-switching energy ratio under V_{dd}^{max} and V_{th}^{ref} and, as such, they can serve as good examples for the V_{dd} and V_{th} tuning range for a particular technology.



(a)



(b)

Figure 4.16: Plot of (a) change in threshold ΔV_{th} and (b) optimal supply voltage $V_{dd}^{opt}/V_{dd}^{ref}$ after performing energy-efficient V_{dd}-$V_{th} - W$ optimization on nominal and parallel-4 designs. Dot represents initial sizing for minimum delay at V_{dd}^{max}, V_{th}^{ref}.

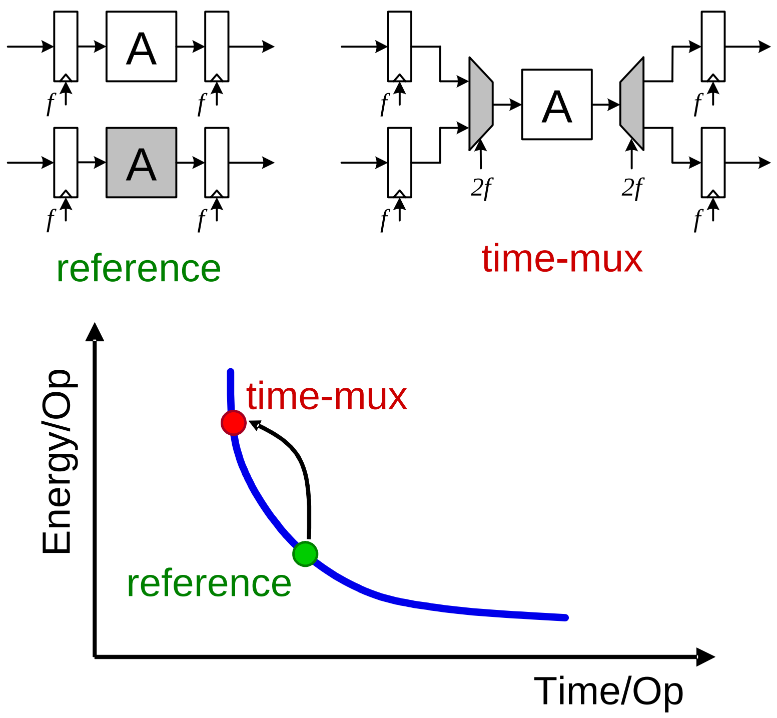


Figure 4.17: Time-multiplexing tightens timing constraints on logic block A.

4.3 Time-Multiplexing

Earlier discussion showed that parallelism, as illustrated in Fig. 4.13, can improve performance or reduce energy at the expense of an increased area. Time-multiplexing, as shown in Fig. 4.17, does the opposite; it reduces the area at the expense of an increase in energy. Parallel-to-serial conversion is required at the input of logic block A, which is shared among multiple incoming data streams. The energy increase is primarily due to increased speed of processing element A that is required to maintain the throughput. As in parallelism, block A is assumed to be the 64-bit adder. Therefore, reduction in area or energy cannot be the only goal in the optimization since there is a trade-off between energy and area.

4.4 Energy-Area Trade-Off

Both area and energy affect the overall cost of a design. Area is most commonly related to the dollar cost of producing chip wafers, while energy is associated with chip cooling or battery capacity in portable designs. Intuitively, the cost of a design is minimized when an optimal trade-off between energy and area is reached. A design cost function $C(x)$ that considers both energy and area can be formulated as given by Eq. (4.5).

$$\begin{aligned} \text{minimize} \quad & C(x) = E(x) + \beta \cdot A(x) \\ \text{subject to} \quad & D(x) \leq D_{con} \end{aligned} \tag{4.5}$$

The quantity $x \in \Re^n$ is an n-dimensional vector of tuning variables; $E(x)$ and $A(x)$ are the total energy and area of the design, respectively. Parameter β is the weight-factor that defines contribution of area in the design cost, and D_{con} is the delay or performance constraint. Some of the optimization variables do not affect area; for example, the supply and threshold voltages affect only energy. Some other variables such as parallelism or time-multiplexing affect both energy and area, with area impact being much larger when these techniques are applied to large blocks.

Since time-multiplexing and parallelism/pipelining differ in area significantly, investigation of energy-area trade-offs under a fixed throughput requirement is needed. Architectures with different levels of parallelism and time-multiplexing span a wide

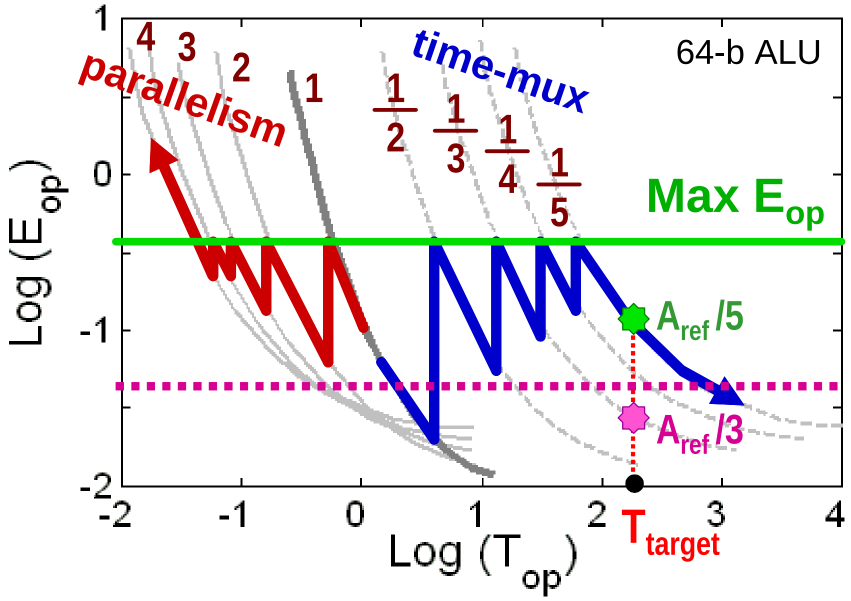


Figure 4.18: Energy-delay space for designs with various levels of parallelism and time-multiplexing. Desired cycle time T_{target} can be achieved with several architectures that differ in energy and area. Numbers represent the area of respective architectures.

performance range while trading energy for area. For a fixed energy budget, parallelism improves performance at the expense of an increase in area, while time-multiplexing is suitable for low-throughput while improving the area.

The basic energy-area trade-off is illustrated in Fig. 4.18 for a 64-bit ALU implemented with varying degrees of time-multiplexing and parallelism. Each of the performance targets can be achieved with several micro-architectural choices differing in energy and area. Assume that the maximum energy per operation *max* E_{op} is indicated by the solid horizontal line. Under this energy constraint, a five-fold time-multiplexed architecture meets the target cycle time T_{target} with minimal area, which is roughly one-fifth the area of the reference architecture. If the energy limit is more

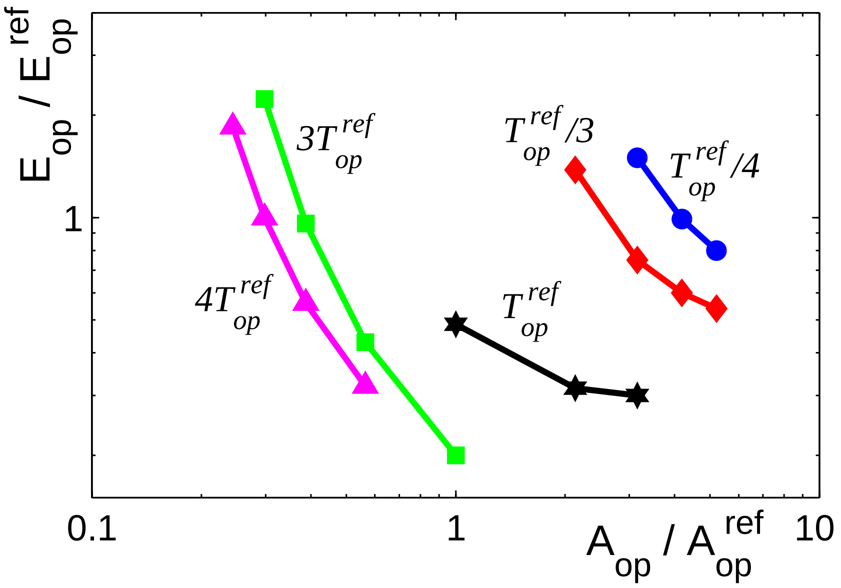


Figure 4.19: Energy-area trade-off under performance constraint. All parameters are normalized to reference architecture optimized for speed under V_{dd}^{ref} and V_{th}^{ref}.

aggressive, as indicated by the dashed horizontal line, three-fold time-multiplexing would have to be used to satisfy the new energy budget. The area required to maintain the target performance also increases from $A_{ref}/5$ to $A_{ref}/3$.

Optimal architecture selection depends on the available energy and area budget. Figure 4.19 shows contours of constant performance in the energy-area space. The results confirm that high-throughput designs generally require larger area than the low-throughput designs since parallelism must be employed for speed improvement. A larger energy budget therefore allows for a smaller circuit area, so the optimum is found at the point where the overall chip cost given by Eq. (4.5) is minimized.

The results derived in this chapter for a simple data-path are fundamental and can be extended to an arbitrary level of complexity. Next chapter takes the architectural concepts established here to study some basic signal processing operations in the context of their energy-delay-area optimality.

Chapter 5

Signal Processing Techniques

This chapter reviews some commonly used signal processing techniques applicable to multi-dimensional algorithms. Techniques of data-stream interleaving and folding are evaluated using the energy-delay trade-off in the pipeline logic, which leads to the exploration of loop retiming. The use of interleaving and loop retiming is illustrated for an iterative square rooting and division. These iterative algorithms are applicable in adaptive multi-carrier wireless communications.

The practical realization of a signal processing technique is tightly coupled with the energy-performance characteristics of the underlying circuits. This basic trade-off explored in previous chapters is repeated in Fig. 5.1. In order to maintain good energy-delay point (indicated with the dot), parallelism is used for higher throughput, while time-multiplexing is used for lower throughput rates. The indicated energy-delay point is good since marginal returns to energy and delay around that point are

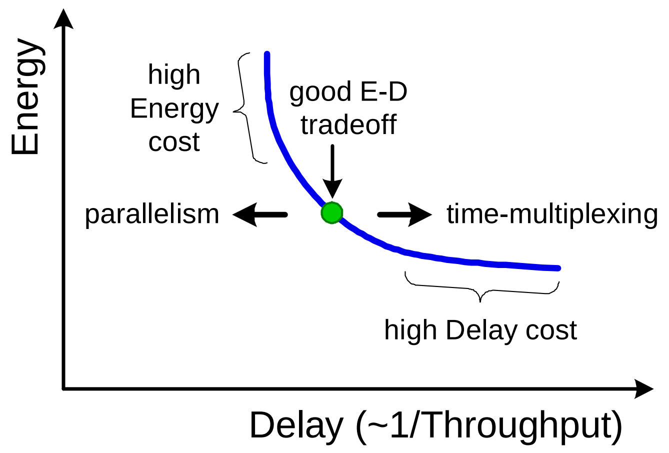


Figure 5.1: Energy-delay trade-off in digital circuits.

similar. Otherwise, energy or delay cost becomes high, as shown in Fig. 5.1.

The choice of micro-architecture is highly influenced by throughput, which requires analysis of techniques that can span a large throughput range. The basic concepts of parallelism and time-multiplexing from Chapter 4 enable this. In this chapter, data-stream interleaving and folding are introduced to support more complex operations with concurrent or time-serial execution, which may also involve feedback loops.

5.1 Data-Stream Interleaving

Data-stream interleaving is a technique that improves area efficiency. In essence it is a way of time multiplexing the data. The analysis in this section is focused on a specific case with feedback loops, which is a concept commonly found in recursive computation. It is shown that interleaving of independent data streams reduces area savings without significant increase in energy per operation, because interleaving essentially requires up-sampling together with deeper pipelining.

The case when the algorithm contains a recursion is analyzed as described with the simple difference equation ($c \neq 0$):

$$z(k) = x(k) + c \cdot z(k-1) \tag{5.1}$$

that updates the output sample at time k as a sum of input sample at time k and scaled version of the previous output sample $z(k-1)$.

The signal processing representation of Eq. (5.1) is shown in Fig. 5.2(a). The latency of one symbol period is represented with an explicit register between the output $z(k)$ and the multiply block. The symbol rate is defined by clock frequency f_{Clk}. This simple model abstracts away any extra latency in the *add* and *multiply* blocks. From an implementation perspective, an N bit *add* operation is less complex in terms of logic gates and area than an $N \times N$ multiplication. Thus, to balance the complexity of datapath logic blocks, the *add* and *multiply* operations require different latencies.

A more realistic model of Eq. (5.1) captures the different latency of addition and multiplication as shown in Fig. 5.2(b). The latency is described by adding a and m registers to the output of the adder and the multiplier, respectively. Going around the loop, the feedback signal $y(k-1)$ will have an increased latency of $a+m$. Increasing the clock rate by a factor $a+m$ maintains the same algorithmic latency. In order to

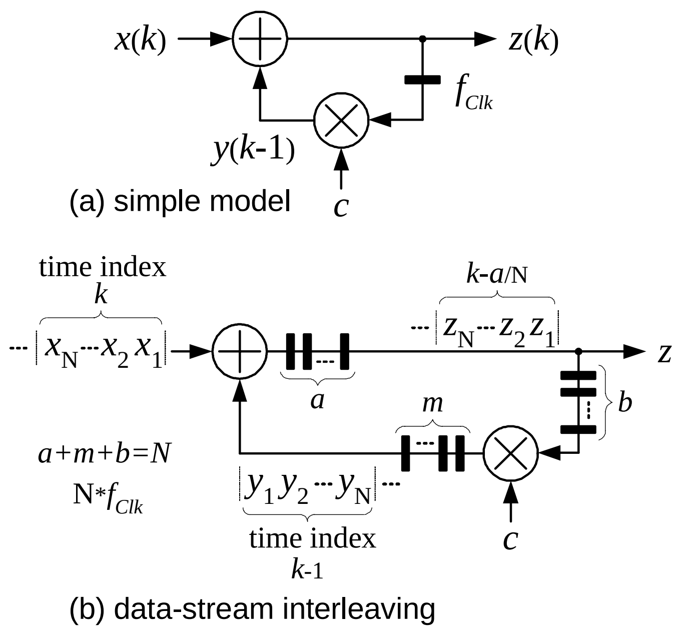


Figure 5.2: Concept of data-stream interleaving. (feedback example)

fill the added pipeline with useful computation, multiple independent signal streams can be interleaved onto the same hardware. If the number of independent signals N is greater than $a + m$, then $b = N - a - m$ additional pipeline registers are required to maintain the latency. Interleaving effectively improves the area efficiency by sharing data-path logic across independent streams of data. A practical use of interleaving is in multi-carrier communications, where the independent narrow-band sub-carrier streams can be time-interleaved.

By sharing logic blocks between pipeline registers, interleaving improves data-path utilization and therefore increases area efficiency. Energy remains the same, to a first order (neglecting the impact of registers, i.e. assuming long chains of logic), since interleaving is equivalent to pipelining and up-sampling. Examining the energy-delay

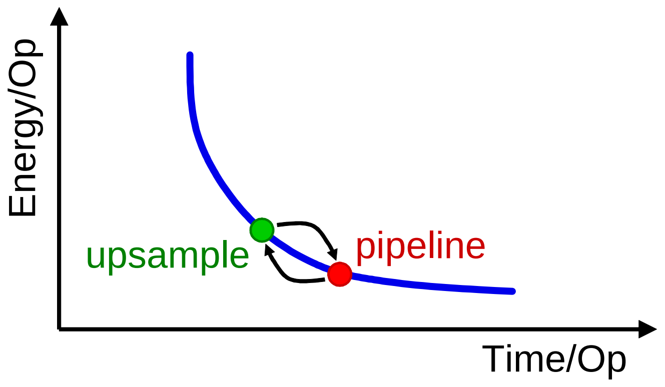


Figure 5.3: Interleaving and Folding are area saving techniques. Timing and energy stay approximately the same (neglecting register overhead).

(E-D) line for data-path logic, Fig. 5.3, suggests that pipelining moves the operating point down toward lower energy and longer delay, but up-sampling brings it back to the original position in E-D space. The technique of interleaving is applicable for concurrent computation. For time sequential ordering, the idea of folding is considered.

5.2 Folding

Folding is much like data-stream interleaving, except that not all data samples are independent. To analyze this, assume serially ordered execution of the same operation as shown in Fig. 5.4, where the first block in the chain takes independent data samples $y_1(k)$. All other blocks down the chain take the result from previous block. Block Alg performs some algorithmic operation with feedback loops.

Assume serially ordered execution of some algorithmic operation Alg as shown in

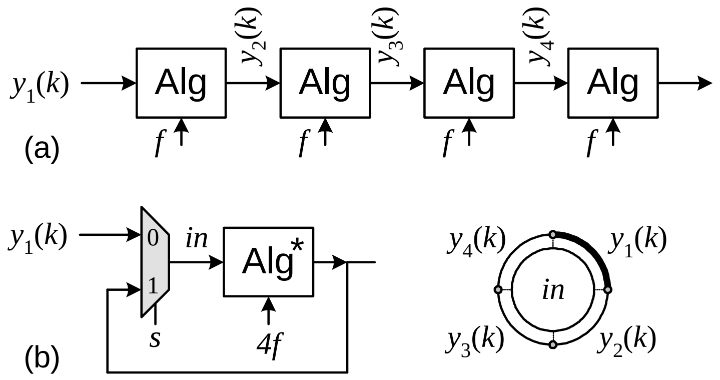


Figure 5.4: Concept of folding: (a) time-serial computation, (b) operation folding. Block *Alg* performs some algorithmic operation. (star indicates deeper pipelining)

Fig. 5.4(a). The first block in the chain takes independent data samples $y_1(k)$, all other blocks take the result from the previous block. The concept of folding is shown in Fig. 5.4(b), [49]. The input to Alg^* is provided by a multiplexer, which selects the primary input y_1 or internal results y_2, y_3, y_4. During the first quarter of the symbol period, up-sampled and interleaved data y_1 is used. The output of Alg^* is then folded over in time, back to its input, to compute y_2, y_3, and y_4. Re-ordering of incoming y_1 samples is needed to align data at input in, as shown in the life-chart in Fig. 5.4(b). Up-sampling the clock in block *Alg* is necessary to sustain the external (y_1) throughput rate.

If *Alg* is a simple feed-forward unit, no additional pipeline registers are needed. In case *Alg* block has internal feedback loops, additional pipelining is necessary to store the internal states. This raises the issue of optimally distributing pipeline registers around loops to maximize throughput or scale voltage for power minimization.

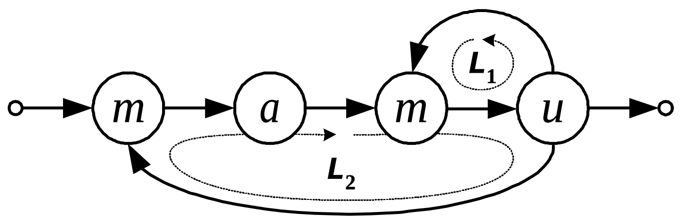


Figure 5.5: Data-flow graph model of iterative division. (m, a, u indicate latency)

5.3 Loop Retiming

Loop retiming is a technique of distributing pipeline registers around loops [50]. The goal is to assign the right amount of latency to functional blocks, and then distribute the pipeline registers inside the blocks to position all internal datapath logic blocks at the same point (shown in Fig. 5.1) in the energy-delay space. This guarantees top-level optimality. As in the case of interleaving, additional balancing registers may be needed to ensure equal loop latency in all recursive loops.

The approach to loop retiming is illustrated in Fig. 5.5 on the example of iterative division. This is a simple example with two nested loops, but the concept can be generalized to virtually arbitrary complexity. The analysis starts from a data-flow graph (DFG) representation of a function, where a, m, u are the latencies of pre-characterized adder, multiplier, and multiplexer blocks, respectively. This extends the retiming algorithm from [51] that assumes equal latency in all arithmetic blocks.

Starting from the DFG representation in Fig. 5.5, the retiming procedure is carried out as follows. For each loop, constraints are formulated as described in Eq. (5.2).

I/O latency is also needed for hierarchical expansion.

$$\begin{aligned} \text{loop } L_1\text{:} \quad & m + u + b_1 = N \\ \text{loop } L_2\text{:} \quad & 2m + a + u + b_2 = N \\ \text{I/O latency } (div^{(1)})\text{:} \quad & 2m + a + u \end{aligned} \tag{5.2}$$

where b_1 and b_2 are the number of balancing registers needed to satisfy the loop latency N. The next step is to solve for the latency parameters (m, a, u). Often times, system of equations appears underconstrained, but this is not really true since the cycle time is common for all the blocks. The cycle time, which is derived from a throughput requirement, determines latency parameters (m, a, u) in various functional blocks as shown in Fig. 7.5 in Chapter 7 for *add* and *multiply* operations. Finally, the right amount of balancing latency is determined in each of the loops in order to meet the loop latency constraints.

5.3.1 Hierarchical Loop Retiming

The retiming strategy above can be easily expanded to multiple layers of hierarchy. Each block hierarchy is characterized with internal loop constraints as well as latency from its primary inputs to its primary outputs. For example, I/O latency in Eq. (5.2) indicates the latency of a divider block at level 1 of hierarchy, $div^{(1)}$. This procedure is then hierarchically extended upward for the overall design optimization. At the top-level, loop constraints from all levels beneath are thus considered.

$$\begin{aligned}
&\text{loop } L_1^{(2)}: && div^{(1)} + 2m + 4a + 2u + b_1 = N \\
&\text{loop } L_4^{(2)}: && 3m + 6a + u + b_4 = N \\
&\text{loop } L_5^{(2)}: && 6m + 11a + 2u + b_5 = N + N
\end{aligned} \tag{5.3}$$

Equation (5.3) illustrates the retiming procedure at level 2 of hierarchy that uses the latency of the divider from Eq. (5.2) from level 1 of hierarchy ($div^{(1)}$ term, loop $L_1^{(2)}$). The hierarchical retiming approach corresponding to Eq. (5.3) is shown in Fig. 5.6 for iterative eigen-mode decomposition. A few loop constraints are detailed for this example (L_1, L_4, L_5) to illustrate the principle of hierarchical expansion. The use of delayed iteration for loop L_5 is presented in Section 5.4.

5.3.2 CPU Runtime Considerations

Retiming at the block level (e.g. adder or multiplier) can be done very efficiently in terms of CPU runtime using commercial tools for chip synthesis. Retiming of more complex feed-forward blocks is also fairly efficient, but the runtime required for retiming of designs with feedback loops increases in a super-linear fashion with the number of loops. For example, loop retiming of iterative *sqrt* and *div* takes about 15 minutes while eigen-mode decomposition block of Fig. 5.6, which is just 15 times more complex in terms of area, took over 40 hours of CPU runtime on a 64-bit dual-opteron machine with 4GB of memory.

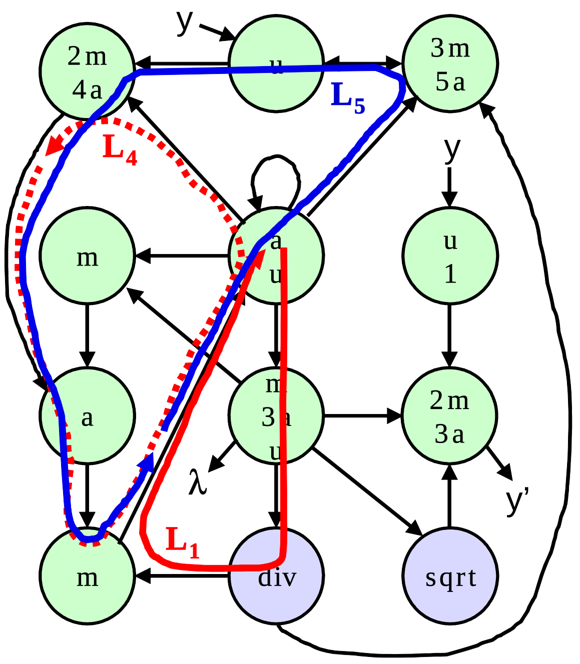


Figure 5.6: Data-flow graph model of an iterative eigen-mode decomposition algorithm. Iterative division in Fig. 5.5 is used. (m, a, u indicate latency)

To improve computational efficiency, the top-level design is partitioned into simple feed-forward blocks that can be efficiently retimed, and the results are incrementally compiled at the top-level. The runtime using this approach is roughly proportional to the number of different blocks. As an example, retiming of the eigen-mode decomposition algorithm takes about 45 minutes[1] with the proposed approach, which is an order of magnitude reduction compared to the straightforward top-level retiming method. The block-based approach is not only more computationally efficient, it also provides insight into the partitioning of the overall design and better understanding of the optimization results.

[1]Actual runtime is highly dependent on design constraints.

5.4 Delayed Iteration

Another technique which can be used for power and area efficient implementation is delayed iteration. The idea is to insert the extra sample period to the performance-critical loops to relax timing constraints for pipeline logic.

This concept is illustrated by Eq. (5.3) for design in Fig. 5.6, where the performance-critical loop L_5 takes delayed sample. Loops L_1, L_2, L_3, and L_4 are performance non-critical, but comprise majority of the design in terms of area and power. Ideally, all loops should have the same loop latency and the same criticality from the performance perspective. Forcing an aggressive constraint on the performance-critical loop L_5 results in a sub-optimal implementation since the non-critical loops operate too fast and consume too much power. A delayed iteration can be considered in such a case, if possible from the algorithmic standpoint. This way, pipeline stages are in balance from the energy-delay standpoint. An ASIC implementation of this block is described in Chapters 8 and 9.

5.5 Iterative Square Rooting and Division

Square rooting and division are operations common to many wireless communication algorithms [52]. For example, space-time decoding [53] such as V-BLAST [54], QR decomposition [55] and the SVD-based algorithms [56], [57] often require vector normalization, which is in essence a sequence of square root and division operations.

Square root and division can be implemented in many different ways. Division algorithms have been extensively studied in the literature, and a taxonomy based upon hardware implementation is found in [58]. Square root algorithms were explored less frequently, yet there is an array of existing implementation architectures such as those using vectoring-mode CORDICs [59], iterative formulas, and look-up tables.

This section studies a particular case of adaptive algorithms with slowly-varying input argument, inspired by requirements in signal processing for indoor wireless channels. Among the algorithms for iterative square root and division, a method based on Newton-Rhapson formulas [60] is selected for the analysis due to its favorable convergence properties. Equations (5.4) and (5.5) describe inverse square root and division, respectively, where N is the input operand, x_s is the result of inverse square root operation, and x_d is the result of division operation.

$$1/sqrt(N){:}\quad x_s(k+1) = \frac{x_s(k)}{2}\cdot\left(3 - N\cdot x_s(k)^2\right), \quad x_s(k) \to \left.\frac{1}{\sqrt{N}}\right|_{k\to\infty} \tag{5.4}$$

$$1/N{:}\quad x_d(k+1) = x_d(k)\cdot\left(2 - N\cdot x_d(k)\right), \quad x_d(k) \to \left.\frac{1}{N}\right|_{k\to\infty} \tag{5.5}$$

Algorithms in Eqs. (5.4) and (5.5) can be analyzed as discrete systems, which have equilibrium points at 0, $\pm 1/\sqrt{N}$ ($1/sqrt$) and 0, $\pm 1/N$ (div). Using this result, analysis of convergence properties and required initial conditions is presented.

5.5.1 Error Dynamics

The analysis of error dynamics allows finding the number of iterations necessary to achieve a given degree of accuracy as well as the choice of proper initial condition. Both algorithms analyzed in this chapter converge quickly. This capability is desirable when dealing with large word-lengths since the number of bits resolved in each iteration is related to the speed of convergence. The algorithms achieve a quadratic convergence rate, which corresponds to two significant bits per iteration.

The convergence of the system described by Eqs. (5.4) and (5.5) can be studied by looking at an equivalent system given by Eq. (5.6), where normalization of x_s and x_d is done such that y_s and y_d converge to 1.

$$\begin{aligned} y_s(k) = \sqrt{N} \cdot x_s(k), \quad y_s(k+1) = \frac{y_s(k)}{2} \cdot \left(3 - y_s(k)^2\right), \quad y_s(k) \to 1 \big|_{k\to\infty} \\ y_d(k) = N \cdot x_d(k), \quad y_d(k+1) = y_d(k) \cdot (2 - y_d(k)), \quad y_d(k) \to 1 \big|_{k\to\infty} \end{aligned} \tag{5.6}$$

Figure 5.7 displays Eq. (5.6) graphically. Several regions define convergence of the *sqrt* algorithm (left plot): a) for $y_s(k) < \sqrt{3}$, the algorithm converges without sign changes, b) for $\sqrt{3} < y_s(k) < \sqrt{5}$, the algorithm converges with possible sign changes, and c) for $y_s(k) > \sqrt{5}$, the algorithm diverges. Similarly, the solid line indicates convergence while the dotted line corresponds to the divergent *div* algorithm (right plot). Square markers label equilibrium points. The desired operating point is $y_s = 1$.

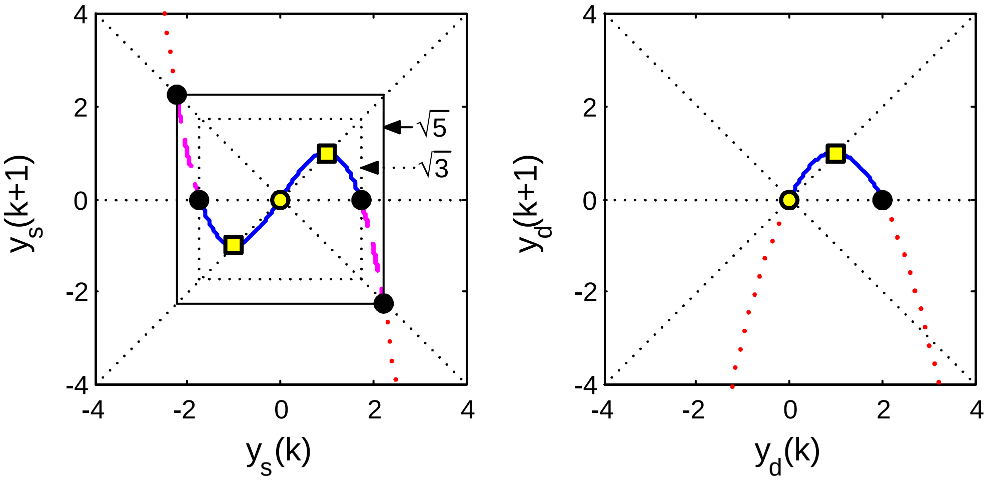


Figure 5.7: Convergence region of iterative square rooting and division. (solid line: convergence, dashed line: converging stripes, dotted line: divergence)

From Equation (5.6), the error dynamics are derived as given by Eq. (5.7).

$$\begin{aligned} e_s(k+1) &= -\frac{1}{2} \cdot e_s(k)^2 \cdot (3 + e_s(k)), \quad \text{where } e_s(k) = y_s(k) - 1 \\ e_d(k+1) &= -e_d(k)^2, \quad \text{where } e_d(k) = y_d(k) - 1 \end{aligned} \tag{5.7}$$

The result shows quadratic convergence of both algorithms. Besides fast convergence, getting to the solution point quickly also requires a good choice of initial condition.

5.5.2 Initial Condition

The algorithm has to be properly initialized first to guarantee convergence. Given an initial condition, it is further interesting to find out how many iterations are required to reach arbitrary level of accuracy. This section analyzes convergence properties of *sqrt* and *div* algorithms in terms of required accuracy and number of iterations.

Table 5.1: Convergence speed of iterative sqrt and div.

Target rel err (%)	0.1%	1%	5%	10%
$e_0 = 50\%$, # iter (s/d)	5 / 4	5 / 3	4 / 3	3 / 2
$e_0 = 25\%$, # iter (s/d)	3 / 3	3 / 2	2 / 2	2 / 1

For fast convergence, it is desirable to have time evolution with descending absolute error. This condition can be formulated as in Eq. (5.8), where $E(x_k)$ is an error term. The quadratic dependence is taken to ensure positive values of $E(x_k)$. The condition for descending absolute error is then given by $V(x_k)$.

$$\begin{aligned} &E(x_k) = (x_k - 1)^2 \\ &V(x_k) = E(x_{k+1}) - E(x_k) < 0, \quad \forall k, \quad k = 0, 1, 2, 3, ... \end{aligned} \tag{5.8}$$

Solving the inequality from Eq. (5.8) with respect to initial condition x_0 yields the solution given by Eq. (5.9). In the general case, one can set $V(x_0) < a$ and find a set S of feasible initial conditions, $S = \{x_0 : V(x_0) < a\}$.

$$\begin{aligned} &V_s(x_0) = \frac{x_0}{4} \cdot (x_0 - 1)^2 \cdot (x_0 + 1) \cdot (x_0^2 + x_0 - 4) \\ &V_d(x_0) = x_0 \cdot (x_0 - 1)^2 \cdot (x_0 - 2)^2 \end{aligned} \tag{5.9}$$

Figure 5.8 shows feasible initial conditions x_0 that satisfy conditions in Eq. (5.9). In the *sqrt* example the range of desirable x_0 is more restrictive than the convergence range, while in the *div* example it coincides with the convergence range.

The number of iterations required for convergence is a function of initial condition

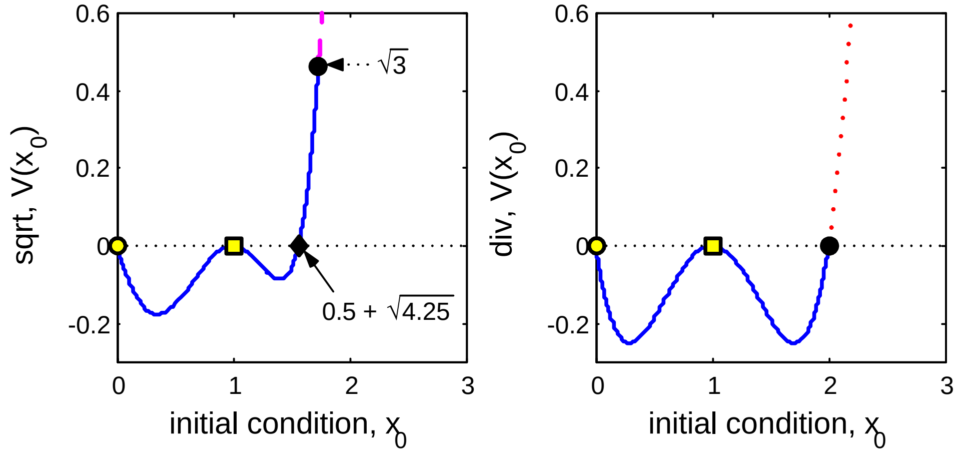


Figure 5.8: Choice of initial condition for decreasing absolute error. ($V < 0$: descending error, dotted line: divergence)

and error dynamics. Table 5.1 summarizes convergence properties for a 25% and 50% initial error, for varying accuracy of the final answer. Even for very large initial error of 50%, only five iterations are needed to achieve accuracy within 0.1%. The results in Table 5.1 also suggest that the error quickly decreases in every iteration. So, if the previous solution is taken as the initial condition for the next iteration, the answer is obtained in just one iteration, assuming a slow-varying input. This concept is illustrated on design examples in Chapters 8 and 9.

Chapter 6

Taxonomy of MIMO Algorithms

This chapter contains a survey of MIMO algorithms for wireless systems. Various conceptual algorithms form the basis for study of computational complexity of some common signal processing kernels and their suitability for hardware implementation. Due to the vast variety of MIMO algorithms available, classification begins with the fundamental diversity vs. spatial multiplexing trade-off in multiple-antenna channels [61], followed by a study of schemes that provide diversity and spatial multiplexing gains. The chapter concludes with an overview of some VLSI realizations.

6.1 Diversity vs. Spatial Multiplexing

Fundamentally, a MIMO system can improve the reliability of a wireless link through increased diversity or improve the channel capacity through spatial multiplexing. Given a MIMO channel, both gains can be simultaneously achieved, but there is a fundamental trade-off between the achievable gains for each type. Zheng and Tse introduced this trade-off in [61] and evaluated existing MIMO schemes with respect to the optimal trade-off curve. The basic definitions are repeated here, followed by investigation of signal processing requirements found in MIMO algorithms.

6.1.1 Diversity Gain

Multiple antennas have traditionally been used as a method to overcome channel fading through increased diversity [54]. The diversity gain can be achieved using multiple transmit or receive antennas that provide multiple signal paths between the transmitter and the receiver. By sending the same information over different paths and averaging over multiple independently-faded replicas at the receiver, more reliable communication is achieved. For example, if the transmitted signal is passed through n different paths, which correspond to n transmit antennas, the average error probability decays as $1/SNR^n$ as opposed to $1/SNR$ for the single-antenna case. Under these conditions, when the error probability decays as $1/SNR^n$, the diversity gain is equal to n [52].

Furthermore, having both multiple transmit and multiple receive antennas improves the link reliability by essentially averaging over multiple independently-faded paths. For a system with m transmit and n receive antennas, the maximum achievable diversity gain is mn. The diversity gains help overcome channel fading, but fading is not necessarily always detrimental. In fact, fading can be helpful if different paths that fade independently combined constructively, exploiting spatial multiplexing gains.

6.1.2 Spatial Multiplexing Gain

Spatial multiplexing is a method that takes advantage of fading by increasing the degrees of freedom available for communication. This is possible under the assumption that path gains between transmit-receive antenna pairs fade independently. In this case, the channel matrix is well conditioned, so different Tx-Rx antenna pairs can be essentially viewed as independent spatial channels. Sending independent data streams over these spatial channels improves throughput proportionally to the number of independent spatial sub-channels.

The spatial multiplexing gain can be quantified with the channel capacity increase. Foschini [54] showed that for a $m \times n$ system with independent identically distributed (i.i.d.) Rayleigh faded gains between individual antenna pairs in a high-SNR regime, the capacity of the channel increases linearly with the number of spatial channels as given by Eq. (6.1).

$$C(SNR) = \min\{m, n\} \cdot \log(SNR) + O(1) \tag{6.1}$$

The number of available spatial channels is the minimum between the number of transmit antennas m and the number of receive antennas n.

6.1.3 Optimal Trade-Off Curve

Optimizing for spatial multiplexing alone typically results in reduction of diversity gain and vice versa. Zheng and Tse analytically formulate this trade-off in [61]. For an $m \times n$ system, they shown that, for a slow-fading environment with random channel gain that is constant over l symbols ($l \geq m + n - 1$), the optimal diversity gain $d^*(r)$ is given by Eq. (6.2).

$$d^*(r) = (m - r)(n - r) \tag{6.2}$$

where r is the achieved multiplexing gain, m is the number of transmit antennas and n is the number of receive antennas.

This trade-off tells us that r antenna pairs provide spatial multiplexing gain, while the remaining $(m - r)(n - r)$ paths provide the diversity gain. In other words, this is a trade-off between the error probability and the data rate. The error probability decays as $1/SNR^{d^*(r)}$ with channel capacity proportional to $r\log(SNR)$.

The optimal trade-off curve can be used an effective tool for evaluation of MIMO algorithms. Diversity and spatial-multiplexing algorithms as shown in Fig. 6.1 are analyzed next.

6.2 Diversity Algorithms

Diversity gain can be achieved using the general principle of space-time coding. A simple way to maximize diversity is to use orthogonal designs, such as a simple rep-

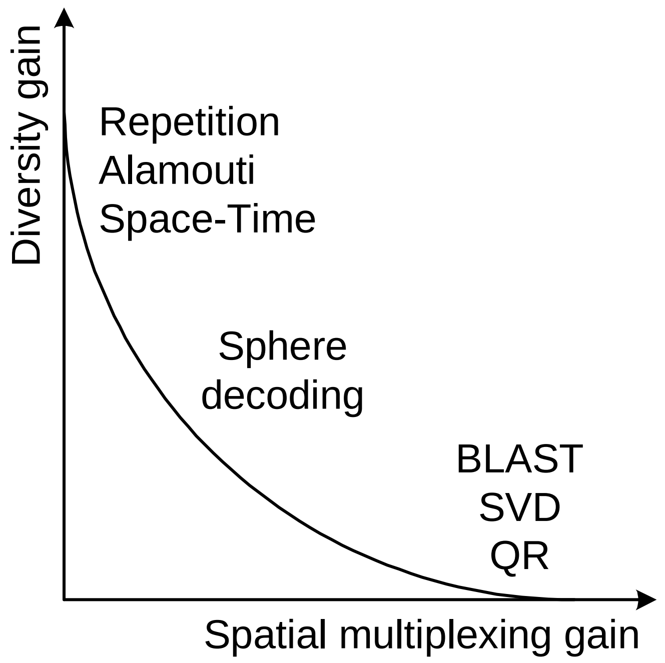


Figure 6.1: Classification of MIMO algorithms in the diversity-multiplexing plane.

etition scheme where the same symbol is repeated over multiple antennas or the Alamouti scheme [62] which relies on the transmission of two data symbols. Orthogonal schemes closely approximate the optimal curve in the low spatial multiplexing region, but they become increasingly sub-optimal for higher r. Space-time coding proposed by Tarokh [63] achieves an improved multiplexing gain. Due to low spatial multiplexing gains, these schemes are not suitable for high data rate links.

6.2.1 Repetition Scheme

A repetition scheme is the simplest way to achieve diversity gain. This can be understood by looking at a simple 2×2 system. Such a system has four Tx-Rx paths, which need to be combined for maximal diversity gain d^*_{max}. This is accomplished with a transmission scheme described in Eq. (6.3).

$$\boldsymbol{X} = \begin{bmatrix} x_1 & 0 \\ 0 & x_1 \end{bmatrix} \tag{6.3}$$

where x_1 is the transmitted symbol, rows of $\boldsymbol{X}$ correspond to time and columns correspond to antennas. This way, the same symbol is repeated over the transmit antennas in consecutive symbol time slots. Since one symbol is transmitted in two symbol times, spatial multiplexing gain is 1/2.

6.2.2 Alamouti Scheme

The Alamouti scheme achieves the same diversity gain as the repetition scheme, but provides increased throughput, since two information symbols are transmitted in two symbol times. This is made possible with a transmission scheme in Eq. (6.4).

$$\boldsymbol{X} = \begin{bmatrix} x_1 & -x_2^\dagger \\ x_2 & x_1^\dagger \end{bmatrix} \tag{6.4}$$

where columns correspond to transmit antennas. In terms of diversity, the Alamouti scheme with two transmit and one receive antenna is equivalent to a maximum ratio receiver combining (MRRC) scheme with one transmit and two receive antennas.

For a 2×2 system, for example, the Alamouti scheme still achieves maximal diversity gain of $d^*_{max}(0) = 4$, but results in sub-optimal spatial multiplexing gain for $r > 0$. Since only two information symbols can be transmitted in two symbol periods, the best achievable multiplexing gain is $r = 1$, which is twice as much as the repetition coding scheme. While achieving the maximal diversity, both repetition and Alamouti codes achieve sub-optimal diversity-multiplexing trade-off due to their inability to maximize the spatial multiplexing gain.

6.2.3 Space-Time Coding

Orthogonal designs based on repetition coding or the Alamouti scheme are not optimized in terms of data rate. Tarokh has shown [63] that with optimal coding, a 2×2 system can achieve full multiplexing gain of $r = 2$, which corresponds to sending two data symbols over two antennas in each symbol period. The same study shows, however, that systems with more than two transmit antennas cannot achieve the full transmission rate of symbol per symbol period. Therefore, the potential of a MIMO channel to support extra degrees of freedom in a form of spatial multiplexing is not fully exploited by the space-time coded orthogonal designs.

6.3 Spatial Multiplexing Algorithms

Spatial multiplexing algorithms primarily focus on exploiting multiple degrees of freedom in MIMO channels to maximize data throughput. Unlike diversity schemes which can work with single receive antenna, spatial multiplexing requires both multiple transmit and multiple receive antennas (MIMO). This section summarizes basic concepts most commonly encountered in various MIMO algorithms, including BLAST, QR decomposition, and SVD based channel estimation.

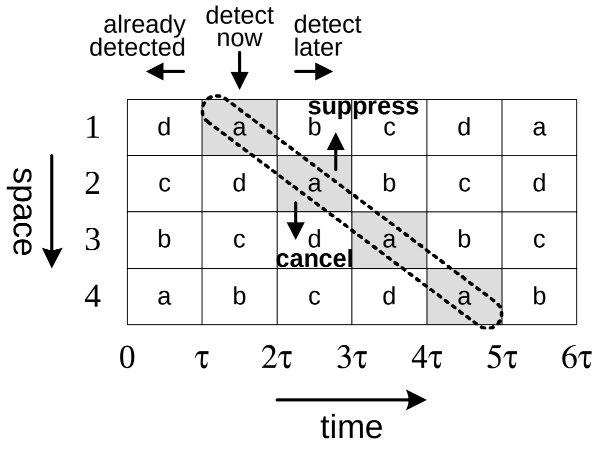
6.3.1 BLAST Algorithms

The pioneering work in MIMO decoding [54], [64] by Foschini and researchers from Bell Labs resulted in formulation of the Bell Labs Layered Space Time (BLAST) algorithms for MIMO communication. In BLAST, codewords are arranged in a space-time grid. The data is encoded over multiple transmit antennas and distributed in time. Depending on how the data is organized in the space-time grid, two algorithms can be distinguished: diagonal (D-BLAST) and vertical (V-BLAST).

D-BLAST

The diagonally layered BLAST (D-BLAST) architecture was proposed by Foschini [54] as a point-to-point communication architecture that uses an equal number of antennas at both the transmitter and the receiver. The architecture is designed for a Rayleigh fading environment where the transmitter does not have knowledge of the channel.

D-BLAST is a space-time coding scheme. Codewords are partitioned in space and time as shown in Fig. 6.2 for the case of a 4×4 MIMO system. Each codeword, a for example, is strategically distributed over space (antennas) and time. The time



D-BLAST decoding scheme

demod a wefeform on:	avoid interference from antenna:
$\tau < t < 2\tau$	–
$2\tau < t < 3\tau$	1
$3\tau < t < 4\tau$	1, 2
$4\tau < t < 5\tau$	1, 2, 3
⋮	

Figure 6.2: D-BLAST decoding scheme. Each codeword is partitioned into four blocks and distributed over time and space.

interval τ contains N symbols.

A training phase precedes the decoding shown in Fig. 6.2. During training, known signals are transmitted and processed at the receiver to initialize the decoding algorithm. The decoding algorithm is based on interference suppression. In time interval $\tau < t < 2\tau$, detection of space-time layer a begins. The layers below a which have already been detected are subtracted from the received waveform. At the receiver, the first block of layer a is estimated through maximum ratio combining. In the time interval $2\tau < t < 3\tau$, layer b (that is above a) is treated as interference to be suppressed. So far, codeword a sees two blocks corresponding to spatial sub-channels 1 and 2. Moving forward in time, during interval $4\tau < t < 5\tau$, layer a can be successfully decoded if the condition in Eq. (6.5) holds.

$$\sum_{i=1}^{4} (1 + SINR_i) \geq R \tag{6.5}$$

In Eq. (6.5), $SINR_i$ is the signal-to-interference-and-noise ratio of the MMSE demodulator at the i^{th} stage of the cancellation, and R is the target data rate. In each time interval, once a partial codeword is decoded, it is subtracted from the received symbol, and the detection process moves forward in time.

In D-BLAST, the detection of each layer is based on successful detection of the underlying layers. Consequently, failure in a layer will likely cause the detection of all subsequent layers to fail. Additionally, the D-BLAST scheme suffers from a rate loss because some of the antennas are not utilized. For a large number of antennas, however, the rate loss gets amortized. Compared to the Alamouti scheme, the signal processing in D-BLAST is more complex. Unlike the Alamouti scheme, however, D-BLAST does generalize to an arbitrary number or transmit and receive antennas.

V-BLAST

In V-BLAST, each coded stream, or layer, extends horizontally in the space-time grid and is stacked vertically. Such vertical organization eliminates boundary waste of D-BLAST at the start and end of a burst.

A V-BLAST communication link is illustrated in Fig. 6.3. Transmit data streams have different data rates, in general. In case the transmitter knows the channel, it can allocate different amounts of power in the different eigen-modes depending on their strengths. When the transmitter does not know the channel and the channel is random, equal amounts of power are allocated across the eigen-modes. A detailed

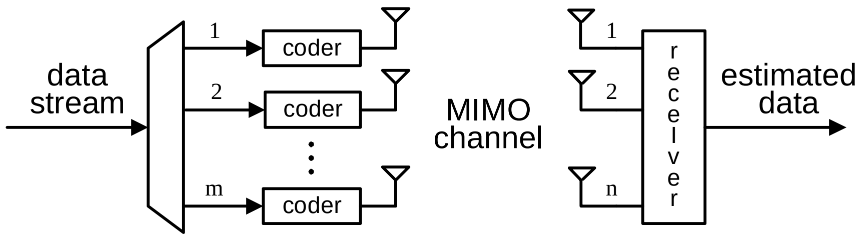


Figure 6.3: Vertical BLAST MIMO communication link.

derivation of this result can be found in [52]. The receiver performs a joint maximum likelihood (ML) decoding of data streams.

The spatial processing at the receiver [64] consists of three steps: a) interference cancellation, b) interference nulling/suppression, and c) compensation. Assume that the receiver has detected the first $i-1$ symbols. During the interference cancellation step, interference from already detected symbols is subtracted out. In the interference nulling step, interference from $m-i$ yet to be detected sub-streams is removed. This is achieved by projecting the result of the first step onto the $m-i$ dimensional subspace spanned by the vectors of the matrix channel response, using the Grahm Schmidt process. Finally, in the compensation phase, the error probability is minimized by optimizing the order of symbol detection such that symbol elements with the highest SNR are detected first. The element with the best decision statistic is moved to the bottom of the stack, and the other sub-streams are shifted up.

Compared to D-BLAST, V-BLAST employs simplified processing without coding across the transmit antennas. In the absence of coding, an outage occurs whenever one of the sub-channels is in a deep fade and cannot support the rate of the stream

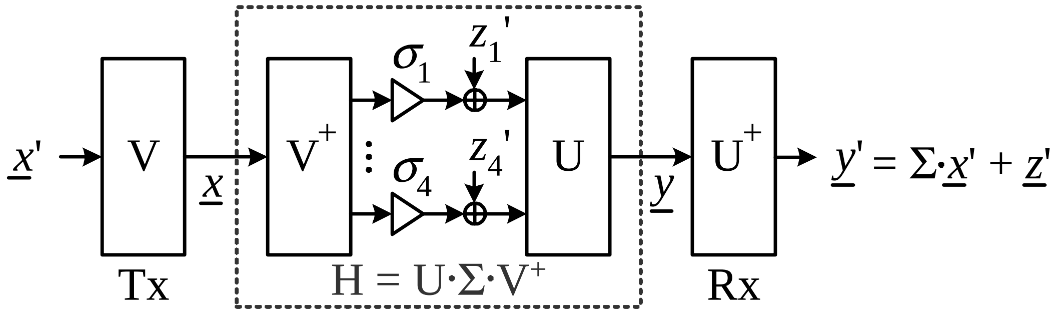


Figure 6.4: Decoupling of MIMO channel through SVD.

using that sub-channel. D-BLAST, on the other hand, has a higher communication efficiency due to encoding, but is also more computationally complex. Zheng and Tse have shown that both algorithms achieve the same spatial multiplexing gain, while D-BLAST delivers higher diversity gains. This makes sense, since fade on any transmit antenna affects only a fraction of the codewords layered in space and time.

6.3.2 SVD

Singular value decomposition (SVD) [65] [66] [67] of a wireless channel relies on partial channel knowledge at the transmitter to extract spatial multiplexing gains. Decoding of a MIMO channel requires matrix inversion which can be done in block-form using an SVD as shown in Fig. 6.4. SVD is a composition of three operations: a rotation ($\boldsymbol{U}$), a scaling operation ($\boldsymbol{\Sigma}$), and another rotation ($\boldsymbol{V}^\dagger$). $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary and $\boldsymbol{\Sigma}$ is a diagonal matrix.

When the channel matrix $\boldsymbol{H}$ is partially known to the transmitter, the optimal strategy is to transmit independent streams in the directions of the eigenvectors of

$\boldsymbol{H}^{\dagger}\boldsymbol{H}$ [52]. Using partial channel knowledge $\boldsymbol{V}$ at the transmitter (Tx) and projecting the received vector $\underline{y}$ onto space of $\boldsymbol{U}^{\dagger}$ at the receiver (Rx), the channel can be effectively orthogonalized, looking between $\underline{x}$' and $\underline{y}$'. The independent data streams are then sent across Tx antennas. This is achieved by pre-rotating the data so that the parallel streams can be sent along the eigen-modes of the channel. At the receiver, data streams arrive orthogonally without interference between the streams.

6.3.3 QR Decomposition

The QR decomposition of a matrix $\boldsymbol{H}$ is a factorization $\boldsymbol{H} = \boldsymbol{QR}$, where $\boldsymbol{Q}$ is a unitary matrix and $\boldsymbol{R}$ is an upper triangular matrix. The QR decomposition is commonly used in various signal processing applications such as beamforming, [68]. Traditionally, the eigen-value decomposition (EVD) of the sample decorrelation matrix or the singular value decomposition (SVD) of the data matrix have been used to compute subspaces in various beam-forming algorithms, [69]. The QR decomposition is used as an alternative since it is more computationally efficient than EVD or SVD.

In the QR decomposition, matrix $\boldsymbol{R}$ is upper triangular. The SVD is difficult to update due to relatively large computational complexity. Signal processing methods that rely on subspace projections generally do not require all of the information provided by the SVD. For example, beamforming requires the eigenstructure information contained in the submatrix $\boldsymbol{\Sigma}$, but this information does not have to be in the form of a diagonal matrix. This gives some leeway in the matrix factorization. Decoding of

$\underline{y} = \boldsymbol{QR} \cdot \underline{x}$ reduces to solving a linear system using simple Gaussian elimination due to the triangular nature of matrix $\boldsymbol{R}$. A comprehensive study of subspace tracking algorithms can be found in [55], [69].

Algorithms that maximize diversity or spatial multiplexing have been well explored and documented in literature. Pure spatial multiplexing or diversity gives limited benefit in the other dimension. Recently, there have been some efforts toward unification in a diversity-multiplexing trade-off sense, [70], [71], [72], [73].

6.4 Unified Algorithms

The optimum decoding method for MIMO channels is maximum likelihood (ML) where the receiver compares all possible combinations of symbols which could have been transmitted with what is observed. The complexity of ML decoding is very high, especially when many antennas are used, [74]. To meet the complexity requirements, lattice decoders have recently gained attention, [70]. This decoding strategy is also known as sphere decoding.

6.4.1 Sphere Decoding

The main challenge of receiver design for MIMO systems is in the non-orthogonality of the channel and superposition of the signals arriving from all transmit antennas. Optimal ML detection requires finding the signal point $\underline{\hat{x}}$ of the transmitter vector signal set that minimizes the Euclidean distance with respect to the received vector $\underline{y}$. This can be equivalently formulated as the problem of finding the closest lattice point in a transformed vector space described by Eq. (6.6).

$$\underline{\hat{x}} = argmin||\underline{y} - \boldsymbol{H}\underline{x}||^2 \tag{6.6}$$

This problem is exponential in the number of possible constellation points, making ML detection very difficult for practical realization. To avoid the exponential complexity of the ML detection problem, the idea is to restrict the search for the closest lattice point in a way that includes only vector constellation points that fall withing a certain search sphere, [75], [76], [77]. This approach allows for finding the ML solution with polynomial complexity.

In its search for the ML solution, the sphere detector evaluates all transmit vector signals x fulfilling the condition in Eq. (6.7)

$$||\underline{y} - \boldsymbol{H}\underline{x}||^2 < SR^2 \tag{6.7}$$

where SR is the search radius of the sphere. The choice of SR influences the complexity of the algorithm. When SR is too large, it leads to a sphere containing a very high number of hypotheses and thus high detection complexity. When SR is too small, this may result in an empty sphere and the search has to be restarted with an increased radius. In searching for SR, decompositions of the channel matrix can be used [78], such as QR decomposition already described.

$$\begin{aligned} ||\boldsymbol{Q}^{\dagger}\underline{y} - \boldsymbol{R}\underline{x}||^2 &< SR^2 \\ ||\underline{y}' - \boldsymbol{R}\underline{x}||^2 &< SR^2 \end{aligned} \tag{6.8}$$

Equation (6.8) is an equivalent form of Eq. (6.7). Using the triangular nature of matrix R, the detection process starts from the last transmit antenna and works its way up until the first antenna is detected. Zimmermann et al. [78] did an extensive study of various modifications of this sphere decoding algorithm and their complexities. They determined that the complexity of sphere decoding increases quadratically with the number of Tx-Rx antenna pairs. Gamal et al. presented a study of lattice coding and decoding schemes [72] for delay-limited MIMO channels in terms of achieving the optimal diversity-multiplexing trade-off. Sphere decoding is a relatively new research area and major effort is now being directed towards developing low-complexity variants of the generalized minimum Euclidian distance lattice decoder.

6.5 Survey of VLSI Implementations

Implementation of MIMO based communication links is an active area of research. This section studies some existing ASIC realizations of MIMO decoders in order to establish a reference in terms of power, speed, and area.

Table 6.1: Summary of ASIC Implementations of 4×4 MIMO Decoders.

Year, [Reference]	2005, [79]	2005, [80]	2004, [81], [82]	2003, [83], [82]
Modulation	16-QAM	QPSK	16-QAM	QPSK
Detection	depth-free sphere	ML-APP	K-best sphere	V-BLAST
Bandwidth	14 MHz*	5 MHz	4 MHz*	26 MHz*
Technology	$0.25\mu m$	$0.18\mu m$	$0.35\mu m$	$0.35\mu m$
Clock frequency	71 MHz	123 MHz	100 MHz	80 MHz
Gate count	50k + preproc.	685k	91k + preproc.	190k
Throughput	169 Mbps	28.8 Mbps	52 Mbps	160 Mbps
Power	473 mW (2.5V)	N/A	626 mW (2.8V)	608 mW (2.7V)
E_{100k} (90nm)	768 pJ (1V)	N/A	162 pJ (0.85V)	94 pJ (0.81V)
Spec.Eff (bps/Hz)	12.07	5.76	13	6.15
FOM (pJ/bps/Hz)	63.6	N/A	12.5	15.28

Table 6.1 summarizes features of some existing 4×4 MIMO implementations. Due to the wide variety of algorithms implemented over multiple technology generations, it is most meaningful to study power/energy efficiency normalized to the same technology. The reported numbers are thus normalized to a 90nm process with nominal supply of 1V. For implementations with scaled voltage (columns 3 and 4), the power is normalized to a 90nm technology with accordingly scaled V_{dd}. To account for area differences in terms of gate count, the power is further normalized to an equivalent 100k gate power. Finally, a 100k gate equivalent energy, E_{100k}, is obtained to eliminate the clock frequency. The V-BLAST design (column 4) turns out to be the most energy efficient, taking 94pJ per 100k gates. This comparison should be taken cautiously since no information about algorithm complexity is available. Due to the lack of information about GOPS, normalization is done with respect to area.

Another interesting parameter is the spectral efficiency, $Spec.Eff$, obtained as the ratio between the throughput and available communication bandwidth. Paper [80] reports communication bandwidth (second column in the table). For other papers that did not explicitly state the bandwidth, the bandwidth is estimated from the decoding throughput and modulation scheme, assuming the spatial gain of 3 in a 4×4 system. For fair comparison, E_{100k} is normalized by $Spec.Eff$ to obtain a figure of merit, FOM in the table. Although the V-BLAST design is the most energy-efficient, the K-best sphere decoder has the highest overall efficiency in terms of 12.5 pJ/bps/Hz. This number is used as a reference to evaluate the results in Chapter 9.

Chapter 7

CAD Methodology and Flow

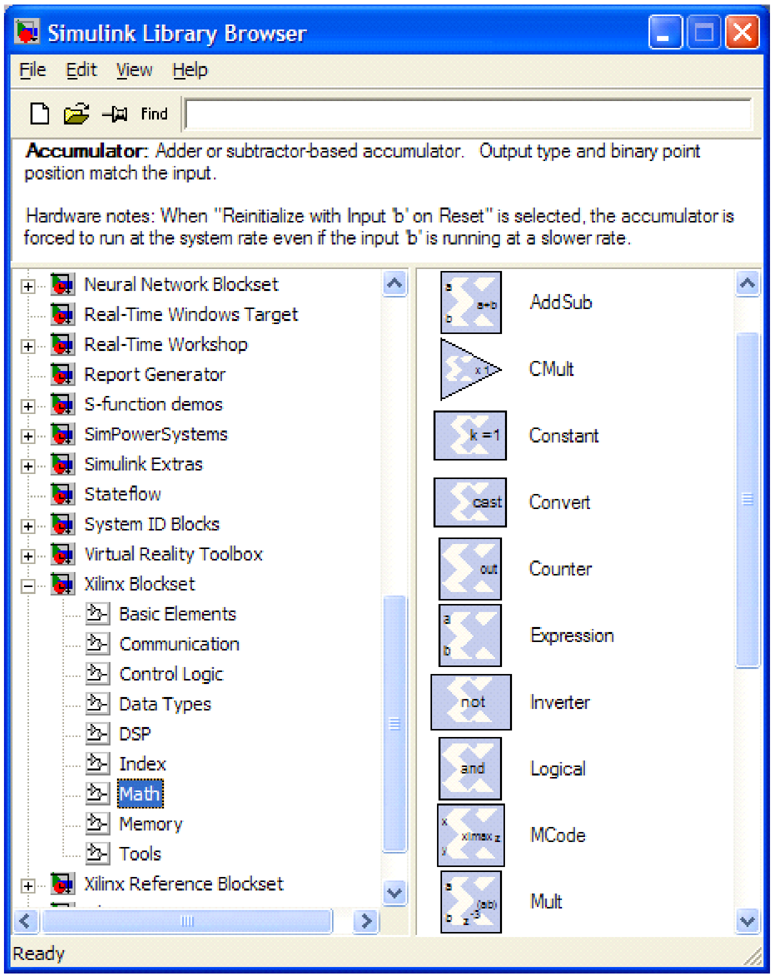
This chapter presents a design methodology that incorporates the results from previous chapters: circuit and micro-architectural optimizations as well as signal processing techniques. As an integral part of the design methodology, key design procedures are integrated into a commercial CAD environment. The idea is to bring together the algorithms, circuits, and VLSI architectures that create the most energy and area efficient implementations in a highly automated fashion.

The design procedure starts with design entry in the widely adopted, graphical Simulink environment to create bit-true, cycle-accurate models using hardware equivalent blocks. Architectural choices are made early in the design process, at the Simulink level, using architectural feedback from logic synthesis. Proper characterization of Simulink blocks for power, speed, and area at the circuit level allows for early estimates of the algorithm implementation. Upon word-size and register retiming optimizations, the physical design is done using standard industry tools. This step is highly automated: a chip with complexity of 0.5M gates can be synthesized in one day (excluding verification steps). An FPGA-based chip testing that uses an additional FPGA board is described. The testing conveniently interfaces with Simulink environment, providing a nice framework to unify the ASIC and FPGA flows.

7.1 Simulink Design Environment

Design re-entry is standard practice today. A design is entered in various forms by different engineering teams, resulting in heterogeneous design descriptions. Algorithm developers tend to work in a Matlab environment, which has an array of built-in functions convenient for quick algorithm modeling and verification. Software code like systemC [84] is another sequential processing entry, which requires more sophisticated programming skills. Still, neither of the representations captures the architecture and information about sampling rate. The architectural description is then created separately by hardware designers who have to completely re-enter the design in hardware description language (HDL). Design re-entry presents an enormous overhead, which can be avoided by using a unified Simulink design environment.

The Simulink design environment is convenient, because it provides an intuitive graphical description for both algorithm developers and chip designers. An algorithm is entered only once is a graphical block form, providing timed data-flow representation of the design and abstract view of design architecture. With technology-specific



Accumulator: Adder or subtractor-based accumulator. Output type and binary point position match the input.

Hardware notes: When "Reinitialize with Input 'b' on Reset" is selected, the accumulator is forced to run at the system rate even if the input 'b' is running at a slower rate.

Figure 7.1: Simulink environment illustrating XSG block library.

data for speed, power, and area of functional blocks, algorithm designers can explore the implementation space while remaining in the Simulink environment to verify the algorithm. Hardware designers can use the Simulink description to generate an HDL description automatically and also be able to verify the results of hardware emulation using Simulink test vectors.

In this work, Simulink is used in the full design cycle, starting with algorithm modeling, through design optimizations, and final ASIC verification.

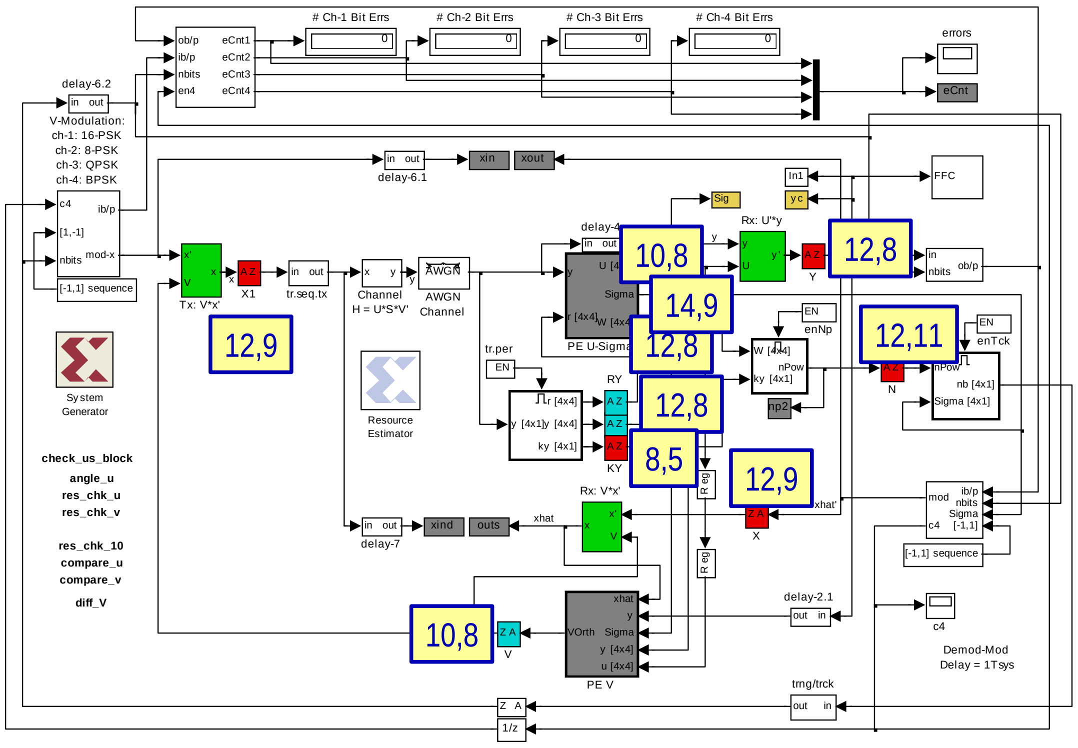


Figure 7.2: Example of Simulink design entry: model of a MIMO communication. Numbers indicate total and fractional bits.

7.1.1 Design Description

The Matlab/Simulink environment is widely used for evaluation of signal processing algorithms because of its widespread use by the algorithmic community. Systems can be quickly modeled using components from the basic floating-point library, which enables cycle-accurate functional verification with ideal, infinite word size. Taking one step further to work with the Xilinx System Generator (XSG) [85] block-set enables the creation of a hardware-equivalent model suitable for implementation. In the XSG library, there are readily available basic arithmetic operators, as shown in Fig. 7.1, that capture hardware parameters, such as word-size and latency.

Fig. 7.2 illustrates an example of a Simulink design description. It is a timed data-flow model of the design showing the block-level representation of a multiple-antenna communication. The model in Fig. 7.2 allows algorithmic exploration in a realistic, closed-loop environment. Each functional block can be described with C or Matlab S-functions (software), or with timed data-flow hardware blocks. This flexibility also allows for comparison of ideal floating-point behavior against the fixed-point realization of the algorithm.

7.1.2 Functional Verification

The bit-true, cycle-accurate behavior of a design is obtained at the Simulink level. Mixed simulation can be performed to compare floating and fixed point behavior, before mapping the fixed point design onto hardware for emulation.

This example illustrates that functional verification can be done in a simple way, with an intuitive graphical display of results. The example in Fig. 7.3 illustrates the result of eigen-mode decomposition done by the multi-antenna algorithm from Fig. 7.2. Plots in Fig. 7.3 show an adaptive MIMO channel decoupling (top) and signal constellations corresponding to different spatial channels/antennas (bottom). Analysis and design of the MIMO example will be presented in detail in Chapter 8.

To speed up verification process, the fixed-point design can be directly mapped to an FPGA without the need to simulate the design in Simulink. Instead, the design can be emulated much faster on real hardware, while remaining in the Simulink

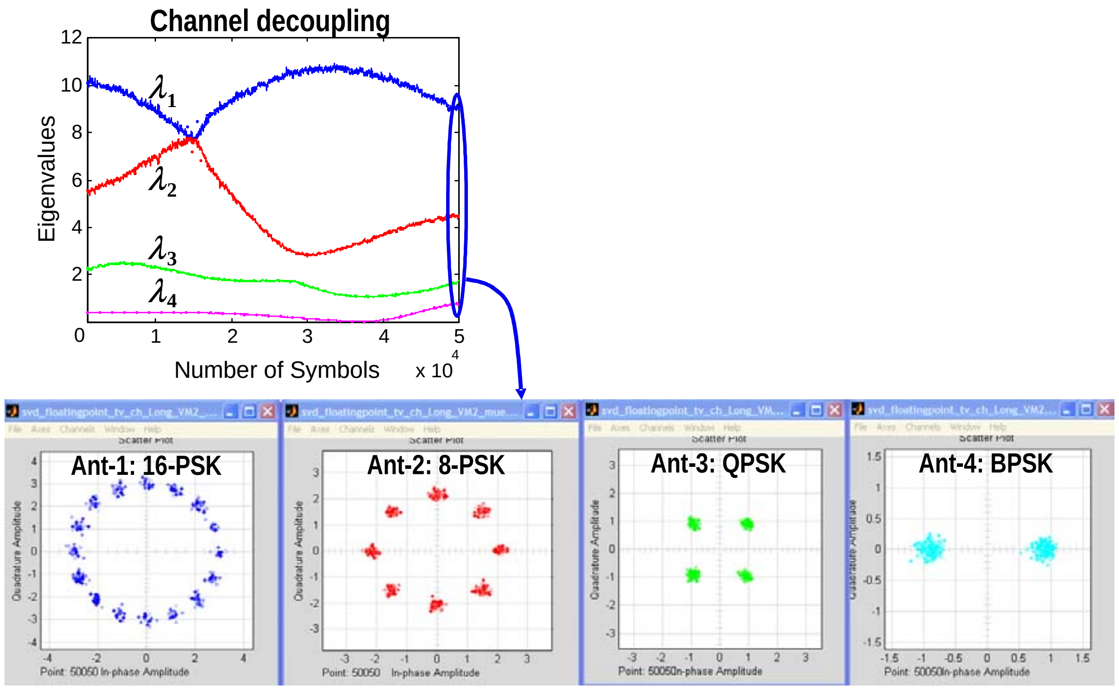
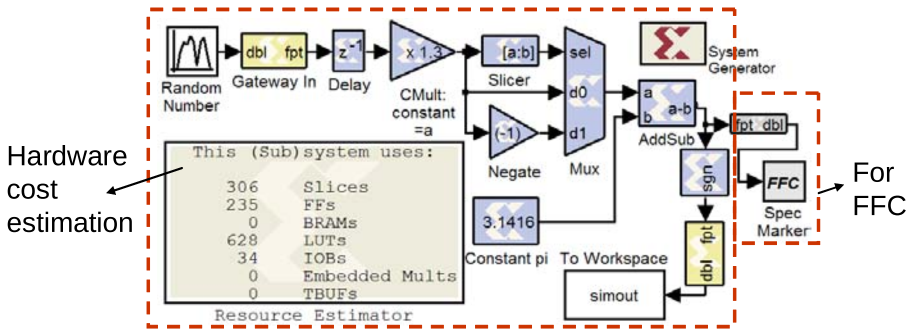


Figure 7.3: Example of functional verification in Simulink.

environment to collect the emulation data. Once bit-true cycle-accurate behavior is obtained, the first step in the overall design optimization procedure is to determine the optimal word-lengths that minimize area of the design. This is done using an in-house tool for word-length reduction.

7.1.3 Word-length Optimization

Word-length optimization is carried out in Simulink, using the floating-to-fixed point conversion (FFC) tool developed by Changchun Shi, [1]. The FFC minimizes hardware cost (FPGA area utilization) subject to user defined performance measures, such as MSE error due to quantization. In essence, the tool determines integer word-



This (Sub)system uses:
```
306   Slices
235   FFs
  0   BRAMs
628   LUTs
 34   IOBs
  0   Embedded Mults
  0   TBUFs
```

Figure 7.4: Illustration of FFC-enhanced Simulink model, [1].

lengths by node profiling and range detection, while fractional word-size optimization is determined by perturbation theory, [86], [1].

A typical FFC environment is illustrated in Fig. 7.4. An FFC spec marker defines MSE spec for a desired node in the system, while hardware cost estimation blocks summarizes FPGA hardware utilization in terms of FPGA slices, flip-flops, look-up tables, etc. After the MSE constraints have been formulated, the FFC tool runs a number of simulations to calculate word-size sensitivities and applies perturbation theory to find the optimal result.

The limitations of the FFC tool are: a) input data dependency and b) simulation based optimization. First, the node profiling is done based on input data range, so representative data has to be provided. To address this issue, the tool can include some extra guard bits in order to avoid long simulation times. Additionally, the designs of large complexity can take very long time to simulate even for input data sets of small size. Specifically related to the example in Fig. 7.2, the tool has been upgraded to support cases with user-defined word-sizes and their respective nodes.

This is then used for hierarchical optimization in which word-lengths of primary inputs and outputs are fixed. A detailed description of the tool can be found in [1] along with numerous design examples and also a user tutorial.

After successful functional verification, the design has to be refined at the architectural level in order to best utilize the underlying technology in terms of power, area, and speed. Extensive circuit-level characterization of building blocks from the XSG library is key to choosing the optimal architecture for the implementation. Efficient algorithm implementation is then possible without requiring much top-level post synthesis information to Simulink.

7.2 Characterization Methodology

The goal of circuit-level characterization is to augment the XSG block-set with technology-dependent information such as speed, power, and area. The complexity of the basic and most commonly used library blocks is very low (granularity of add, multiply, shift, mux, register etc.), so full characterization over a range of latency and word-size parameters is possible.

The approach for block-level characterization is illustrated in Fig. 7.5 for cases of 16-bit addition and multiplication. These two blocks differ in logic gate complexity, which translates into different latency of their implementations, for equal cycle time. Since cycle time is common for the entire top-level design, the blocks are characterized for speed on the latency versus cycle time plot shown in Fig. 7.5. Each point along

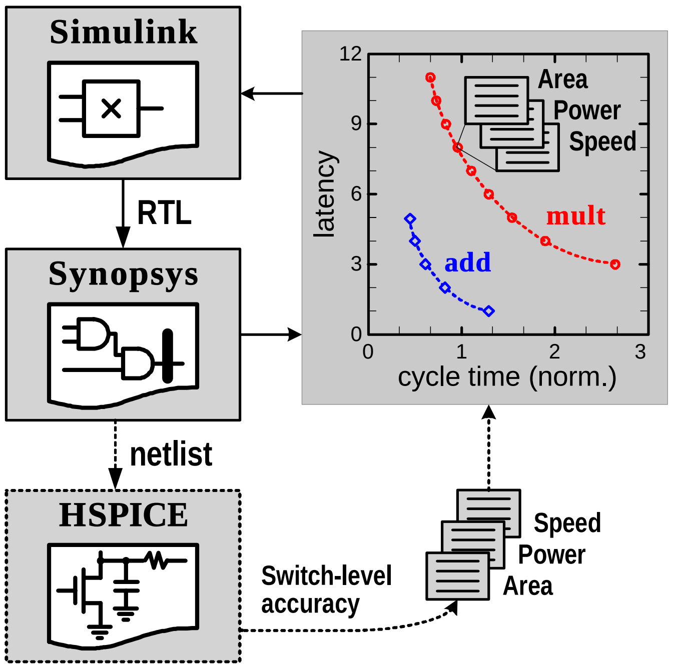


Figure 7.5: Block-level characterization (*mult*, *add* examples).

the curve is also characterized for power and area. Equal cycle time implies equal complexity (i.e. logic depth) of pipelined logic. This approach enables hierarchical expansion around the simple block-set.

7.2.1 Power Estimation

Estimates for power can be taken from physical synthesis or further refined with switch-level accuracy. The switch-level characterization can be very time consuming. For example, Eldo simulation over several clock cycles of a 16-bit multiplier with a fully distributed *RC* interconnect model took 19 days! The results of switch-level simulator can result in 2× higher power than that from synthesis, with the difference depending on the accuracy of the wire load models used in synthesis and extraction.

Refining the power estimate can be time-consuming for another reason: the power also depends on the switching activity of the block inputs and the corresponding activity profile inside the block. An accurate calibration requires long simulations over a large number of cycles, which can be managed with extensive characterization at few points and extrapolation of the results. Still, no matter how accurate characterization results are, the numbers may change when the pre-characterized blocks are placed in a different environment. Usually, synthesis estimates with default activity are fairly realistic since, to a first order, the effects of overestimating the activity factors and underestimating the parasitics nearly cancel out.

7.2.2 Area Estimation

The area of a design can be characterized in terms of FPGA resource utilization or silicon area. Simulink/XSG provides area estimates in terms of FPGA resources (slices, flip-flops, lookup tables, etc.) while the ASIC flow gives estimates in terms of physical silicon area. To link the two, characterization of several blocks revealed the following observation: 10,000 FPGA slices = $1mm^2$ of layout area in a 90nm CMOS technology (with $\sim 80\%$ layout density). This number is obtained from linear extrapolation of area dependency vs. the number of slices and silicon area for three examples that are about an order of magnitude apart from each other in terms of complexity: adder, multiplier, and inverse square root circuits. This way, an early estimate of chip area is provided at the Simulink level.

It is common practice in IC design to normalize silicon area by the area of a gate from standard cell library (e.g. 2-input NAND). At higher levels of granularity, the area can be normalized to an atomic operation, such as *add* or *multiply-accumulate*.

7.3 "Chip-in-a-Day" Design Flow

An automated "chip-in-a-day" design flow is illustrated in Fig. 7.6. The flow starts with design description in Simulink using the XSG block-set. The Simulink description of the block interconnect is used to generate the hardware description for mapping the design onto a FPGA array. The same description is also used to create another form of hardware description for ASIC place and route tools.

Making correct architectural decisions in Simulink relies on architectural feedback from synthesis. This is needed as early in the flow as possible to avoid unnecessary iterations. First estimates for area, speed, and power are fed back after initial logic synthesis, Fig. 7.6. These results can be refined, if desired, by more accurate estimates after physical layout synthesis. Efficient algorithm implementation is possible without requiring much post-synthesis information to Simulink, assuming that extensive characterization of basic building blocks is done as described in previous section.

VLSI implementations rely on low-level blocks from the hardware Simulink library that can be readily synthesized by commercial back-end synthesis tools. The block-set includes add, multiply, shift, mux, register, counter, etc. Logic synthesis is performed based on the behavioral HDL description provided by the INSECTA tool [87].

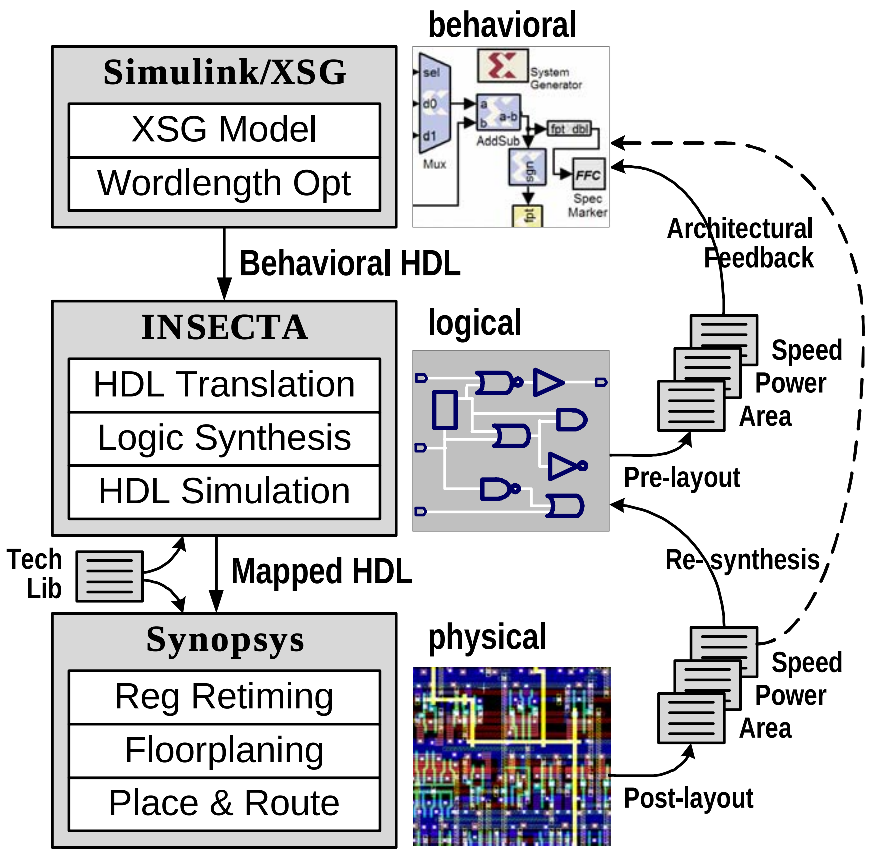


Figure 7.6: Simulink based "Chip-in-a-Day" design flow.

7.3.1 INSECTA

INSECTA is an in-house tool which performs HDL translation, first synthesis, and functional verification through HDL simulation. Each of the steps is described below.

HDL Translation

INSECTA begins by producing HDL descriptions that are compatible with the Synopsys Design Compiler and Module Compiler tools, suitable for mapping to standard cell libraries. The Module Compiler Language (MCL) allows a function to be described with a single parameterized file, with conditional I/O ports, a capability not supported by VHDL or Verilog, but necessary for compatibility with XSG. Most basic blocks, including adders, multipliers, registers, and multiplexers are supported with this approach, with the goal of mapping the blocks to standard cells provided by the ASIC foundry. If the foundry provides smaller and more efficient IP blocks such as memory, the designer can include them by writing a VHDL wrapper that translates the timing and signaling expected by the original libraries into the timing expected by the ASIC vendor IP.

Logic Synthesis

Once the MCL and VHDL have been generated, INSECTA builds scripts to run Module Complier and Design Compiler, and executes these tools. Module Compiler is first run to generate VHDL for library primitive blocks. INSECTA allows constraints to be passed to Module Compiler to improve compilation results.

Design Compiler is then executed to map the complete design to the gate level, producing both structural VHDL and Verilog output files. In addition, an SDF file is produced, containing library parameters that are passed to Modelsim to improve the accuracy of the functional simulation. The functional simulation is performed using testbench created in Simulink.

Upon completion, area, speed, and power estimates are obtained which provide early architectural feedback for design comparison and analysis.

HDL Simulation

Once the design is synthesized, Modelsim can be run to verify functional equivalence and to extract statistics to improve performance estimates. The same test-bench and data recorded by Simulink can be applied to the synthesized design to confirm that the results are still cycle accurate and bit accurate. If custom VHDL code is written for IP such as memories, the functional simulation also allows validation of these wrappers. INSECTA creates command scripts for Modelsim that allow user-defined libraries to be included during compilation and simulation.

To produce more accurate power estimates later in the flow, the simulation uses the SDF file generated by the first synthesis to give Modelsim more accurate timing information about the standard cell libraries. Modelsim can then record signal activity factors for each internal node. By leveraging the SDF file for more accurate timing simulation, glitch activity will affect the activity factor and lead to more realistic power estimates during the second synthesis.

7.3.2 Chip Synthesis

With activity factors in hand, Design Compiler can then re-synthesize the design to produce a more efficient result, as well as more accurate power estimates. As before, the estimates might suggest areas for improving the architecture to reduce power consumption. The designer can iterate this step until desired results are obtained.

Register Retiming

The netlist is then optimized using a set of custom scripts for register retiming and logic optimization, before going to the final stage of physical layout synthesis. The "optimize_registers" command in Design Compiler is used to perform register retiming on a mapped gate-level netlist. It determines the placement of registers in a design to achieve a target clock period and minimizes the number of registers while maintaining that clock period. It improves designs that already contain registers. Retiming is done on simple feed-forward blocks followed by top-level incremental compile. This procedure can be easily expanded to loop retiming as described in Chapter 5.

Physical Synthesis

After the logic synthesis has completed, the results are exported to a vendor-specific *place*-and-*route* design flow. Different ASIC vendors generally provide recommended design flows from HDL or RTL descriptions through synthesis to place-and-route tools. Our back-end flow is based on a standard Synopsys suite of tools for *floorplanning* and *place*-and-*route*. An Astro flow was used for the back-end with occasional use of Physical Compiler for placement optimization. All of the steps are highly automated by our ASIC vendor.

The resulting netlist from physical synthesis is simulated for functionality using test vectors from Simulink, in addition to formal logic verification. Detailed checks for timing, cross-talk, DRC, and LVS are performed on the extracted final layout.

7.4 Design Example: Iterative Square Rooting

The proposed design methodology is applied to the iterative square root algorithm from Chapter 5. The intent is to illustrate a procedure that is generally applicable.

Step 1: Design Entry in Simulink

The first step in the design procedure involves modeling in Simulink. The goal is to create a cycle-accurate model of the design using the XSG blocks. This step may also involve functional verification against an ideal floating-point model described either in block form or in M-code.

Figure 7.7 illustrates a typical top-level view of an algorithm model. The algorithm functionality is realized with the XSG blocks ("sqrt (fix.pt)") and also with ideal floating point block-set ("sqrt (float.pt)"), which allows comparison of the results and their graphical display ("zout"). The System Generator block provides control of system and simulation parameters and is used to invoke the code generator. Every Simulink model containing any element from the XSG library must contain at least one System Generator block. Once a System Generator block is added to a model, it is possible to specify how code generation and simulation should be handled. The inputs are stored in ROM memories for an FPGA emulation purposes.

The Xilinx Resource Estimator block provides fast estimates of FPGA resources required to implement a System Generator subsystem or model. These estimates are based on block-specific estimators for Xilinx blocks, which obtain estimates of lookup

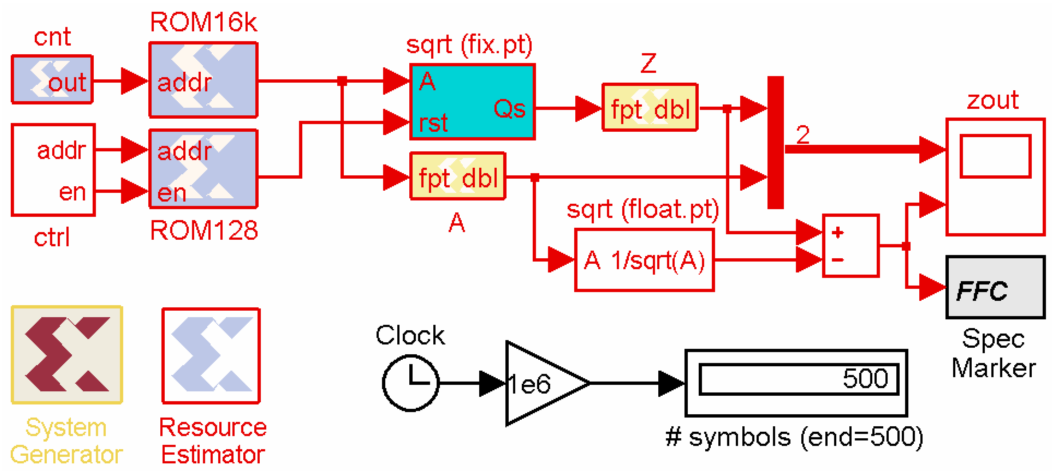
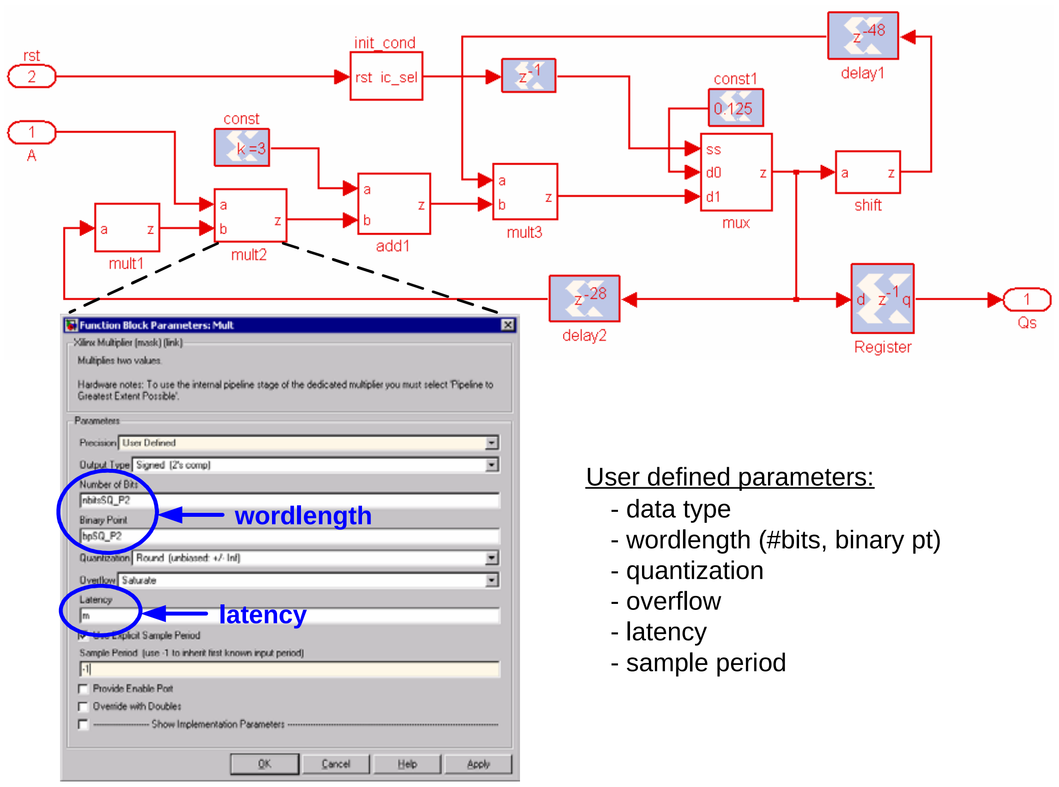


Figure 7.7: Simulink model of inverse square rooting algorithm from Chapter 5.

tables (LUTs), flip-flops (FFs), block memories (BRAM), 18×18 multipliers, tristate buffers, and I/Os. An estimator block can be placed in any subsystem of a model to provide estimates for individual subsystems.

Each functional block from the XSG library can be configured with user-defined parameters. Figure 7.8 shows macro-architecture of the implemented algorithm. Blocks that implement arithmetic operations such as *add* and *multiply* are grouped in subsystems for retiming purposes later. This grouping is done to keep track of naming hierarchy in Design Compiler. The insert in Fig. 7.8 is parameter dialog box for the "mult2" subsystem. User can specify a range of parameters including data type, wordlengths, quantization, latency, cycle time, etc.

The *multiplier* block also supports a size-performance trade-off in its implementation. It can be implemented either as a parallel multiplier that operates on the full width data (faster and larger) or as a sequential multiplier that computes the result



Function Block Parameters: Mult

Xilinx Multiplier (mask) (link)

Multiplies two values.

Hardware notes: To use the internal pipeline stage of the dedicated multiplier you must select 'Pipeline to Greatest Extent Possible'.

Parameters

Precision: User Defined

Output Type: Signed (2's comp)

Number of Bits: nbitsSQ_P2

Binary Point: bpSQ_P2

Quantization: Round (unbiased: +/- Inf)

Overflow: Saturate

Latency: m

[✓] Use Explicit Sample Period

Sample Period (use -1 to inherit first known input period): -1

[ ] Provide Enable Port

[ ] Override with Doubles

[ ] Show Implementation Parameters

OK Cancel Help Apply

User defined parameters:
- data type
- wordlength (#bits, binary pt)
- quantization
- overflow
- latency
- sample period

Figure 7.8: Simulink/XSG model of "sqrt (fix.pt)" block from Fig. 7.7. Each module has a mask with user-defined parameters (word-length and latency are emphasized).

from smaller partial products (slower and smaller). This choice affects the hardware implementation only. The simulation behavior of the block is not affected. For the ASIC flow, this does not matter since each module is later synthesized by INSECTA. Mandatory parameters for the ASIC flow are latency and word-length, Fig. 7.8.

Step 2: Word-length Optimization

Upon establishing functionally correct behavior, the next step is word-length optimization to reduce the area/power of the implementation. The goal is to obtain the word size at all nodes. For this purpose, the FFC spec marker block is placed in the

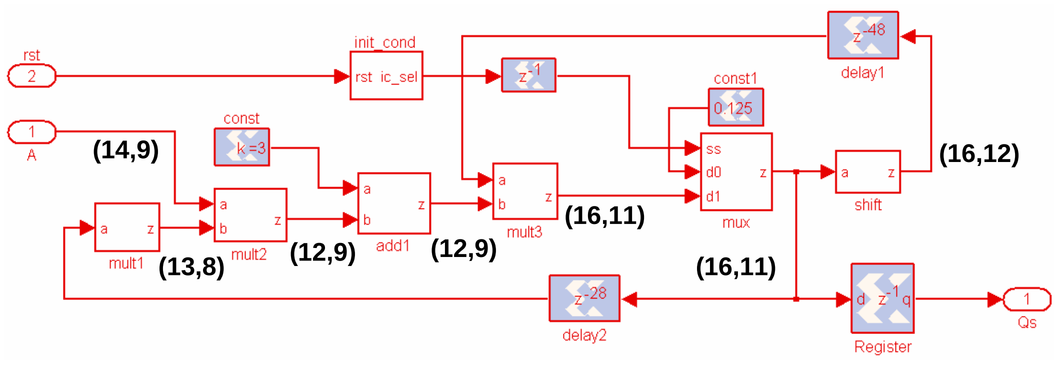


Figure 7.9: Word-length optimized design. The numbers indicate (total, fractional) bits, respectively. Area before opt: 743 slices, area after opt: 668 slices.

model shown in Fig. 7.7, so a user can specify the error tolerance due to quantization.

The result of word-length optimization is shown in Fig. 7.9. The numbers in brackets indicate total and fractional bits, respectively. This model is simulated to verify the algorithm behavior under finite word size. A sinusoidal input spanning the desired input range is used. Figure 7.10 is the result of functional check. The plot on the left shows the input waveform and output from floating and fixed point simulations, and the difference is displayed in plot on the right. The algorithm is initialized with 0.125, which results in discrepancy for few iterations until the algorithm converges. Each iteration is then initialized with solution of previous step to achieve single-cycle convergence for continuously varying input.

Step 3: Technology Characterization

The architecture has to be selected before technology mapping in order to avoid costly design iterations. The choice of architecture is highly dependent on characteristics of

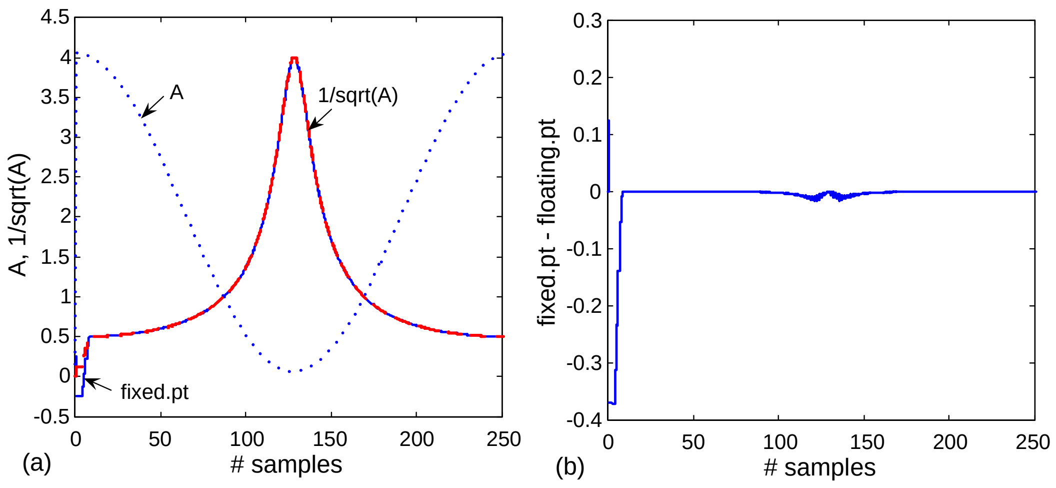


Figure 7.10: Functional verification of iterative square rooting: a) fixed and floating point outputs for sinusoidal input A, b) error due to finite wordlength effects.

the underlying technology. The objective of this step is to determine relative speed of the technology compared to the required cycle time for the application, to be able to optimally combine circuit-level and architecture-level parameters.

At the circuit level, standard calibration test is to determine the energy-delay trade-off for a fanout-of-four (FO4) inverter, which is a metric commonly used for delay scaling in digital circuits. The shape of the E-D curve for an inverter as shown in Fig. 7.11(a) can be used to predict the E-D scaling for CMOS logic in general, to a first approximation. Given a clock period specification, the designer can determine the FO4 complexity of pipelined logic that maintains a good E-D trade-off point as indicated on the plot. Related to the prior discussion, this point corresponds to the general trade-off illustrated in Fig. 5.1 in Chapter 5. The optimal point is reached through supply voltage reduction, which is the most efficient way to reduce power, and sizing optimization as discussed in following steps.

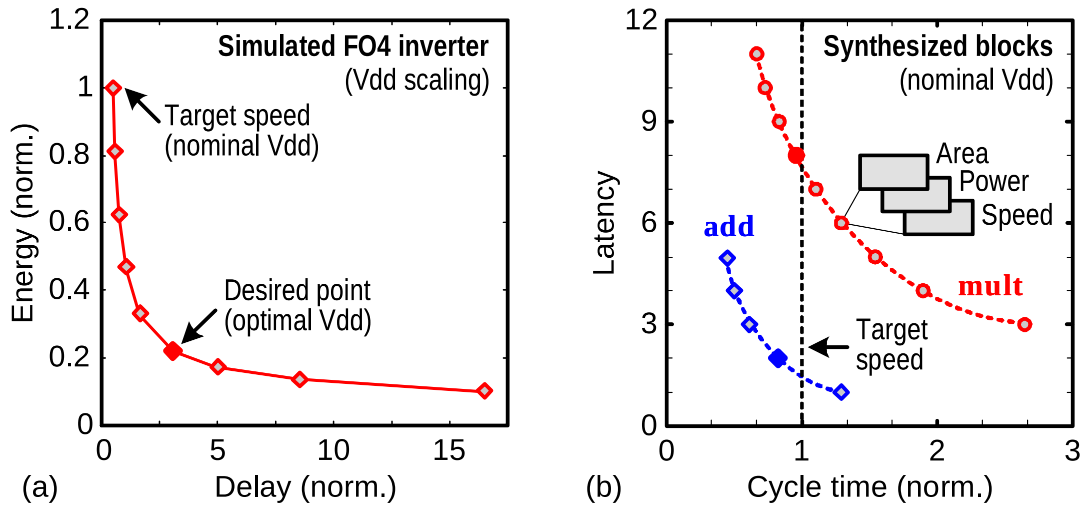


Figure 7.11: Technology characterization: a) Simulated Energy vs. Delay of a FO4 inverter (normalized to nominal Vdd), b) Latency vs. Cycle time of building blocks.

Going up toward the architectural level, it is most important to understand right amount of cycle latency associated with each arithmetic block. The goal is to balance the complexity in pipelined logic at the top level. The block characterization, as explained earlier in this chapter, is the key to obtaining this information. The desired operating point from Fig. 7.11(a) translates into some equivalent target speed for logic synthesis at the nominal supply as shown in Fig. 7.11(b), which is used to determine the cycle latency for the required speed.

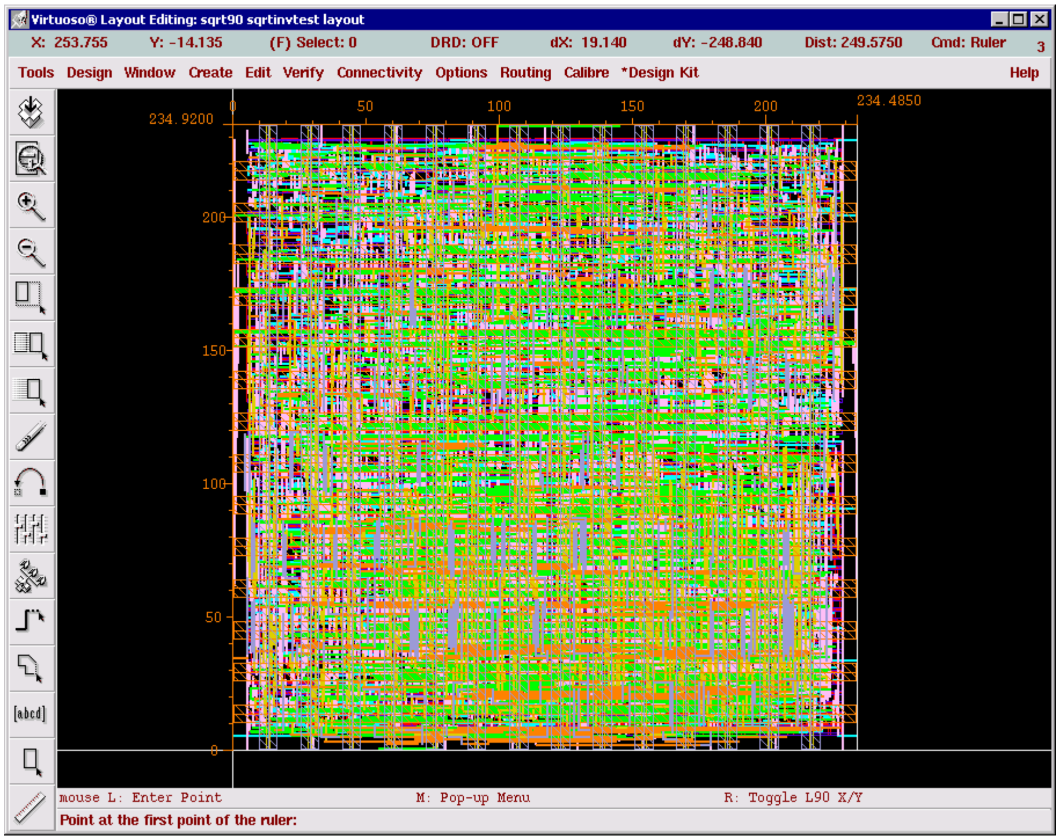
Step 4: Architectural Optimization

Architectural optimization starts with a direct mapped parallel implementation. The objective is to minimize the power and area of the design. The outcome of this step highly depends on the area and energy requirements for a specific design. A number of architectural choices exist in energy-area space that satisfy the same throughput as shown in previous chapters. Still, some general guidelines exist.

A good heuristic for simultaneous power and area minimization is outlined here. In overall design optimization for power and area, the variables affecting only the power or only the area should be exploited first, followed by others that affect both power and area. For example, supply voltage is a major parameter for power reduction, but it does not affect the area of the design. Similarly, interleaving and folding reduce area without affecting power, to a first order, as discussed in Chapter 5. Generally, power is more constraining than area, so the first goal in the optimization is to position the operating point at the trade-off point illustrated in Fig. 7.11(a), compare the achieved circuit speed with the required speed, and finally choose a suitable micro-architecture. Time-multiplexing should be used if the achieved speed is too high, otherwise pipelining or parallelism should be used. In most wireless applications, the required speed is very low so time-multiplexed architectures should be used, which is against the popular belief that parallelism is the best choice due to its power efficiency (this is only true if the required throughput is very high).

Step 5: Chip Synthesis

The final step for a fixed architecture design is chip synthesis. This step is fairly straightforward with the support of highly automated, commercial back-end tools. Due to a limitation of synthesis tools, joint optimization of gate sizing and supply


Tools Design Window Create Edit Verify Connectivity Options Routing Calibre *Design Kit Help
234.9200
234.4850
0
50
100
150
200
[abcd]

Virtuoso® Layout Editing: sqrt90 sqrtinvtest layout

X: 253.755 Y: -14.135 (F) Select: 0 DRD: OFF dX: 19.140 dY: -248.840 Dist: 249.5750 Cmd: Ruler 3

mouse L: Enter Point M: Pop-up Menu R: Toggle L90 X/Y

Point at the first point of the ruler:

Figure 7.12: Layout view of the inverse sqrt design. ($235\mu m \times 235\mu m, 90nm$ CMOS)

voltage cannot be performed, because the standard cell libraries are typically characterized for a single value of supply voltage.

Following the lessons of continuous circuit-level optimization, gate sizing is most effective for small incremental delays, so the clock cycle constraint for initial synthesis should be set to be about $20\% - 30\%$ more aggressive than the required speed. The result of initial synthesis is basically a fixed architecture of the design (e.g. type of adder and multiplier). The next step is to perform incremental compilation on this design to utilize gate sizing for power reduction.

Register Retiming (Optional)

Optionally, a design can be retimed to optimally (re)-distribute registers in the design. Hierarchical retiming is performed using the simple procedure outlined in Chapter 5. The key to highly efficient retiming in terms of CPU runtime is in correctly specifying block latencies, using block characterization results like those in Fig. 7.5. Block-based retiming is performed followed by incremental compilation at the top-level.

The final phase in producing chip layout is physical synthesis through commercial place and route tools. A good practice in avoiding synthesis iterations is to over-constrain the design with some extra budget for clock tree synthesis and routing steps. Typically, about 10% excess timing is sufficient. The final result of physical synthesis is a design layout, such as that illustrated in Fig. 7.12 for the inverse square root block.

Step 7: Power/Area Estimation

Finally, final estimates of power, speed, and area are obtained based on the netlist with complete parasitic information extracted from layout. In this particular example, a 16-bit multiplier consumes about 6$\times$ the area of a 16-bit adder. The overall complexity of the design in Fig. 7.9 is estimated to be about 20 equivalent adders, including latency balancing registers and control logic. Technology specific, absolute numbers are omitted here for the sake of generality.

The inverse square root example presented in this chapter is used as a building block to implement a much more complex algorithm: an adaptive 4×4 singular value decomposition (SVD).

Chapter 8

Design Example: 4×4 SVD

This chapter presents decoupling of a MIMO wireless channel through singular value decomposition, as an example of joint power and area minimization. The SVD algorithm performs adaptive real-time decoupling of a 4×4 complex matrix. Signal processing, architecture and circuit level techniques are simultaneously considered during design optimization. Adaptive signal processing for MIMO channel decoupling is analyzed with focus on the VLSI realization of main building blocks: eigen-mode decomposition and Grahm-Schmidt orthogonalization. The optimization procedure and preliminary estimates of power and area are discussed.

8.1 MIMO System Model

Performance of the SVD algorithm can be best evaluated in a realistic closed loop environment as shown in Fig. 8.1. The channel matrix is estimated at the receiver,

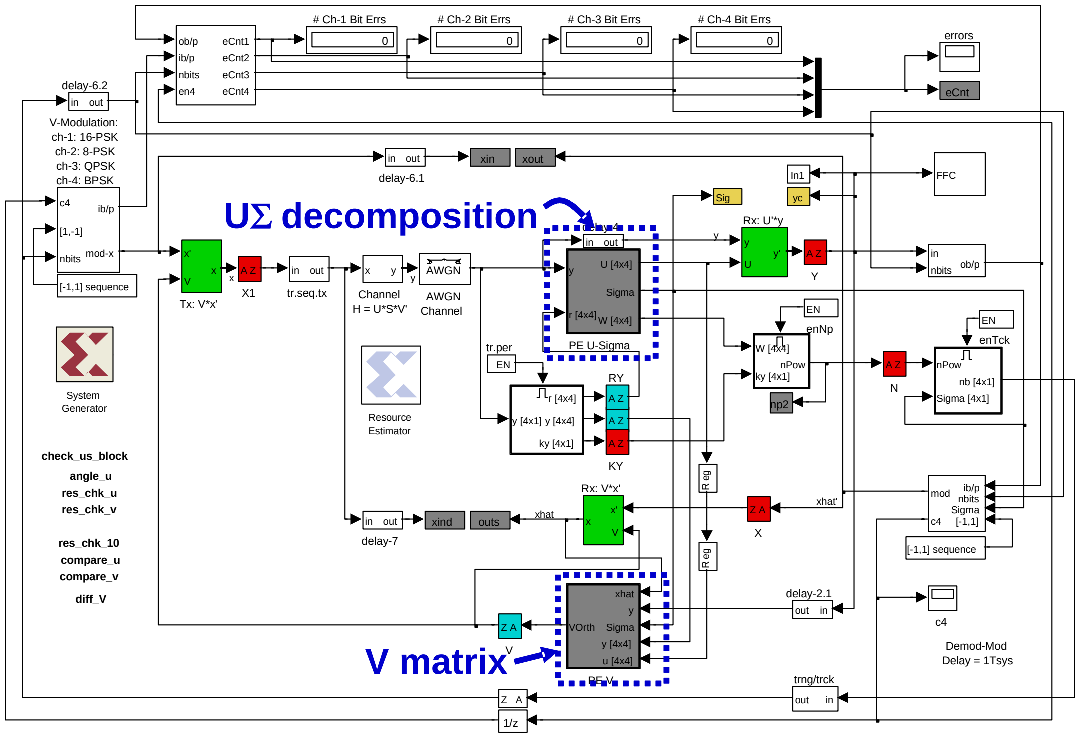


Figure 8.1: Simulink model of adaptive SVD.

which periodically sends partial channel information $\boldsymbol{V}$ back to the transmitter to assist in channel decoupling. The throughput is maximized using adaptive modulation at the transmitter, based on the estimated SNR in spatial sub-channels. The model also monitors bit errors on a per-channel basis to verify bit-accurate behavior under ideal channel conditions.

The main building blocks of the SVD algorithm [56] are highlighted in Fig. 8.1. Blind tracking of the channel is realized in $\boldsymbol{U\Sigma}$ decomposition and $\boldsymbol{V}$ matrix blocks. System is modeled using a mix of M-code (S-functions) and fixed-point blocks to speed-up the simulations. A “fixed-point” behavior is modeled with a floating-point block followed by register from the XSG library.

The narrow-band algorithm from [56] is extended to wide bandwidth by simply interleaving multi-carriers. Ideally, decoupling of the channel has to be performed on all sub-carriers simultaneously. This can be properly modeled in Simulink with the choice of appropriate sampling period and time re-ordering of samples to represent data-stream interleaving. The Simulink model is finally refined with word-size optimization that minimizes hardware area, subject to user defined MSE error due to quantization. After determination of the optimal word-lengths and functional verification of the algorithm, the next step is to analyze core building blocks.

8.2 SVD: Signal Processing View

The key task in algorithm description using hardware blocks is to create a partition of functional blocks, which are described in equation format, into processing elements that can be mapped to hardware. The first order grouping for simulation purposes primarily has to satisfy block-level latency requirements to ensure functionally correct behavior. Block latencies are parametrized to allow hierarchical design specification as explained in Chapter 7.

8.2.1 Eigen-mode Decomposition

This section studies signal processing operations required for adaptive eigen-mode decoupling of indoor wireless channels, as part of the SVD algorithm. The SVD in

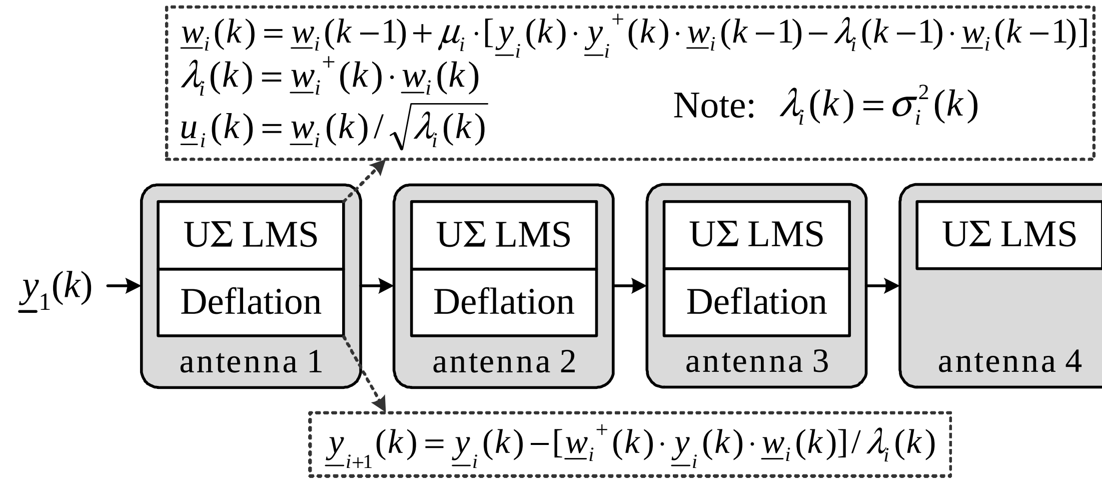

$\underline{w}_i(k) = \underline{w}_i(k-1) + \mu_i \cdot [\underline{y}_i(k) \cdot \underline{y}_i^+(k) \cdot \underline{w}_i(k-1) - \lambda_i(k-1) \cdot \underline{w}_i(k-1)]$
$\lambda_i(k) = \underline{w}_i^+(k) \cdot \underline{w}_i(k)$
$\underline{u}_i(k) = \underline{w}_i(k) / \sqrt{\lambda_i(k)}$
Note: $\lambda_i(k) = \sigma_i^2(k)$
$\underline{y}_1(k)$
UΣ LMS
Deflation
antenna 1
UΣ LMS
Deflation
antenna 2
UΣ LMS
Deflation
antenna 3
UΣ LMS
antenna 4
$\underline{y}_{i+1}(k) = \underline{y}_i(k) - [\underline{w}_i^+(k) \cdot \underline{y}_i(k) \cdot \underline{w}_i(k)] / \lambda_i(k)$

Figure 8.2: Adaptive 4-by-4 eigen-mode decomposition algorithm.

[56] reduces matrix operations to vector operations at the expense of extra square root and division operations. The core of the algorithm is adaptive least mean squares (LMS) and deflation associated with each antenna.

The algorithm performs LMS-based estimation of eigen-pairs ($\boldsymbol{u}_i$, σ_i), and deflation for successive rank reduction as shown in Fig. 8.2. The eigen-modes are updated every symbol period. Square root and division are implemented using an iterative Newton-Rhapson method achieving single-iteration convergence under slow channel variations. The algorithm is highly adaptive; the step-size of the UΣ LMS is also adaptively adjusted as $0.05/\lambda_i$.

The narrow-band algorithm described in Fig. 8.2 is extended over 16 sub-carriers. Each sub-carrier performs the same operation, which is convenient for the data-stream interleaving presented in Chapter 5.

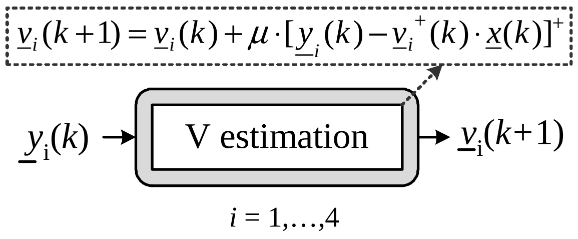

$\underline{v}_i(k+1) = \underline{v}_i(k) + \mu \cdot [\underline{y}_i(k) - \underline{v}_i^+(k) \cdot \underline{x}(k)]^+$
$\underline{y}_i(k)$
V estimation
$\underline{v}_i(k+1)$
$i = 1,\ldots,4$

Figure 8.3: Adaptive algorithm for tracking of $\boldsymbol{V}$ matrix.

8.2.2 Estimation of V matrix

The estimation of $\boldsymbol{V}$ matrix is quite simple in comparison to the eigen-mode decomposition. Tracking of the $\boldsymbol{V}$ matrix is done concurrently on a per-vector basis as shown in Fig. 8.3. This information is sent back to the transmitter with periodicity of a couple hundred symbols. For reduced frequency of feedback information, estimation of the $\boldsymbol{V}$ matrix is done using the detected symbols at the receiver instead of transmitter-based estimation that relies on previously transmitted symbols [56].

One caveat for the SVD algorithm is that one of the matrices $\boldsymbol{U}$ or $\boldsymbol{V}$ has to be periodically orthogonalized to guarantee closed loop convergence. Since the estimation of $\boldsymbol{V}$ is done based on detected symbols, which have prior assumption of $\boldsymbol{U}$, small phase of amplitude variations can drive the algorithm out of stability if the vectors are not orthogonal. Orthogonalization of the $\boldsymbol{V}$ eigen-vectors is forced and the orthogonalized set of vectors is sent back to the receiver.

The Grahm-Schmidt orthogonalization procedure [88] is used for vector orthogonalization. It is an interesting signal processing operation by itself, since vector projection onto an orthogonal base is commonly found in communication algorithms.

Grahm-Schmidt Orthogonalization

The Grahm-Schmidt procedure constructs a set of orthonormal waveforms out of a set of finite energy signal waveforms, [88]. The first waveform is simply constructed by normalization to unit energy. Each successive waveform is constructed by first computing the projections of prior waveforms from the new base onto the current waveform, then normalizing the result to unit energy. This procedure is detailed on the example using a set of four vectors $\{\underline{v}_1, \underline{v}_2, \underline{v}_3, \underline{v}_4\}$, Eq. (8.1).

$$\begin{aligned}
\underline{v}_{1n} &= \frac{\underline{v}_1}{\sqrt{\underline{v}_1^{\dagger} \cdot \underline{v}_1}} \\
\underline{v}_{2p} = \underline{v}_2 - \underline{v}_{1n} \cdot \underline{v}_2, \quad \underline{v}_{2n} &= \frac{\underline{v}_{2p}}{\sqrt{\underline{v}_{2p}^{\dagger} \cdot \underline{v}_{2p}}} \\
\underline{v}_{3p} = \underline{v}_3 - \underline{v}_{1n} \cdot \underline{v}_3 - \underline{v}_{2n} \cdot \underline{v}_3, \quad \underline{v}_{3n} &= \frac{\underline{v}_{3p}}{\sqrt{\underline{v}_{3p}^{\dagger} \cdot \underline{v}_{3p}}} \\
\underline{v}_{4p} = \underline{v}_4 - \underline{v}_{1n} \cdot \underline{v}_4 - \underline{v}_{2n} \cdot \underline{v}_4 - \underline{v}_{3n} \cdot \underline{v}_4, \quad \underline{v}_{4n} &= \frac{\underline{v}_{4p}}{\sqrt{\underline{v}_{4p}^{\dagger} \cdot \underline{v}_{4p}}}
\end{aligned} \tag{8.1}$$

8.3 SVD: Circuit Perspective

The main idea to explore at the circuit level is how to make use of the excess speed available in today's technology, compared to the clock rate required from a direct-mapped parallel realization of a digital baseband. This allows for aggressive voltage scaling and gate sizing to minimize power consumption. The goal is to keep the energy-delay sensitivity balanced in across pipelined logic.

Circuit-level results are essential in choosing the optimal micro-architecture, which is tightly coupled with the required supply voltage. An important trade-off to consider at very low V_{dd} is whether to store data in memory or flip-flops, since these structures are first to fail at low voltage, [31].

Supply voltage scaling is beneficial for the minimization of active power in logic blocks and leakage power in memories. The lower limit to V_{dd} scaling is commonly found in SRAM memories.

8.3.1 Voltage Scaling Limit: SRAM Memory

This section examines the lower limit on supply voltage scaling in a standard SRAM module optimized for nominal supply voltage. By reducing the standby supply voltage to its limit, that is the Data Retention Voltage (DRV), leakage power can be substantially reduced. An analytical model for DRV as a function of process and design parameters is found in [89], [90]. The derivation presented is based on sub-threshold current model, applicable to very low V_{dd}. The model is verified using simulations and also measurements from 32Kb SRAM chip in a $0.13\mu m$ technology.

A commercial SRAM module with a high-V_t process is shown to be capable of sub-300mV standby data preservation, Fig. 8.4. Under this low standby V_{dd}, leakage power saving of more than 90% can be achieved with a dual-rail standby scheme in [90]. The DRV is observed to be a strong function of process variation and SRAM cell sizing. Due to large on-chip variation, the DRV of the 32Kb SRAM module ranges

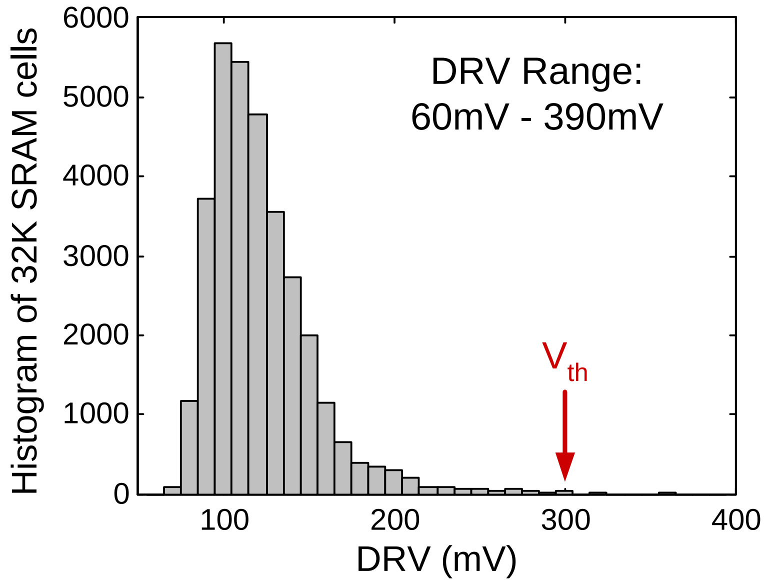


Figure 8.4: Histogram of data retention voltage in SRAM cells (130nm process).

between 60mV and 390mV, with the mean value around 120mV.

While the memory states can be preserved at sub-300mV V_{dd}, adding an extra guard band of 100mV to the standby V_{dd} enhances data stability. With the resulting 490mV standby V_{dd}, SRAM leakage current can still be reduced by 85%. The long tail of the histogram increases the minimal V_{dd} quite substantially. An important design decision is to store data directly in pipeline registers to be able to aggressively reduce the supply voltage below DRV.

8.3.2 Optimal Design at Scaled Vdd

This section presents a procedure for design optimization at a reduced V_{dd} withing synthesis environment that works with nominal supply voltage. Transistor level simulations of a 90nm standard-cell inverter optimized for 1V shows that V_{dd} can be

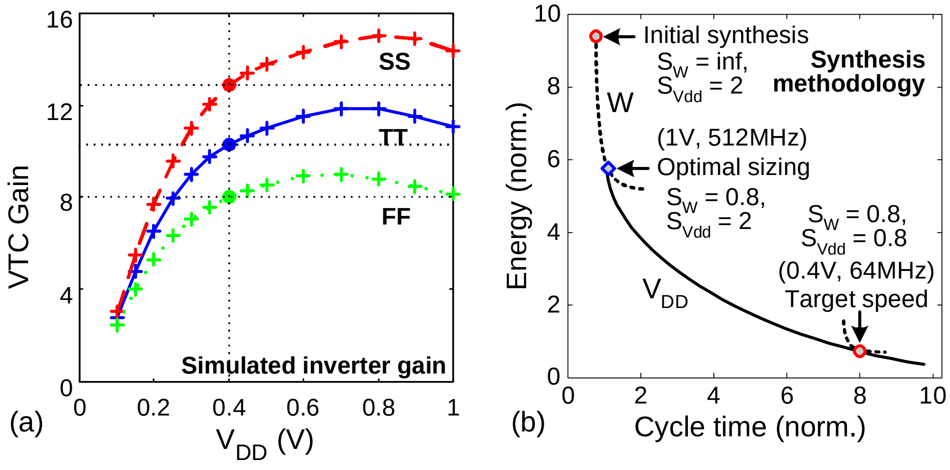


Figure 8.5: (a) Inverter VTC does not degrade at V_{DD}=0.4V. (b) Energy minimization via gate sizing and V_{DD} reduction.

scaled down to 0.4V, without compromising static VTC characteristic of logic gates, Fig. 8.5(a). The target clock speed for a chosen architecture is 64MHz at 0.4V in the SS corner which translates into 512MHz timing constraint for logic synthesis under the worst case model (0.9V, 125C).

Due to limitations of the synthesis tool, a sequential procedure is applied to balance the trade-offs with respect to logic sizing (W) and supply voltage (V_{dd}) [91]. First, logic synthesis is constrained with a 20% slack at the nominal supply, followed by sizing optimization, as illustrated in Fig. 8.5(b). The supply voltage is then scaled down to 0.4V to balance the sensitivities to 0.8, resulting in a design optimized for the target speed of 64MHz. The sensitivity information with respect to gate sizing can be simply obtained from the power and speed estimates of incremental compilation around the design point, while V_{dd} sensitivity is calculated using Eqs. (3.9) and (3.10) from Chapter 3.

This procedure can be applied to any fixed architecture design. Architectural optimization is done in Simulink, based on the energy-area-delay estimates of building blocks such as *add* or *multiply*. Word-length reduction is also performed using an automated Simulink-based FFC optimizer [86] that minimizes hardware area subject to quantization error at the output. The word-length optimization results in a 30% reduction in area and energy compared to a 16-bit design for the case of UΣ decomposition block.

8.4 Optimizing VLSI Architecture

Architectural optimization minimizes area and energy/power for a fixed throughput. The minimum in area and energy is achieved using the architectural/circuit techniques that yield largest area or energy savings for a given decrease of throughput, starting from a direct-mapped architecture at a nominal supply. This process is repeated until all the techniques are balanced, [91]. The clock speed required to implement the algorithm in a direct-mapped parallel architecture is significantly below the capability of current technology. This is used to reduce power and area through concurrent architecture and circuit-level optimizations.

Techniques for energy and area minimization are illustrated in Fig. 8.6. Starting from a reference design with optimal V_{dd} and W, interleaving and folding reduce the area without an energy increase, as shown in Fig. 5.3 in Chapter 5. Both techniques introduce pipeline registers around feedback loops, but also speed-up the clock to

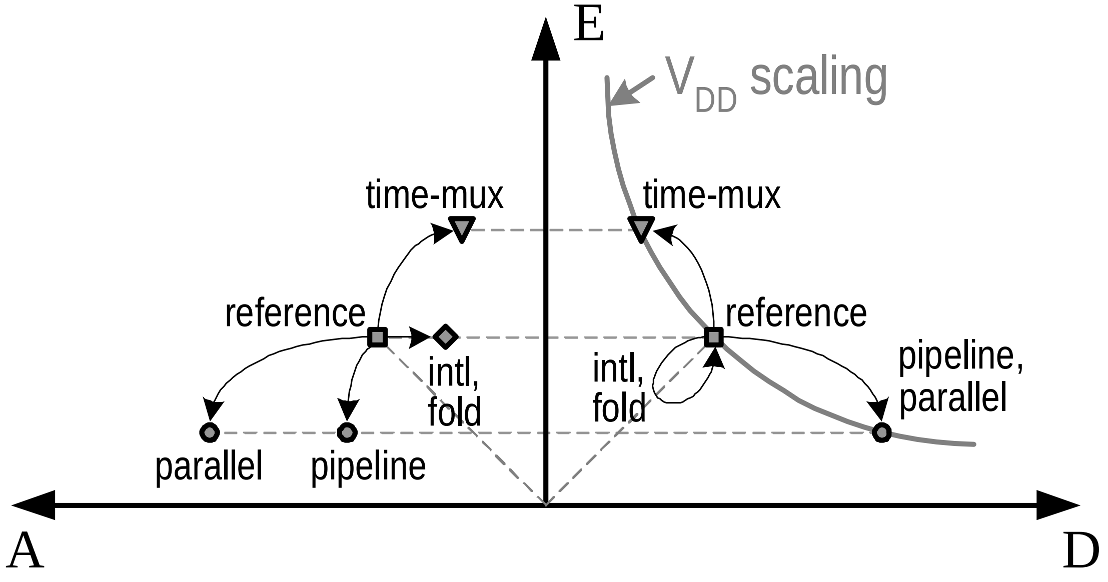


Figure 8.6: Basic Energy-Area-Delay trade-offs for constant throughput.

maintain throughput, thus coming back to the original point on the energy-delay line of pipelined logic blocks (assuming negligible overhead in registers). Parallelism and pipelining map out to approximately the same energy point, with parallelism requiring more implementation area. This favors the pipelining approach in technologies with high leakage power. Time-multiplexing decreases the area, but may increase energy substantially, unless the delay of the pipelined logic blocks in the reference architecture is very long.

Architectural transformations illustrated in Fig. 8.6 are the core of the design methodology. The architectural techniques outlined here are applied next to the eigen-mode (UΣ) decomposition algorithm and the Grahm-Schmidt orthogonalization procedure.

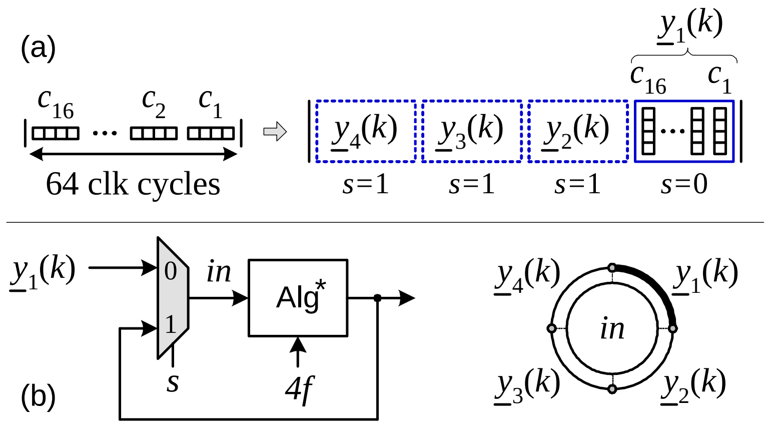

(a)
c_{16} c_2 c_1
64 clk cycles
$\underline{y}_4(k)$ $\underline{y}_3(k)$ $\underline{y}_2(k)$
$\underline{y}_1(k)$
c_{16} c_1
s=1 s=1 s=1 s=0
(b)
$\underline{y}_1(k)$
0
1
s
in
Alg*
4f
$\underline{y}_4(k)$ $\underline{y}_1(k)$
in
$\underline{y}_3(k)$ $\underline{y}_2(k)$

Figure 8.7: (a) Vectoring and time-serial ordering of interleaved data. (b) Folding of *Alg* operation (UΣ LMS and deflation).

8.4.1 Architecture for Eigen-mode Decomposition

The UΣ block is optimized for 16 simultaneous 4×4 matrix decompositions on 1MHz-wide sub-carriers. Since 1MHz-wide sub-channels require 1MS/s to process the data, a direct-mapped architecture needs a 1MHz baseline clock to realize the Alg^* operation with one cycle of latency. The critical loop of the algorithm has 6 real *multiply*, 11 *add*, and 2 *mux* operations. While this is feasible within 1μs, even at a reduced V_{dd}, area minimization dictates a streamed architecture with folding, resulting in a 64MHz clock rate (1MHz × 16 sub-carriers × 4 antennas).

Using data-stream interleaving over antennas/vectors and sub-carriers, Fig. 8.7(a), the area and routing complexity are reduced. Vectoring and re-organization in time domain allows for folding over the antennas, Fig. 8.7(b), for further area reduction. This is the final architecture used in the ASIC realization. Memory inside the Alg^* block (∼64Kb) is directly realized in pipeline registers to allow aggressive V_{dd} scaling.

Table 8.1: Area of UΣ block architectures. ($A^{SEQ}/A^{LOG} = 1\%$, $A_{Def}/A_{U\Sigma} = 0.5$)

Direct mapped	$16 \cdot \left(4A_{U\Sigma}^{SEQ,LOG} + 3A_{Def}^{SEQ,LOG}\right)$
Interleaved	$16 \cdot \left(4A_{U\Sigma}^{SEQ} + 3A_{Def}^{SEQ}\right) + 4A_{U\Sigma}^{LOG} + 3A_{Def}^{LOG}$
Intl/Fold	$64 \cdot \left(A_{U\Sigma}^{SEQ} + A_{Def}^{SEQ}\right) + A_{U\Sigma}^{LOG} + A_{Def}^{LOG}$

In the direct-mapped parallel implementation (reference design), the area of the sequential elements is about 1% of the logic area, with the deflation block being about half the size of the UΣ block. Starting from the reference design, the area of architectures with interleaving and combined interleaving and folding is obtained using the equations in Table 8.1. The area of data-path logic is shared by sub-carriers and also over antennas, leading to a 36× reduction in total area due to interleaving and folding combined.

Techniques for energy and area minimization are illustrated in Figs. 8.8, 8.9. Upon word-size optimization, architectural optimization is performed based on the circuit-level energy-delay characteristics and the top-level throughput specification. The final architecture is taken through logic synthesis that also optimizes the gate size. All techniques combined result in 64× area and 16× power reduction.

The example presented in this section is quite complex, but highly regular; the same signal processing operation is performed on multiple independent data-streams. Operations are partitioned on a per vector basis, interleaving and folding the data for substantial area reduction. The next example examines the Grahm-Schmidt orthogonalization procedure, where such a convenient partitioning is not possible.

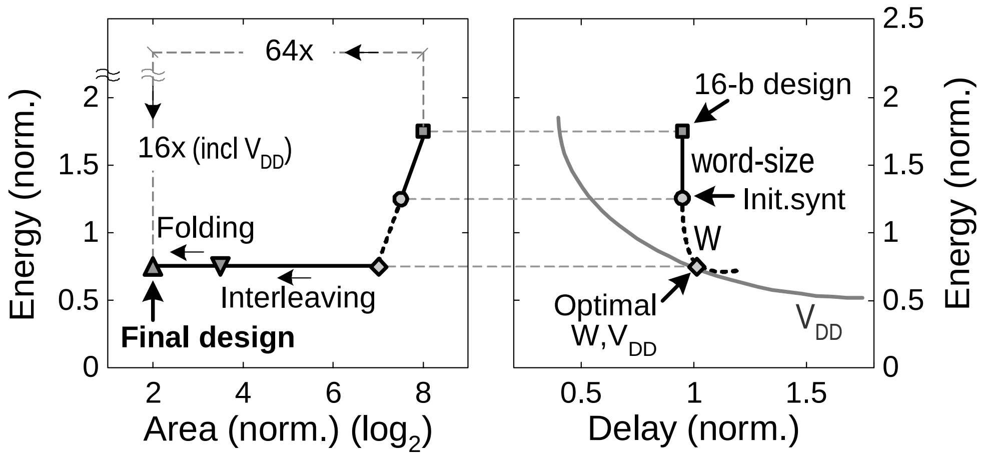


Figure 8.8: Energy and Area minimization in $\boldsymbol{U\Sigma}$ decomposition example.

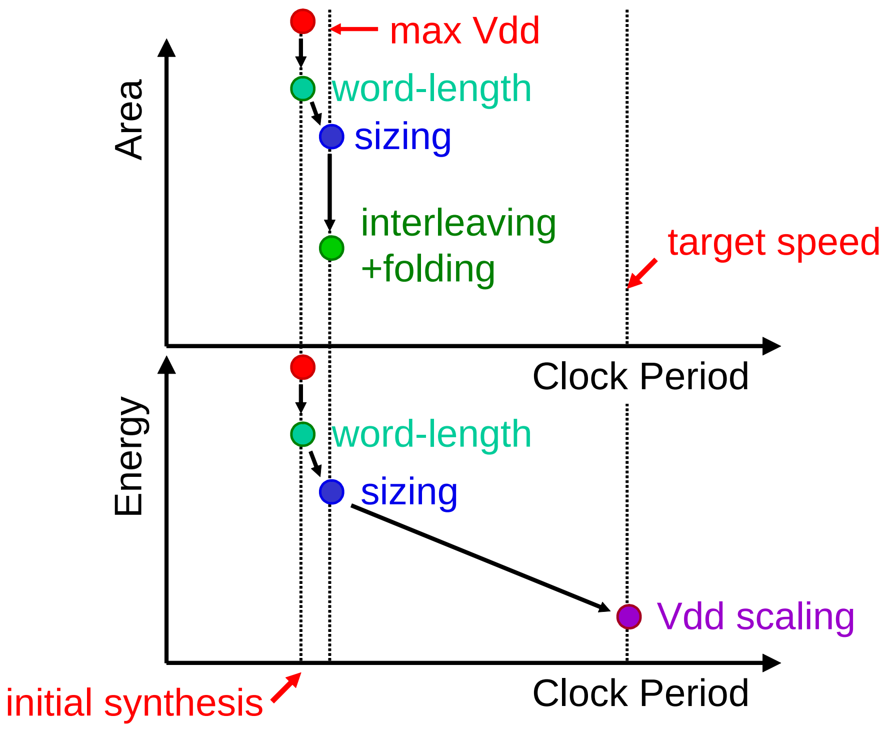


Figure 8.9: Conceptual view of optimization techniques used in $\boldsymbol{U\Sigma}$ block.

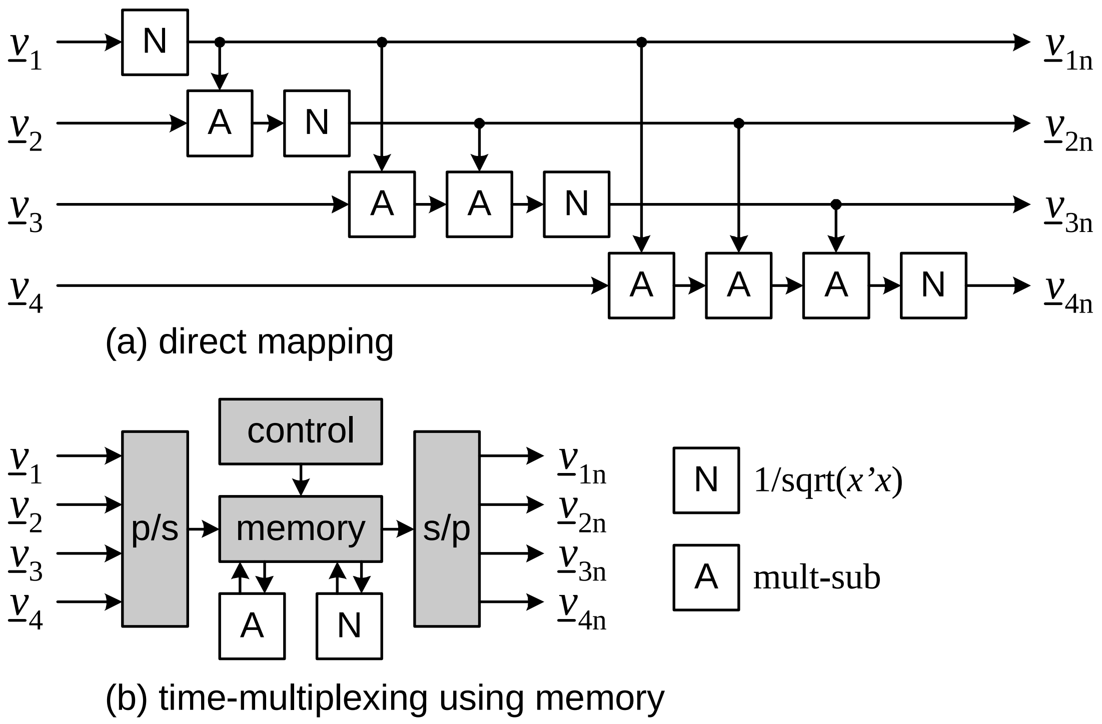


Figure 8.10: Grahm Schmidt Orthogonalization: (a) direct mapped architecture, (b) memory based time-multiplexing approach.

8.4.2 Architecture for Grahm-Schmidt Orthogonalization

The concept of Grahm-Schmidt orthogonalization (GSO) is repeated in Fig. 8.10(a). Two basic operations are indicated in boxes A and N. In the figure, $\underline{v}_1 - \underline{v}_4$ are complex vectors of dimension 4, and $\underline{v}_{1n} - \underline{v}_{4n}$ is their orthogonal form. Since input vectors $\underline{v}_1 - \underline{v}_4$ experience different signal processing operations, this algorithm is not suitable for straightforward interleaving. Instead, time-multiplexing is applied to the repetitive operations A and N.

The regular time-multiplexing (TM) introduced in Chapter 4 is possible, but this results in enormous routing complexity. Each of the lines in Fig. 8.10 represents a

Table 8.2: Summary of GSO implementations.

Architecture	Direct mapped	Time mux (TM)	TM w/ memory
Area (silicon/FPGA)	2.6mm^2/60k	1.2mm^2/14k	1.0mm^2/14k
Total wire length	6.6m	3.2m	2.2m

complex vector with a certain number of bits. For example, a vector of dimension 4 and 16 bits for in-phase and quadrature components is equivalent to $4 \times 16 \times 2 = 128$ signal wires in hardware, and this number is further multiplied by the level of time-multiplexing (4 for block N, 6 for block A). So, traditional time-multiplexing results in sizable multiplexing overhead in terms of interconnect complexity.

8.4.3 Routing: Memory as a MUX

A more attractive hardware solution for the GSO implementation is the memory-based time-multiplexed design, as shown in Fig. 8.10(b). This approach eliminates some of the routing overhead from traditional time-multiplexing, by sharing a 128 bit data bus over multiple operands. To illustrate the savings in interconnect as well as area, three different architectures are synthesized: direct-mapped, regular and memory-based time-multiplexing.

The summary of silicon area and interconnect length provided in Table 8.2 shows a 50% reduction in total interconnect length when time-multiplexing is done with memory rather than using the traditional approach. This is beneficial for place and route and also for reduction of the power dissipated in switching of interconnect.

8.5 Estimates from Synthesis

The algorithm for adaptive UΣ decomposition is implemented in silicon. Expected power and timing estimates from synthesis are presented here, followed by an analysis of energy and area efficiency. Measured results are found in the next chapter.

8.5.1 Timing and Power

Timing and power are based on the estimates at nominal supply voltage. Logic synthesis is done at the nominal voltage used for characterization of the standard-cell library, as discussed earlier in this chapter. For correct timing, the design is constrained to meet the timing constraints under the worst case (125C, 0.9V). A standard-V_t process is chosen to allow more leakage in order to balance the leakage and switching components of power, [91]. A 20% timing margin is also included for routing overhead and avoidance of synthesis iterations.

The power estimation from logical and physical synthesis under different process corners is summarized in Table 8.3. Data corresponding to the typical corner is used in further analysis since expected operation is at room temperature. Scaling the supply voltage from 1V down to the desired 0.4V operating point results in a 10× reduction in clock frequency and a 70× reduction in total power. The expected power consumption at the desired operating point (64MHz, 0.4V) is 14mW.

The power estimates are further analyzed down at the block level. Figure 8.11 illustrates the power estimates back-annotated in Simulink. The area of the blocks

Table 8.3: Power estimation from chip synthesis.

Corner	worst (125C, 0.9V)			typ (25C, 1.0V)			best (-40C, 1.1V)		
P(mW)	P_{sw}	P_{lk}	P_{tot}	P_{sw}	P_{lk}	P_{tot}	P_{sw}	P_{lk}	P_{tot}
Synth	550	252	802	659	3.3	662	792	3.7	796
Layout	804	316	1120	971	5	976	1174	6	1180

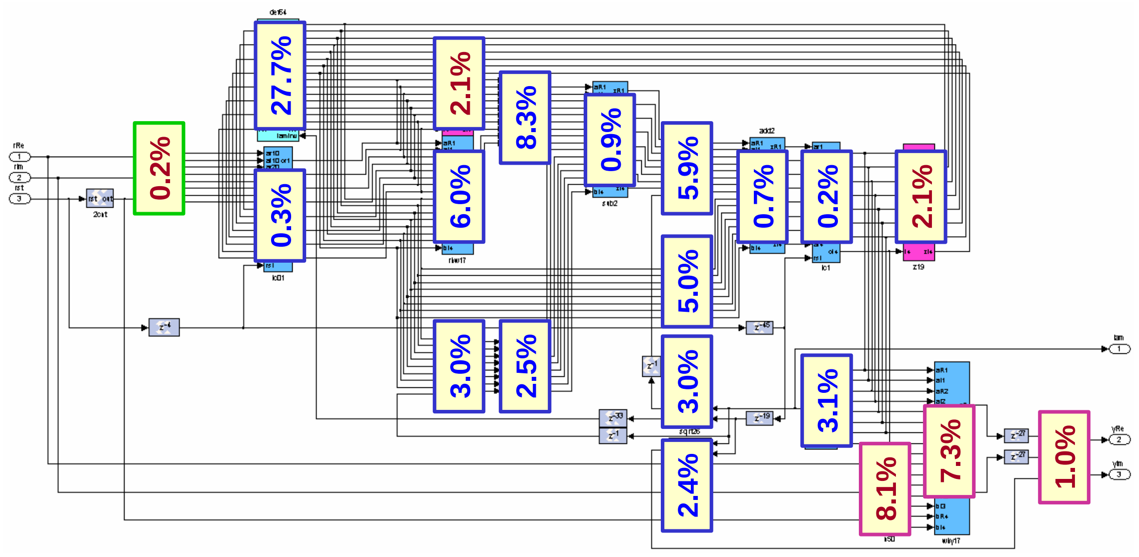


Figure 8.11: Power estimates back-annotated to Simulink environment.

indicated in the figure is proportional to their power since the design is retimed to balance hardware intensity in data-path logic. This partitioning allows for comparison of various building blocks. About 20% of the total power/area is used for data re-ordering that supports interleaving and folding.

8.5.2 Study of Energy and Area Efficiency

The estimates from chip synthesis are used to predict the power and area efficiency of the design. This can be simply done, starting from basic definitions.

Energy Efficiency

Energy efficiency is defined as the amount of energy required to perform an operation. Counting the number of operations and energy per unit time, it turns out that energy and power efficiency are the same, both expressed in MOPS/mW, Eq. (8.2).

$$\begin{aligned}\text{Energy efficiency} &= \frac{\#\text{ operations}}{\text{energy required}} = \frac{\#\text{ operations}}{\text{nJ}} \\ &= \frac{\text{op/s}}{\text{nJ/s}} = \frac{\text{MOPS}}{\text{mW}} = \text{Power efficiency}\end{aligned} \tag{8.2}$$

In this study, a 12-bit equivalent addition is used as the atomic operation. Block characterization shows that an equivalent multiply is about 6 times more complex, in terms of area and power. The total computational complexity of the UΣ block is equivalent to 700 12-bit equivalent *add* operations. At 64MHz, this amounts to 44.8GOPS, resulting in an estimated energy/power efficiency of 3.2GOPS/mW.

To analyze the achieved efficiency, it is instructive to examine the design parameters that affect the energy efficiency the most, as suggested by Eq. (8.3). Neglecting the leakage power (which is estimated at about 0.5% of the total), the efficiency formula simplifies to the inverse of the switching energy. Energy/power efficiency can be thus improved by minimizing the total capacitance $C_{sw,op}$ charged per operation and/or by minimizing the supply voltage V_{dd}.

$$\text{P/E efficiency} \propto \frac{f}{f \cdot C_{sw,op} \cdot V_{dd}^2 + P_{leakage}} \rightarrow \frac{1}{C_{sw,op} \cdot V_{dd}^2} \tag{8.3}$$

The above formula reveals that the highest efficiency per computation is achieved with techniques that allow voltage scaling and down-sizing of the data-path logic. Parallelism and pipelining enable this, while time multiplexing does not. Time-multiplexed architectures also have sizable overhead in data flow control and memory required to support the time-multiplexing, which adds to their inefficiency. A comprehensive study of the energy efficiency in various architectural techniques can be found in [7].

Looking at the energy efficiency alone is misleading, since it can be improved by parallelism with increased area. More complete information about the cost of a design is gathered by looking into the efficiency of silicon area utilization.

Area Efficiency

Area efficiency is defined as the amount of silicon area required to perform an operation per unit time. For highest area efficiency, the goal is to maximize the utilization of data-path logic. This can be done by interleaving streams of data such as in the UΣ example.

By sharing data-path logic, UΣ decomposition is implemented in just $3.5mm^2$ of core chip area in a 90nm CMOS technology (for comparison, the Simulink/XSG model takes 35k FPGA slices). The area related to the pipeline registers is about 40% of the total chip area. The estimated computational complexity of 44.8GOPS translates into a 12.8GOPS/mm^2 area efficiency.

The area efficiency can be improved by increasing the speed of computation, which is implicit to time-multiplexing. This also increases the energy, so both metrics have to be simultaneously considered. The next chapter presents a chip implementation that demonstrates simultaneous power/area optimization.

Chapter 9

Experimental Verification

This chapter presents experimental validation of the concepts developed throughout the dissertation. An ASIC realization of the MIMO baseband processing for a multi-antenna WLAN is described. The chip core which operates at reduced supply voltage is presented, followed by discussion of the on-chip level converter that interfaces the core with the I/O pads operating at standard 1V supply. The experimental Simulink-based setup is then described, followed by an analysis of measured results.

9.1 Test Chip

The chip implements a 4×4 adaptive eigen-mode decomposition algorithm outlined in the previous chapter, with combined power and area minimization achieving a power efficiency of 2.1GOPS/mW in just $3.5mm^2$ in a $90nm$ digital CMOS technology. The die micrograph is shown in Fig. 9.1. The chip is fabricated in a standard-V_t process

Table 9.1: Summary of main optimization techniques.

Opt technique	Area reduction	Energy reduction
Word-length opt	30%	30%
Gate sizing	20%	40%
V_{DD} scaling	N/A	7×
Interleaving / folding	13.8× / 2.6×	−2%

that offers good balance between the leakage and active components of power. Since V_{th} is reduced, a lower V_{dd} is required for the same speed, which results in reduced overall power consumption. The leakage power is measured to be 10% of the total power in active mode.

9.1.1 Eigen-mode Decomposition Core

The chip core is optimized for operation at $400mV$ as discussed in Chapter 8. Scaling of supply voltage is the key to achieving high power/energy efficiency. The main optimization techniques used in this design and their impact on energy and area efficiency are summarized in Table 9.1. The area of the design is substantially reduced through the combined interleaving and folding of the Alg operation (Figs. 8.2, 8.7 in Ch. 8). This is achieved without much penalty in energy (only a 2% increase) since multiplexing overhead is negligible compared to the complexity of the Alg operation.

Physical synthesis is done using an Astro based flow, which achieves high layout density. The first attempt with an 80% initial density could not satisfy the routing constraints, so the initial density of standard cells was then set at 75%. The resulting

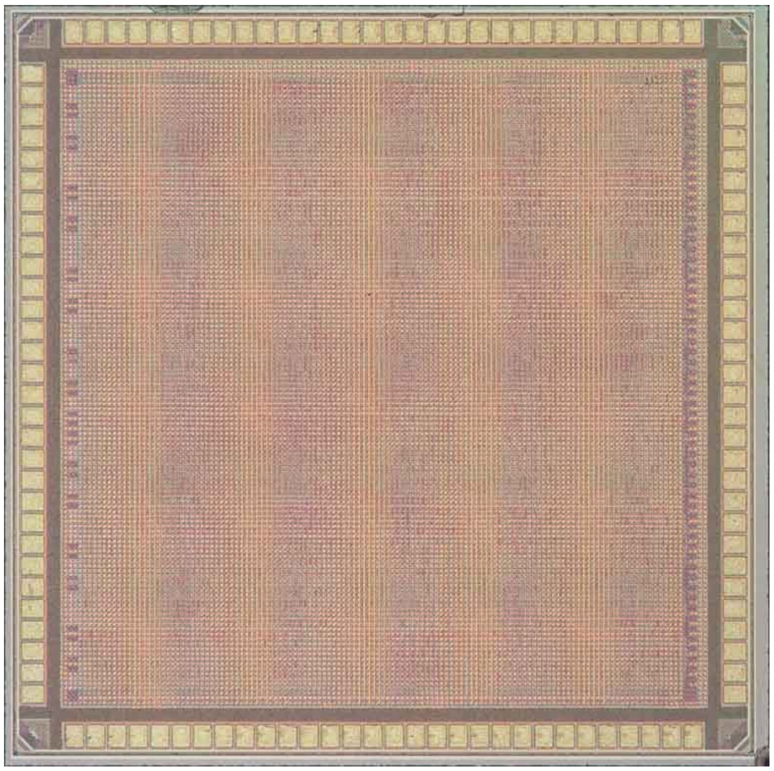


Figure 9.1: Die micrograph of eigen-mode decomposition chip. Die size: $2.3 \times 2.3mm$, chip core: $1.9 \times 1.9mm$. (7M 1P $90nm$ CMOS technology)

layout after routing, clock tree optimization steps, and insertion of hold-time fixing buffers is 88% dense. More details about synthesis steps are found in Appendix A.

9.1.2 On-Chip Level Converter

The standard cross-coupled PMOS level converting circuit [92] is used, as shown in Fig. 9.2. A dual-well approach is used in layout to separate V_{DDL} from V_{DDH} and maintain the standard cell height. Level converting cells are placed at the core boundary, close to the output pads. Level converters are used only at the output, while core inputs are driven with a full-swing signal.

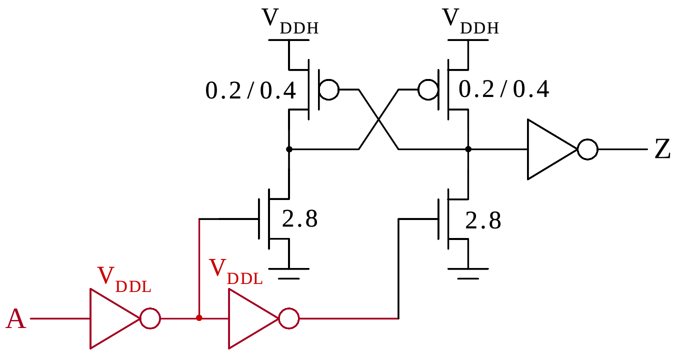


Figure 9.2: Schematic of an on-chip level converter ($V_{DDL} = 0.4V, V_{DDH} = 1V$, numbers indicate transistor W in μm, $L = L_{min}$ unless indicated otherwise).

Chip outputs are delivered to the printed circuit board (PCB) through digital pads which operate at a standard $1V$ supply. An extra analog pad, connected to the input of level converter, is used to allow testing of the level converter. Both the chip and level converter are functionally verified down to 250mV, 10MHz operation.

9.2 Experimental Setup

The Simulink environment is also used for experimental verification. This is the final touch-point between the FPGA and ASIC flows. The Simulink design is programmed into the FPGA which feeds the data into the ASIC and samples outputs for real-time comparison. The Simulink model that enables such an approach is shown in Fig. 9.3. Outputs of the comparison are stored in block RAMs on the FPGA, which can be read out through the serial port. Signals that control the read and write ports of the block RAMs as well as other controls, such as reset and clock enable, are set by the user.

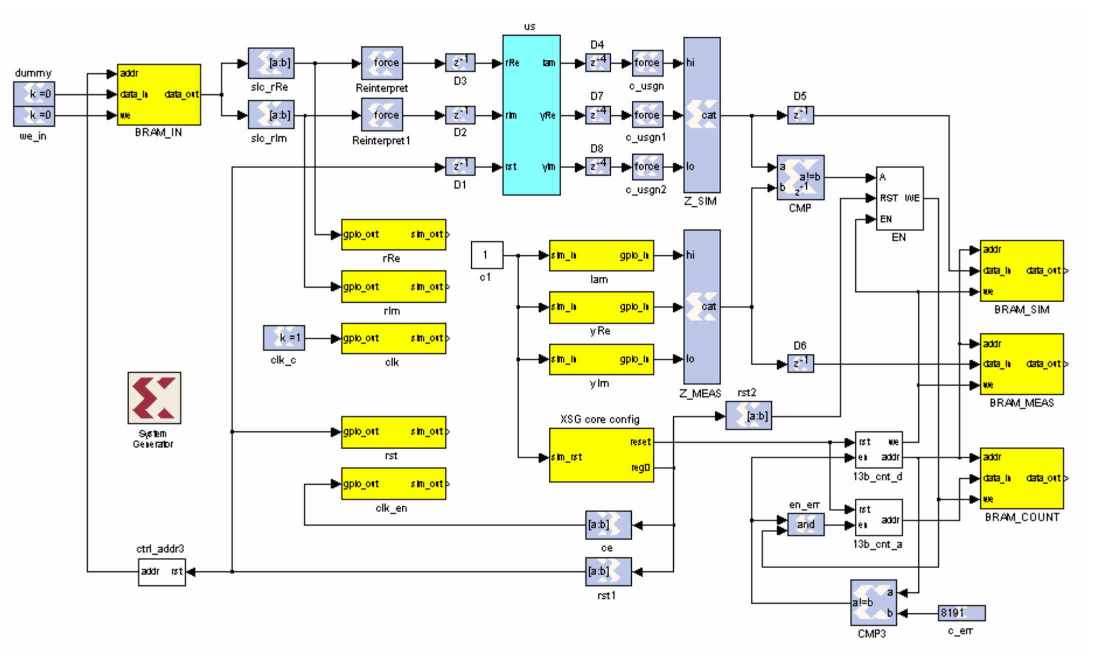


Figure 9.3: Simulink environment for real-time hardware comparison.

Figure 9.4 shows part of the experimental setup that includes the FPGA and chip boards. The FPGA board is an infiniband break-out board (i-BOB) that is a part of much larger system at the BWRC, [93]. In that system, the i-BOB is used as RF interface with an array of FPGAs that emulate the baseband. In this experimental setup, the i-BOB is used purely for signal processing purposes. The FPGA chip (Xilinx Virtex-II Pro) has dedicated 18-bit multipliers, a PowerPC processor, a memory, and a regular FPGA fabric. Dedicated FPGA multipliers are used throughout the design in order to save the FPGA fabric for other functional blocks. With this approach, the entire design fits in one chip quite easily (13,400 slices, less than 50% utilization).

Communication between the i-BOB and the chip board is done through general

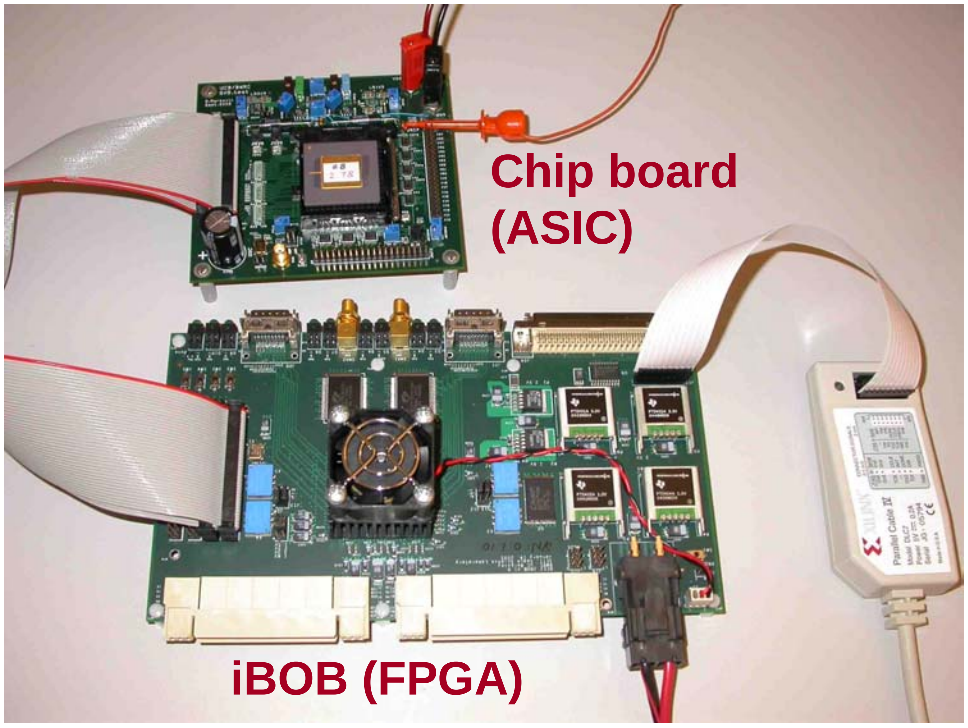


Figure 9.4: Test setup that performs real-time hardware comparison.

purpose I/Os with $3.3V$, single-ended signaling, as required by the FPGA. Level conversion from 3.3V (FPGA) to 1V (ASIC) is realized on the ASIC board in a form of simple resistive divider with 50Ω termination, while up-conversion is done with integrated comparators.

9.3 Measured Results

The result of the adaptive 4×4 eigen-mode decomposition is shown in Fig. 9.5. The chip is trained with a stream of identity matrices, after which it goes into blind tracking mode. In the blind tracking mode, PSK modulated data is sent over the

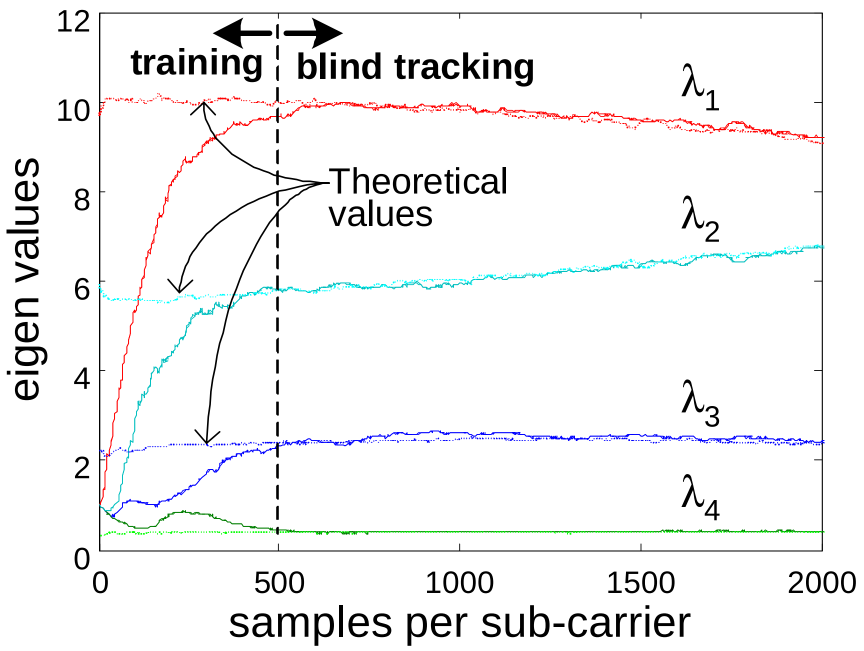


Figure 9.5: Measured tracking of slowly varying eigen-modes in a 4×4 MIMO channel.

antennas with constellations varying according to the estimated SNR, achieving up to a 250Mbps over 16 sub-carriers. The achieved throughput is quite high, considering that the implemented algorithm is limited to use of constant magnitude PSK modulation. On average, 10 bits per symbol are transmitted.

A 100MHz operation is measured at supplies from 385mV to 425mV over 9 die samples, with a 2× variation in leakage power. This performance level corresponds to TT corner (the chip was optimized for 64MHz in SS corner). The leakage and clocking power are 12% and 30% of the total power, respectively. Area and power of functional blocks at 100MHz are given in Table 9.2, where area and power breakdown is based on synthesis estimates. Die features are summarized in Table 9.3.

Table 9.2: Area and power of functional blocks.

100MHz	Power (mW)	Area (mm^2)	GOPS
$U\Sigma$ LMS	20	2.31	42.6
Deflation	14	1.19	27.4

Table 9.3: Chip features. (op = 12-b add)

Technology	90nm CMOS
Core area	$1.9 \times 1.9mm$
Die area	$2.3 \times 2.3mm$
Pad count	120
IO/core V_{DD}	1V / 0.4V
Cell count	420,304
Frequency	100MHz
Power (act/leak)	30mW / 4mW
Power Efficiency	2.1GOPS/mW
Area Efficiency	20GOPS/mm^2

The computational throughput of 70GOPS is implemented with 0.5M gates at a 100MHz clock and 385mV supply, dissipating 34mW of power. Measured power is 30% higher than the gate-level post-layout estimate. With optimal channel conditions the implemented algorithm delivers up to 250Mbps over 16 sub-carriers.

Comparison to Prior Work

Using derivations in Chapter 6, Table 6.1, the resulting FOM of 8.1 pJ/bps/Hz is achieved, which is about 2× better than the reference point established in prior work. More details for existing designs, such as the number of operations, is needed for accurate comparison.

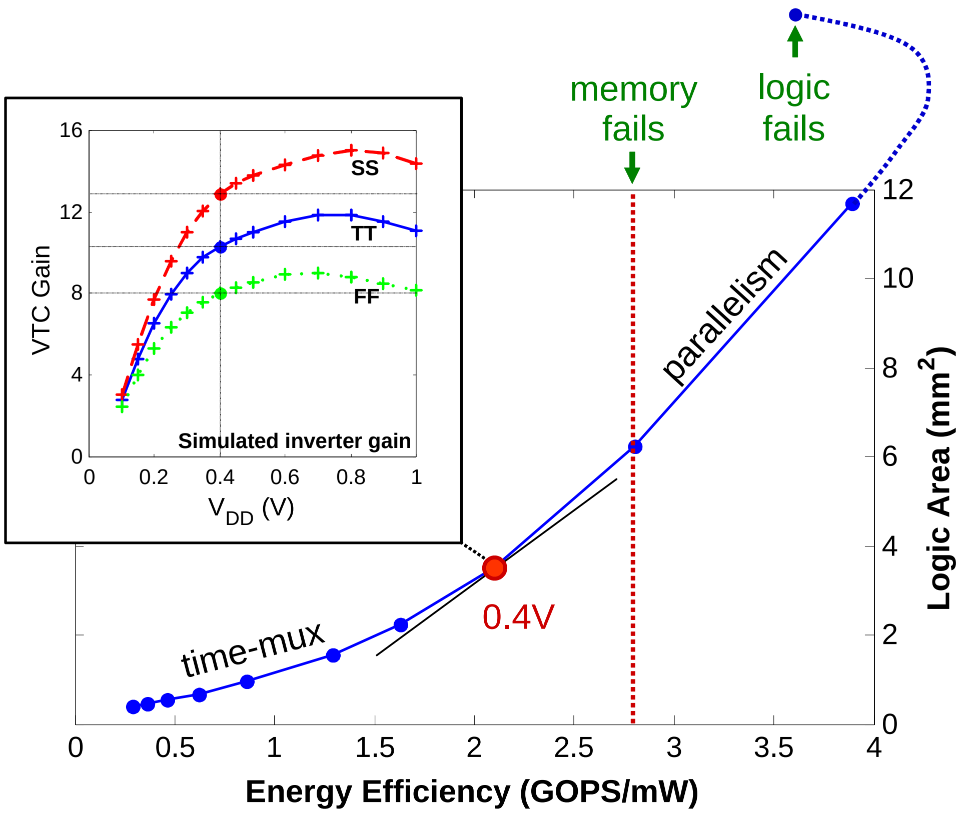


Figure 9.6: Trade-off between Energy-Efficiency and Logic Area in the test design.

Exploration of Energy-Area Space

The design optimized for 0.4V is just a single point in the energy-area space. It can be optimized for a different voltage using a different architecture, based on energy-area tradeoff discussed in Chapter 4.

Taking into account the area of logic gates and pipeline registers in the experimental design, Figure 9.6 plots the trade-off between energy-efficiency and logic area (including pipeline registers). The dot indicates a design optimized for 0.4V operation. Starting from this point, energy-efficiency can be traded for smaller area by increasing the level of time-multiplexing, which requires increase in supply voltage and hence reduced energy-efficiency. This graph is not completely accurate, because it neglects area overhead of time-multiplexing and parallelism, but does indicate a general trend.

Going in the direction of increased parallelism, the energy-efficiency can is improved by further scaling the supply voltage, but the area increases with the number of parallel units. However, there is a limit to supply voltage scaling. As V_{dd} scales down, at some point memory will fail, followed by a failure in flip-flops, and finally the logic gates. Another effect illustrated in Fig. 9.6 is that at some point leakage energy will start rapidly increasing due to increased delay at very low voltage. The energy-efficiency peaks at the point of minimum energy. With further V_{dd} scaling, overall energy increases, so the energy-efficiency decreases despite increased area.

The result in Fig. 9.6 is quite meaningful in the optimization of power constrained designs. When power is limited, which is often the case in practical designs, the area can be minimized as follows. Given the power limit, desired power efficiency is calculated from the required amount of functionality, using a known relationship between the energy per operation and supply voltage. This determines the desired operating point on E-D line. The optimization procedure then applies appropriate transformations to reach the desired point.

Chapter 10

Conclusion

This dissertation presents a sensitivity based optimization framework for power and area efficient VLSI realization of signal processing algorithms for wireless communications. The choice of architecture is highly influenced by the power-speed capability of the underlying technology. It is thus important to obtain these technology estimates as early in the design process as possible. Characterization of a few basic blocks from the Simulink hardware library provides much needed information, so that optimal architecture can be chosen early at the Simulink level, without the need for extra feedback information. The design process starts from the algorithm description in a simple Matlab-Simulink environment, followed by an automated flow for rapid FPGA evaluation and VLSI implementation. Several examples commonly found in communication signal processing are demonstrated. A power efficiency of a 2.1GOPS/mW is experimentally verified in the 4×4 SVD example.

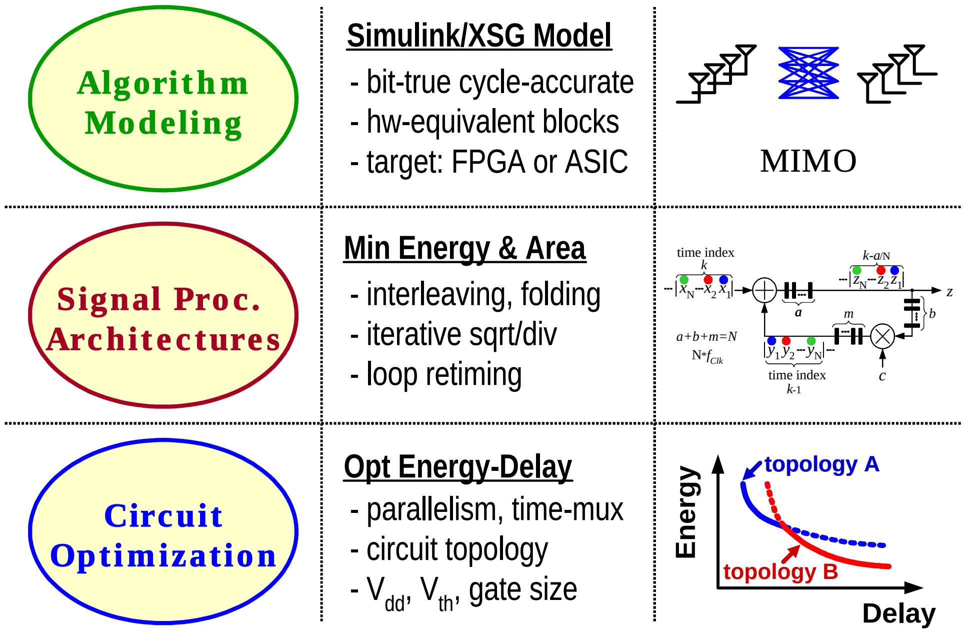


Figure 10.1: Design space exploration.

10.1 Research Contributions

The goal of this research is to significantly improve power/area efficiency in wireless baseband signal processing through effective design optimization across the boundary of algorithm, architecture, and the underlying circuits. This dissertation presents a methodology that allows for large design space exploration including algorithm modeling, signal processing architectures, and circuits, as illustrated in Fig. 10.1. All of these areas are combined under a unified framework for rapid hardware evaluation and implementation of very complex signal processing algorithms. Several key contributions which address the goal of this research are:

- Developed a sensitivity based optimization framework that provides insight into the effectiveness of various tuning variables in the design. A fixed point in the optimization is reached when marginal returns to all variables are balanced.

- Applied the sensitivity framework to minimize the energy consumption subject to a delay constraint at the circuit level. Multi-variable optimization of gate size, supply and threshold voltage is applied to an inverter chain, a memory decoder, and an adder.

- Demonstrated significant energy savings at the circuit level: a 50-70% of energy reduction can be achieved with only 10% excess delay, compared to the design optimized for minimum delay through gate sizing. It has been also demonstrated that significant energy reduction can be achieved without delay penalty, by simply balancing the sensitivities to all variables in the design.

- Provided insight into effectiveness of various optimization variables at the circuit level, starting from the design sized for minimum delay: 1) gate sizing is the most effective for small incremental delays, up to about 20% excess delay due to infinite sensitivity at the minimum delay, 2) supply voltage scaling is the most effective for large incremental delays, above 20%.

- Detailed analysis of micro-architectural techniques of parallelism, pipelining, and time-multiplexing revealed that the energy is best minimized when the leakage and switching components of energy are about equal. The minimum of the total energy is fairly broad in the range of E_{Lk}/E_{Sw} from 0.1 to 10.

- Established formalism for architectural selection in the energy-area space, for a fixed throughput, as demonstrated with an ALU design implemented with varying degrees of parallelism and time-multiplexing.

- Exported main optimization results from the circuit level into a commercial chip design flow: gate sizing is exploited by synthesizing designs with a 20% slack, supply voltage is exploited by constraining the design with an equivalent specification at the reference voltage for a technology.
- Developed a Matlab/Simulink model for an adaptive 4×4 SVD algorithm that evaluates algorithm performance under hardware constraints, such as latency and finite word-lengths.
- Developed heuristic for hierarchical word-size selection and applied it to the SVD algorithm. This was necessary due to limited memory resources in a simulation-based perturbation approach.
- Developed a methodology for hierarchical loop retiming in Simulink-based ASIC chip design flow. The proposed approach allows optimization of macro-architecture at the Simulink level. Block based approach also significantly improves runtime compared to top-level retiming done by synthesis tools.
- Applied aforementioned concepts to design and power/area optimal implementation of an adaptive 4×4 SVD algorithm over 16 sub-carriers. Prototype silicon achieves the energy-efficiency of $2.1 GOPS/mW$ and area efficiency of $20 GOPS/mm^2$, running at a 100MHz clock under 385mV supply voltage.

- Developed FPGA assisted ASIC test and debug methodology that enables ASIC testing through real-time hardware co-simulation. Test vectors are exported from Simulink, thereby greatly improving design efficiency and shortening the design cycle.

The methodology established in this dissertation therefore enables rapid ASIC implementation of complex signal processing algorithms, from algorithm description, through design optimizations, and final ASIC verification, within a unified Matlab/Simulink environment.

10.2 Future Work

This dissertation has provided the framework for a variety of continuing research directions. Further research is required on flexible architectures and CAD flow methodologies addressing energy and power efficiency in future wireless devices.

Looking forward, it is interesting to consider the cost of adding flexibility in multi-standard wireless devices. Two possible directions are as follows: investigate the flexibility required for the execution of common operations across multiple standards and study the flexibility of having multi-functionality on a single hardware unit. Study of various baseband algorithms is required to identify common computational kernels that need to be optimized for power and area. Additional research is needed to quantify the cost of varying degrees of flexibility on power and area.

As process technology continues to advance, the impact of interconnect on power and area becomes more pronounced. Thus, further investigation of energy reduction principles related to interconnect, such as low-swing signaling, is needed. Similarly, with shrinking device dimensions, the impact of process parameter variations on power and performance is increasing. Robust circuit design methodologies have to be therefore developed to reduce the impact of variations.

As an integral part of the design cycle, new concepts in VLSI design have to be ported over to automated CAD environment, to improve the design productivity. Additional research is needed to investigate power-area-speed trade-offs in logic blocks and memories in deeply scaled technologies, to determine optimal architectures for future wireless devices.

Bibliography

[1] C. Shi, *Floating-point to Fixed-point Conversion.* PhD thesis, University of California, Berkeley, Spring 2004.

[2] `http://www.intel.com/multi-core/index.htm`

[3] S. Naffziger, B. Stackhouse, and T. Grutkowski, "The Implementation of a 2-core Multi-Threaded Itanium®-Family Processor," in *Proc. International Solid-State Circuits Conference, ISSCC'05*, pp. 182–183, Feb. 2005.

[4] G. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.

[5] J. Thomson *et al.*, "An Integrated 802.11a Baseband and MAC Processor," in *Proc. International Solid-State Circuits Conference, ISSCC'02*, pp. 126–127, Feb. 2002.

[6] `http://grouper.ieee.org/groups/802/11/`

[7] R. Brodersen, "Technology, Architecture, and Applications," in *Proc. International Solid-State Circuits Conference, ISSCC'02, Special Topic Evening Session: Low Voltage Design for Portable Systems*, Feb. 2002.

[8] C. Mead and L. Conway, *Introduction to VLSI Design.* Reading, MA: Addison-Wesley, 1980.

[9] J. Burr and A. Peterson, "Ultra Low Power CMOS Technology," in *Proc. NASA VLSI Design Symposium*, pp. 4.2.1–4.2.13, Oct. 1991.

[10] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, Aug. 1997.

[11] P. Penzes and A. Martin, "Energy-Delay Efficiency of VLSI Computations," in *Proc. Great Lakes Symposium on VLSI, GLSVLSI'02*, pp. 104–111, Apr. 2002.

[12] H. Hofstee, "Power-Constrained Microprocessor Design," in *Proc. International Conference on Computer Design, ICCD'02*, pp. 14–16, Sept. 2002.

[13] A. Martin, "Towards an Energy Complexity of Computation," *Information Processing Letters*, vol. 77, pp. 181–187, Feb. 2001.

[14] J. Fishburn and A. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," in *Proc. International Conference on Computer-Aided Design, ICCAD'85*, pp. 326–328, Nov. 1985.

[15] A. Conn, R. Haring, C. Visweswariah, P. Coulman, and G. Morrill, "Optimization of Custom MOS Circuits by Transistor Sizing," in *Proc. International Conference on Computer-Aided Design, ICCAD'96*, pp. 174–180, Nov. 1996.

[16] A. Conn *et al.*, "Gradient-Based Optimization of Custom Circuits Using a Static-Timing Formulation," in *Proc. Design Automation Conference, DAC'99*, pp. 452–459, June 1999.

[17] H. Lin and L. Linholm, "An Optimized Output Stage for MOS Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-10, no. 2, pp. 106–109, Apr. 1975.

[18] S. Ma and P. Franzon, "Energy Control and Accurate Delay Estimation in the Design of CMOS Buffers," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 9, pp. 1150–1153, Sept. 1994.

[19] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[20] T. Kuroda *et al.*, "Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, pp. 454–462, Mar. 1998.

[21] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "Dynamic Voltage Scaled Microprocessor System," in *Proc. IEEE International Solid-State Circuits Conference, ISSCC'00*, pp. 294–295, Feb. 2000.

[22] D. Liu and C. Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltage," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 1, pp. 10–17, Jan. 1993.

[23] K. Nose and T. Sakurai, "Optimization of Vdd and Vth for Low-Power and High-Speed Applications," in *Proc. Asia South Pacific Design Automation Conference, ASP-DAC'00*, pp. 469–474, Jan. 2000.

[24] T. Gemmeke, M. Gansen, H. Stockmans, and T. Noll, "Design Optimization of Low-Power High-Performance DSP Building Blocks," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 7, pp. 1131–1139, July 2004.

[25] V. Stojanovic, D. Markovic, B. Nikolic, M. Horowitz, and R. Brodersen, "Energy-Delay Tradeoffs in Combinational Logic using Gate Sizing and Supply Voltage Optimization," in *Proc. European Solid-State Circuits Conference, ESSCIRC'02*, pp. 211–214, Sept. 2002.

[26] R. Brodersen, M. Horowitz, D. Markovic, B. Nikolic, and V. Stojanovic, "Methods for True Power Minimization," in *Proc. International Conference on Computer-Aided Design, ICCAD'02*, pp. 35–42, Nov. 2002.

[27] T. Sakurai and R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, Apr. 1990.

[28] R. Rao, J. Burns, A. Devgan, and R. Brown, "Efficient Techniques for Gate Leakage Estimation," in *Proc. IEEE International Symposium on Low-Power Electronics and Design, ISLPED'03*, pp. 100–103, Aug. 2003.

[29] International Technology Roadmap for Semiconductors, `http://public.itrs.net`

[30] V. Zyuban and P. Strenski, "Unified Methodology for Resolving Power-Performance Tradeoffs at the Microarchitectural and Circuit Levels," in *Proc. International Symposium on Low Power Electronics and Design, ISLPED'02*, pp. 166–171, Aug. 2002.

[31] B. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and Sizing for Minimum Energy Operation in Subthreshold Circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, Sept. 2005.

[32] J. Burr and J. Shott, "A 200mV Self-Testing Encoder/Decoder using Stanford Ultra-Low-Power CMOS," in *Proc. IEEE International Solid-State Circuits Conference, ISSCC'94*, pp. 84–85, Feb. 1994.

[33] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of General Class of Recurrence Equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[34] Z. Huang and M. Ercegovac, "Effect of Wire Delay on the Design of Prefix Adders in Deep-Submicron Technology," in *Proc. 34th Asilomar Conference on Signals, Systems and Computers*, pp. 1713–1717, Oct. 2000.

[35] J. Tschanz *et al.*, "Comparative Delay and Energy of Single Edge-Triggered & Dual Edge-Triggered Pulsed Flip-Flops for High-Performance Microprocessors," in *Proc. International Symposium on Low Power Electronics and Design, ISLPED'01*, pp. 147–152, Aug. 2001.

[36] G. Gerosa *et al.*, "A 2.2 W, 80 MHz Superscalar RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 12, pp. 1440–1454, Dec. 1994.

[37] I. Sutherland, B. Sproul, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan Kaufmann, 1st ed., 1999.

[38] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ: Prentice Hall, 2nd ed., 2003.

[39] R. Zlatanovici and B. Nikolic, "Power-Performance Optimal 64-bit Carry-Lookahead Adders," in *Proc. European Solid-State Circuits Conference, ESSCIRC'03*, pp. 321–324, Sept. 2003.

[40] V. Oklobdzija, B. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-Delay Estimation Techniques for High-Performance Microprocessor VLSI Adders," in *Proc. International Symposium on Computer Arithmetic, ARITH-16*, pp. 272–276, June 2003.

[41] S. Kao, R. Zlatanovici, and B. Nikolic, "A 250ps 64-bit Carry-Lookahead Adder in 90nm CMOS," in *Proc. IEEE International Solid-State Circuits Conference, ISSCC'06*, pp. 438–439, Feb. 2006.

[42] H. Ling, "High Speed Binary Adder," *IBM Journal of Research and Development*, vol. 25, no. 3, pp. 156–166, May 1981.

[43] T. Burd, *Energy-Efficiency Processor System Design*. PhD thesis, University of California, Berkeley, Spring 2001.

[44] M. Hamada, Y. Ootaguro, and T. Kuroda, "Utilizing Surplus Timing for Power Reduction," in *Proc. IEEE Custom Integrated Circuits Conference, CICC'01*, pp. 89–92, Sept. 2001.

[45] H. Kawaguchi and T. Sakurai, "A Reduced Clock-Swing Flip-Flop (RCSFF) for 63% Power Reduction," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 4, pp. 807–811, May 1998.

[46] Y. Shimazaki, R. Zlatanovici, and B. Nikolic, "A Shared-Well Dual-Supply-Voltage 64-bit ALU," in *Proc. IEEE International Solid-State Circuits Conference, ISSCC'03*, pp. 104–105, Feb. 2003.

[47] V. Oklobdzija, V. Stojanovic, D. Markovic, and N. Nedovic, *Digital System Clocking: High-Performance and Low-Power Aspects*. Hoboken, NJ: IEEE Press, Wiley-Interscience, 1995.

[48] A. Chandrakasan, *Low Power Digital CMOS Design*. PhD thesis, University of California, Berkeley, Summer 1994.

[49] K. Parhi, *VLSI Digital Signal Processing Systems.* New York, NY: John Wiley & Sons, 1999.

[50] V. Srinivasan and R. Vemuri, "A Retiming Based Relaxation Heuristic for Resource-Constrained Loop Pipelining," in *Proc. Eleventh International Conference on VLSI Design: VLSI for Signal Processing, VLSID'98*, vol. 2, pp. 435–441, 1998.

[51] Y. Yu, R. Woods, L. Ting, and C. Cowna, "High Sampling Rate Retimed DLMS Filter Implementation in Virtex-II FPGA," in *IEEE Workshop on Signal Processing Systems, SiPS'02*, pp. 139–145, Oct. 2002.

[52] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication.* Cambridge, MA: Cambridge University Press, 1st ed., 2005.

[53] G. Raleigh and J. Cioffi, "Spatio-Temporal Coding for Wireless Communication," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 357–366, Mar. 1998.

[54] G. Foschini, "Layered Space-Time Architecture for Wireless Communication in a Fading Environment When Using Multi-Element Antennas," *Bell Labs Technical Journal*, pp. 41–59, Autumn 1996.

[55] D. Rabideau, *Computational Methods for Real-Time Adaptive Multichannel Signal Processing.* PhD thesis, Cornell University, Jan. 1995.

[56] A. Poon, D. Tse, and R. Brodersen, "An Adaptive Multiple-Antenna Transceiver for Slowly Flat-Fading Channels," *IEEE Transactions on Communications*, vol. 51, no. 13, pp. 1820–1827, Nov. 2003.

[57] A. Poon, *Use of Spatial Dimension for Spectrum Sharing.* PhD thesis, University of California, Berkeley, Spring 2004.

[58] S. Oberman and M. Flynn, "Division Algorithms and Implementations," *IEEE Transactions on Computers*, vol. 47, no. 8, pp. 833–854, Aug. 1997.

[59] T. Lang and E. Antelo, "CORDIC Vectoring with Arbitrary Target Value," *IEEE Transactions on Computers*, vol. 47, no. 7, pp. 736–749, July 1998.

[60] C. Ramamoorthy, J. Goodman, and K. Kim, "Some Properties of Iterative Square-Rooting Methods Using High-Speed Multiplication," *IEEE Transactions on Computers*, vol. C-21, no. 8, pp. 837–847, 1972.

[61] L. Zheng and D. Tse, "Diversity and Multiplexing: A Fundamental Tradeoff in Multiple-Antenna Channels," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[62] S. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[63] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-Time Block Code from Orthogonal Designs," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1456–1467, July 1999.

[64] G. Foschini, G. Golden, R. Valenzuela, and P. Wolniansky, "Simplified Processing for High Spectral Efficiency Wireless Communication Employing Multi-Element Arrays," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[65] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 2nd ed., 1989.

[66] M. Greenberg, *Differential Equations and Linear Algebra*. Upper Saddle River, NJ: Prentice Hall, 2001.

[67] G. Strang, *Introduction to Linear Algebra*. Wellesley, MA: Wellesley-Cambridge Press, 1998.

[68] L. Chang and C. Yeh, "Performance of DMI and Eigenspace-Based Beamformers," *IEEE Transactions on Antennas Propagation*, vol. 40, no. 11, pp. 1336–1347, Nov. 1992.

[69] D. Rabideau, "Fast, Rank Adaptive Subspace Tracking and Applications," *IEEE Transactions on Signal Processing*, vol. 44, no. 9, pp. 2229–2244, Sept. 1996.

[70] E. Viterbo and J. Boutros, "A Universal Lattice Code Decoder for Fading Channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[71] M. Damen, A. Chkeif, and J. Belfiore, "Lattice Codes Decoder for Space-Time Codes," *IEEE Communication Letters*, vol. 4, pp. 161–163, May 2000.

[72] H. Gamal, G. Caire, and M. Damen, "Lattice Coding and Decoding Achieve the Optimal Diversity-Multiplexing Tradeoff in MIMO Channels," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 968–985, June 2004.

[73] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative Decoding for MIMO Channels Via Modified Sphere Decoding," *IEEE Transactions on Wireless Communications*, vol. 3, no. 6, pp. 2299–2311, Nov. 2004.

[74] D. Gesbert, M. Shafi, D. Shiu, P. Smith, and A. Naguib, "From Theory to Practice: An Overview of MIMO Space-Time Coded Wireless Systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[75] M. Damen, H. Gamal, and G. Caire, "On the Complexity of ML Detection and the Search for the Closest Lattice Point," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 2400–2414, Oct. 2003.

[76] H. Vikalo and B. Hassibi, "On Sphere Decoding Algorithm. II. Generalizations, Second-Order Statistics, and Applications to Communications," *IEEE Transactions on Signal Processing*, pp. 2819–2834, Aug. 2005.

[77] B. Hassibi and H. Vikalo, "On Sphere Decoding Algorithm. I. Expected Complexity," *IEEE Transactions on Signal Processing*, pp. 2806–2818, Aug. 2005.

[78] E. Zimmermann, W. Rave, and G. Fettweis, "On the Complexity of Sphere Decoding," in *Proc. International Symposium on Wireless Personal Multimedia Communications, WPMC'04*, vol. 2, pp. 500–504, Sept. 2004.

[79] A. Burg *et al.*, "VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

[80] D. Garrett, G. Woodward, L. Davis, and C. Nicol, "A 28.8 Mb/s 4×4 MIMO 3G CDMA Receiver for Frequency Selective Channels," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 320–330, Jan. 2005.

[81] Z. Guo and P. Nilsson, "A VLSI Architecture for the Schnorr-Euchner Decoder for MIMO Systems," in *Proc. IEEE CAS Symposium on Engineering Technologies: Frontiers of Mobile and Wireless Communications*, pp. 65–68, June 2004.

[82] Z. Guo, *MIMO Decoding, Algorithm and Implementation.* PhD thesis, Lund University, Sweden, 2005.

[83] Z. Guo and P. Nilsson, "A VLSI Implementation of MIMO Detection for Future Wireless Communications," in *Proc. IEEE Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'03*, pp. 2852–2856, Sept. 2003.

[84] `http://systemc.org`

[85] `http://www.xilinx.com`

[86] C. Shi and R. Brodersen, "Automated Fixed-point Data-type Optimization Tool for Signal Processing and Communication Systems," in *Proc. IEEE Design Automation Conference, DAC'04*, pp. 478–483, June 2004.

[87] `http://bwrc.eecs.berkeley.edu/Research/Insecta/default.htm`

[88] J. Proakis, *Digital Communications*. New York, NY: McGraw Hill, 3rd ed., 1995.

[89] H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, and J. Rabaey, "Standby Supply Voltage Minimization for Deep Sub-Micron SRAM," *Elsevier Science Microelectronics Journal*, vol. 36, no. 9, pp. 789–800, Sept. 2005.

[90] H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, and J. Rabaey, "SRAM Leakage Suppression by Minimizing Standby Supply Voltage," in *Proc. International Symposium on Quality Electronic Design, ISQED'04*, pp. 55–60, Mar. 2004.

[91] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Brodersen, "Methods for True Energy-Performance Optimization," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, Aug. 2004.

[92] K. Usami *et al.*, "Automated Low-Power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, Mar. 1998.

[93] `http://bwrc.eecs.berkeley.edu/Research/BEE/BEE2/index.htm`

Appendix A

Chip Synthesis & Test

A.1 CPU Runtime

Complete physical synthesis of the chip in Fig. 9.1 completed in 26 hrs ($\sim$ 20,000 gates / hour). CPU runtime details are summarized in Table A.1.

Table A.1: CPU runtime of back-end steps for chip in Fig. 9.1.

Design phase	CPU runtime (hr)	Total runtime (hr)
Import	0:05	0:05
Floorplan	0:10	0:15
Placement	3:30	3:45
Tree synthesis	0:30	4:15
Post CTS	2:30	6:45
Route	3:30	10:45
Post route	15:00	25:15
Finish	1:00	26:15

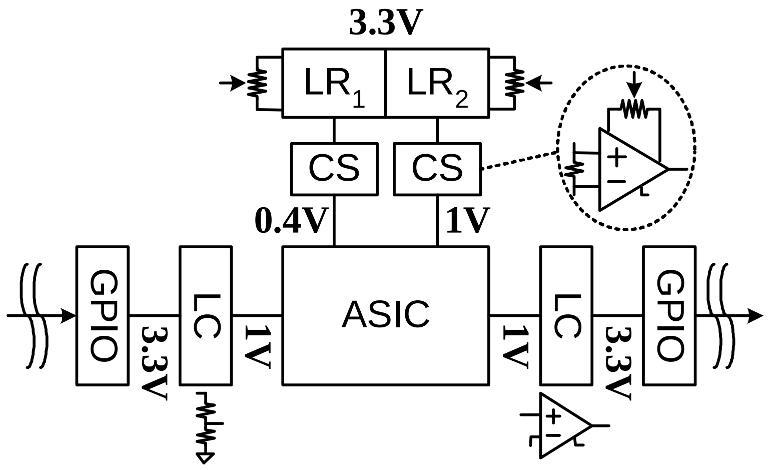


Figure A.1: Conceptual diagram of ASIC test board.

A.2 Test Methodology

The ASIC test board is sketched in Fig. A.1. The board has thee power supply domains: a board-level 3.3V supply which is compatible with general purpose I/O (GPIO) pins that interface with the FPGA board, a 1V supply for chip I/Os, and a 0.4V supply for the ASIC core. Both 1V and 0.4V supplies are generated using tunable linear regulators (LR). A resistive divider bank is used to down-shift inputs to the chip, while active comparators are used to up-convert outputs back to 3.3V. Power is measured using current sensing (CS) circuitry consisting of an instrumentation amplifier with a tunable gain and the output level adjusted with a voltage reference.