DESIGN OF ROBUST ENERGY-EFFICIENT DIGITAL CIRCUITS USING GEOMETRIC PROGRAMMING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dinesh Patil

March 2008

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mark Horowitz) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Stephen Boyd)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(S. S. Mohan)

Approved for the University Committee on Graduate Studies.

Abstract

Power dissipation has become a critical design constraint in all digital systems. Designers must focus on creating energy-efficient circuits to achieve the highest performance within a specified energy budget or dissipate the lowest energy for a given performance. In addition, as technology scales, increasing process variations can significantly degrade circuit timing and power. These variations must be accounted for during design to produce robust circuits that guarantee a desired performance after fabrication.

This thesis focuses on the automated design of robust energy-efficient digital circuits. Using device sizes, and supply and threshold voltages as variables, we formulate the energy-efficient circuit design problem as a Geometric Program. GPs are a special class of convex optimization problems and can be solved efficiently. We develop analytical models of gate delay and energy, and include different design scenarios like changing logic styles, discrete threshold voltages, wire resistances and capacitances, signal slope constraints and so on in the optimization. To facilitate design entry and post-optimization analysis, we have built the Stanford Circuit Optimization Tool (SCOT). As a design case study we explore the energy-delay tradeoff of different 32bit adder topologies using SCOT. These tradeoff curves show that adders with an average fanin of two per stage having the fewest logic stages and smallest wire overhead are most energy-efficient.

Gate delay uncertainties due to process variations cause a spread in the overall circuit

delay. We show how deterministic sizing, which optimizes the nominal delay ignoring process variations, can make the overall delay much worse under variations because it results into many critical paths that may contain small devices.

To solve this problem, we propose two heuristics that guide the optimizer to create a solution that comes close to optimizing the performance that guarantees a desired yield. First, we augment the gate delay models with standard deviation delay margins. Second, we use a "soft max" function to combine path delays at converging nodes. Using these heuristics retains the original GP form of deterministic sizing and therefore incurs only a modest computational overhead. The improvement in robustness over deterministic sizing depends on the circuit topology and the extent of variations specified by the technology. For a 90nm technology, assuming a 15% standard deviation in the delay of a 1μ wide drive transistor, results show that using the proposed heuristic techniques of statistical sizing with the correct statistical estimate of overall energy improves the energy-delay tradeoff curve by 10% for 32-bit adders.

Acknowledgements

In all these years of my Ph.D., I have gained immensely in knowledge and perspective from many people. I feel fortunate to have had the opportunity of working with some of the best people in my field and to have brushed shoulders with some of the brightest students.

First and foremost, I would like to thank my adviser, Prof. Mark Horowitz, who, quite simply, taught me "how to think". I am constantly amazed by his depth of understanding and I am grateful for his patience during all the time I took to understand new ideas, get the big picture and learn habits of effective research. His honest critique has taught me to be a good reviewer of my own work. I only hope that someday I will come close to his level of thinking. I also had the privilege of working with Prof. Stephen Boyd, who mentored me as my co-adviser. Along with his enthusiasm and his valuable insights in optimization, he taught me the value of consistency and precision in technical writing and presentation that makes for lucid, beautiful papers. I am very grateful for his excellent teaching and support throughout my research. I would also like to thank Dr. S. S. Mohan for reviewing my thesis and inspiring me with his constant smile and industry, both during my internship at Barcelona Design Inc. and during my thesis writing. I am thankful to Prof. Krishna Saraswat and Prof. Thomas Lee for many insightful discussions I have had with them.

My incredible research group at Stanford consists of brilliant people from many different technical backgrounds. I have learned a lot from their work habits. I thank my

seniors Ken Mai, Ron Ho, Dean Liu and Vladimir Stojanovic for initializing me into circuit design and for their invaluable guidance and tips. Ron continues to be my mentor at Sun Microsystems. I thank my group-mates Samuel Palermo, Hae-Chang Lee, Elad Alon, Vicky Wong, Valentin Abramzon, Amir Amirkhany, Bita Nezamfar, Francois Labonte, Jim Weaver, Alex Solomatnikov, Amin Firoozshahian, Xiling Shen and Omid Azizi, for all their company and friendship during courses, research projects and out-door trips. I would also like to thank Sunghee Yun and Seung-Jean Kim from Prof. Boyd's research group for their invaluable support and contribution towards my research.

Special thanks are due to Teresa Lynn and Penny Chumley for being great admins and promptly resolving my administrative issues with amazing ease. I am grateful to MARCO C2S2 for funding my research and providing a platform for collaborating with the industry. I would also like to thank Dr. Arash Hassibi, for giving me the opportunity of interning at Barcelona Design Inc.

Words cannot express my gratitude to all my friends at Stanford. They have enriched my life with many wonderful thoughts and inspiring ideas. I thank them for making my stay on campus so enjoyable and memorable. I thank the members of Dhwani, the Indian music group at Stanford, for the great times I have had playing some beautiful music, and Pt. Habib Khan, for sharing his knowledge of music with me. I thank Poorvi for her understanding and support, while I stole time from her to give to my thesis.

Finally, I would like to offer my gratitude to my mom Nisha Patil, my dad Dilip Patil and my grand parents. Every success of my life is ultimately the consequence of their relentless efforts to see that I get the best of education. I dedicate this thesis to them.

Contents

List of Tables

List of Figures

List of Symbols and Acronyms

V_{dd}	Supply voltage	6
V_{th}	Threshold voltage	3
V_{thN}	NMOS threshold voltage	43
V_{thP}	PMOS threshold voltage	43
t_{ox}	Gate oxide thickness	19
v_{sat}	Saturation velocity of carriers in silicon	9
C_{ox}	Gate oxide capacitance	8
V_{od}	Gate overdrive (V_{dd} - V_{th})	9
E_{c}	Critical electrical field	9
L_{min}	Minimum gate length	16
I_{d}	Transistor drive current	6
I_{dsat}	Transistor saturation current	8
I_{leak}	Leakage current	6
SEC	Single Error Correction	73
DED	Double Error Detection	73

Chapter 1

Introduction

For the past three decades, technology scaling has enabled us to develop faster, smaller, cheaper logic gates [60], leading to their use in many systems spanning a wide range of power and performance. Figure 1.1 shows a few example systems starting from the very low power battery operated devices like hearing aids and pacemakers on the lower right to high-power high-performance servers and mainframes on the upper left. The power consumption of a pacemaker is around 10μW [97], while modern high end processors dissipate around 100W [57, 50]. Even though the power consumption of these systems differ by over 6 orders of magnitude, energy-efficiency is a crucial factor in all of them. In portable devices, power dissipation directly affects the battery life, therefore these systems have to minimize their energy consumption while delivering the required performance. For high performance devices, maximum energy dissipation is constrained by power delivery and cooling system costs. Not only has it become increasingly difficult to supply the high currents needed by high-end microprocessors, the cost effective air cooling limit, which sits at around 100W, means that to increase the performance, we need to decrease the energy per operation to keep the chip power dissipation within this limit. The goal in these systems is

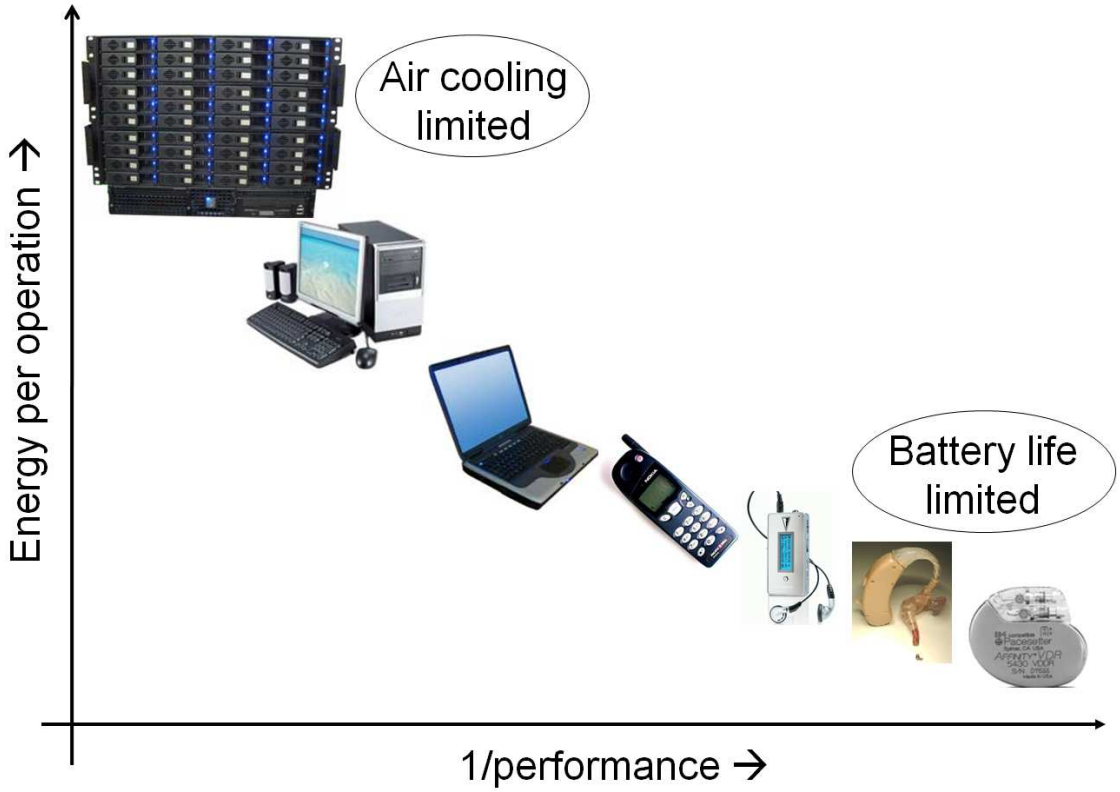


Figure 1.1: Growing number of digital systems across a wide power-performance range

to maximize the performance under the system energy constraints. Consequently, instead of designing for highest performance or lowest power, designers need to make "energy-efficient" systems which consume the least energy for a desired performance or deliver the highest performance for a given energy budget.

What makes this problem even more challenging is that with shrinking feature size, local random process variations have an increasing effect on the performance and energy of digital circuits. The amount of guard banding needed to accommodate variations traditionally has been decided by analyzing the circuit at the worst process corners. Increasing

local random variability means that our current method of guard banding can give very pessimistic values for the guard band. Since it does not consider the averaging effect of these variations, it causes the circuit to be over-designed, which adversely affects its energy-efficiency. Process variations hurt in many other ways as well. The exponential dependence of leakage energy on $V_{\rm th}$ causes the overall average leakage energy to increase significantly with $V_{\rm th}$ variation. Thus in designing for the highest performance point without respecting $V_{\rm th}$ variability, the leakage for most fabricated chips will be unacceptable, wasting a lot of energy. To minimize this loss of energy-efficiency during fabrication, design optimization must account for process variations.

Robust energy-efficient design can be done at many levels of system design hierarchy – problem formulation, architecture, circuits and devices. This thesis focuses on robust energy-efficient circuit design. The design variables are circuit topology, logic style, transistor sizes, and supply and threshold voltages. The design metrics are the specified performance and energy, while the boundary constraints include input signal arrival times, output loads and signal transition times. We formulate the digital circuit design optimization problem as a **G**eometric **P**rogram (Appendix A). A GP is a special type of convex optimization problem which can be efficiently solved using interior point methods [10]. To facilitate design entry and analysis, we created SCOT – the Stanford Circuit Optimization Tool, which was useful for creating optimized designs using Geometric Programming. We analytically model the energy and delay of digital circuit blocks as GP compatible mathematical functions of the design variables. Using this tool, we have investigated the energy-efficiency of different adder topologies and extracted the overall energy-delay tradeoff curve of 32bit adders. Such energy-delay tradeoff curves can be used at upper levels in the design hierarchy to create energy-efficient architectures.

To make the design robust against random process variations, models for saturation

and leakage current variation are incorporated in the design optimizer itself. While doing this exactly can be difficult, we have developed efficient heuristics to guide the optimizer in choosing appropriate values of the design variables, with a modest overhead in design time. Depending on the topology, the resulting designs can be significantly more robust to the process variations than the nominal designs.

Chapter 2 describes the aspects of CMOS technology scaling that cause the two big issues we face today – power dissipation and process variability. Next, Chapter 3 looks at the power dissipation problem without considering variations. It describes our formulation of the energy-efficient digital circuit design problem using analytical energy and delay models and uses these models to do a case-study using a 32-bit adder to show how the optimization framework works. Although circuits designed this way are energy-efficient, they are not optimal in the face of process variations. Chapter 4 starts by describing ways of analyzing circuit timing with uncertain gate delays. We show the negative effect of of process variations on otherwise optimally designed digital circuits. With this motivation we discuss efficient techniques to include process variability in the design optimizer to generate statistically robust circuits.

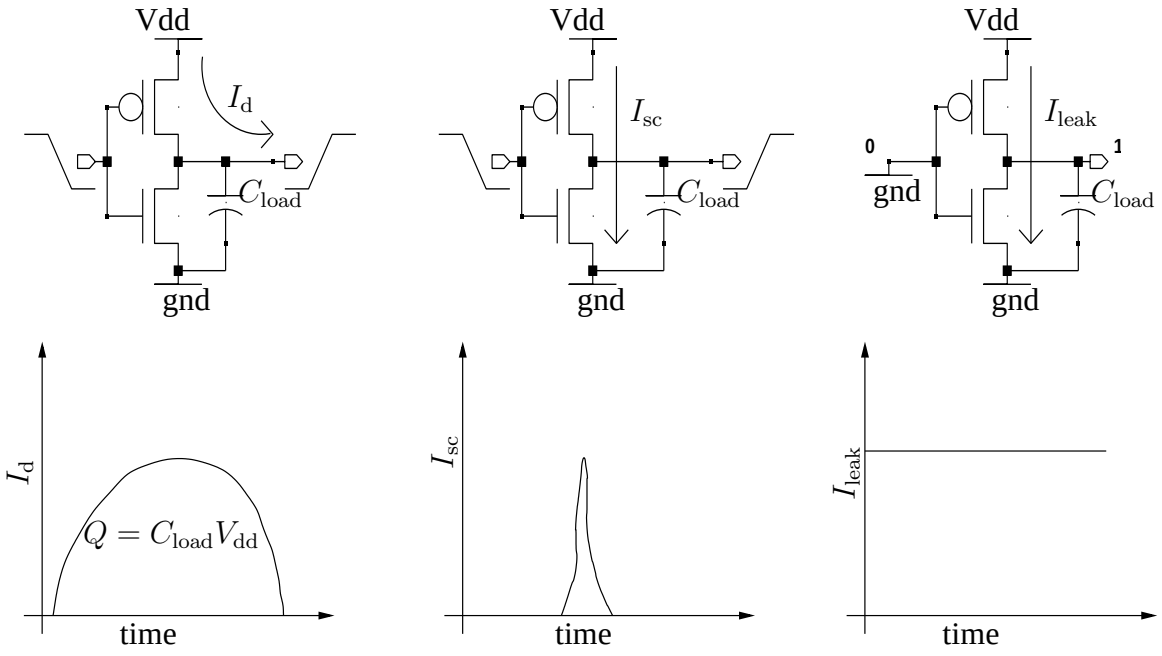
Chapter 2

Energy, delay and scaling

Power is now a critical issue for integrated circuit designers. Before describing methods of addressing this issue by creating energy efficient designs, this chapter will look at the basic mechanisms that cause energy dissipation and delay in CMOS circuits. We will then create simple, but accurate models that will allow us to estimate these quantities for arbitrary CMOS gates, and use these models to explain why power has increased during the past 30 years of scaling, and why the power problem has gotten much worse recently. Next we turn our attention to another factor that affects energy-efficiency – process variability. Given that devices have manufacturing tolerances, we need to add margins to ensure that the manufactured designs meet spec. This has traditionally been done using "corner files" information about the worst-case points in the manufacturing flow. With technology scaling, local fluctuations across a single die have become more critical. This chapter explains why these on-die variations are critical, and how they affect the design optimization problem.

2.1 Modeling CMOS energy and delay

There are three main sources of energy dissipation in CMOS circuits as shown in Figure 2.1.



(a) Charging current (b) Short circuit current (c) Leakage current

Figure 2.1: Forms of energy dissipation in CMOS circuits. The graphs shows the currents associated with the corresponding energy dissipation.

1. **Dynamic energy** consists of the energy dissipated in charging and discharging gate, diffusion and wire capacitances while switching a signal.

2. Signal transitions are never instantaneous. Whenever a gate switches, the pull up and pull down devices may be simultaneously on for a brief period until the input transition is complete, causing short circuit crow-bar current to flow between the power rails. The energy dissipated in this way during signal transition forms the **Crow-bar energy**.

3. A static source-drain leakage current flows through a transistor even when the transistor is off, i.e. when the gate source voltage $V_{\rm gs}$ is well below $V_{\rm th}$. Energy dissipated statically due to this sub-threshold conduction constitutes the **Leakage energy**.

The total energy dissipation is the sum of all these components over all gates and wires in the netlist.

Modeling circuit delay is slightly more difficult as it is determined by the slowest signal path during each clock cycle. As a digital signal propagates through the circuit, it turns some transistors on and others off. These transistors then drive their output nodes, charging and discharging different capacitances and changing the state of other transistors which drive the next output nodes. The delay of a path of logic is the sum of signal transition delays through the transistor stages along the path. As delay at every stage consists of charging or discharging a load capacitance C to or from voltage V using the driving transistor's drain current I, the delay per stage can be written as kCV/I, where k is a constant. The drive current I and the gate capacitance of a transistor are both directly proportional to the its width. Consequently, if the load capacitance C is a fixed multiple of the driving transistor's gate capacitance, the kCV/I delay becomes independent of transistor sizing. It depends only on the intrinsic driving properties of the devices in the technology. Therefore, kCV/I delay measured in this way is a good metric for defining the speed of a technology.

2.1.1 CMOS transistor characteristics

In order to correctly model the transistor current I, it is important to understand the CMOS transistor behavior and include in the device model all the important factors that affect delay and energy of CMOS circuits. Figure 2.2 shows a 3D view of a typical NMOS device structure that has been scaled over the years. For transistors with long channel length, the lateral source-drain electric field is small and the channel carrier velocity ν is proportional

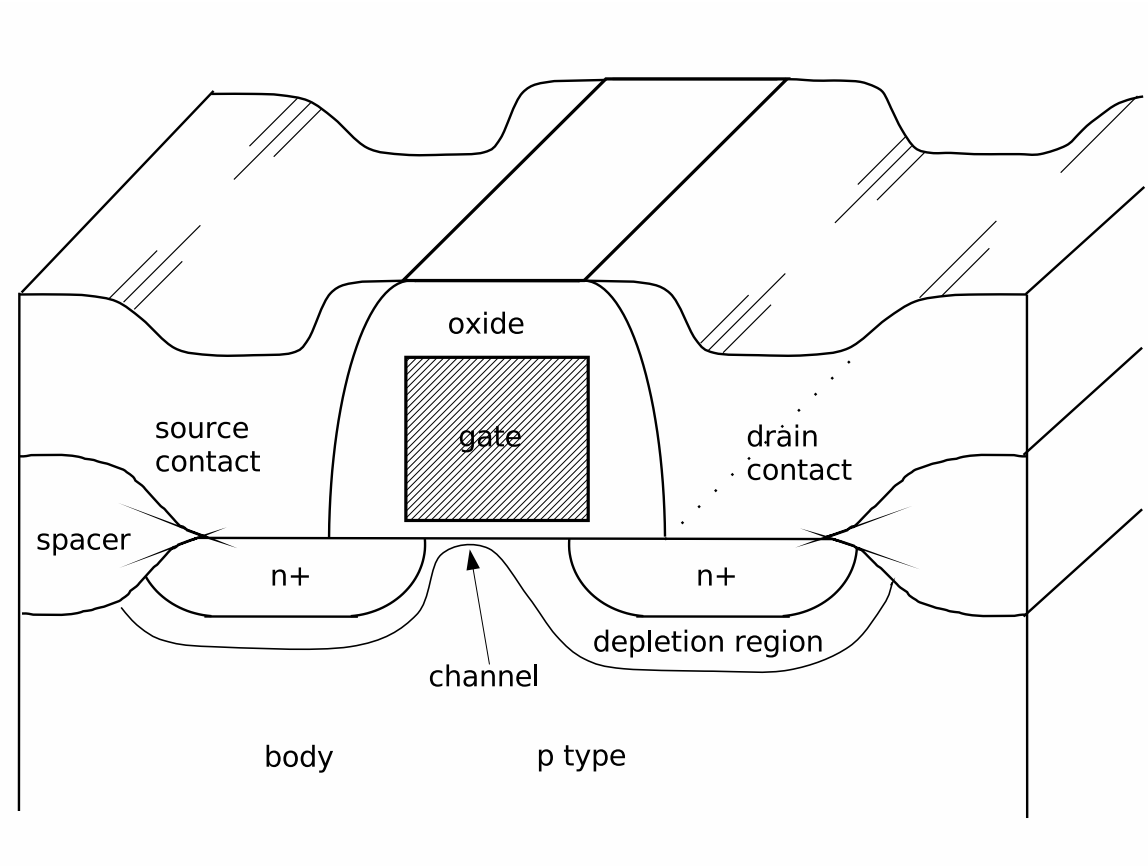


Figure 2.2: 3D view of the NMOS transistor

to the channel electric field E.

$$\nu = \mu E,$$

where μ is the mobility, which is largely constant over the operating range. In this scenario, the drain current in saturation $I_{\rm dsat}$ for a transistor of width W and length L can be modeled quite accurately by the simple quadratic formula [72]

$$I_{\rm dsat} = \frac{W}{2L}\mu C_{\rm ox} V_{\rm od}^2, \tag{2.1}$$

where $V_{\rm od}$ is the overdrive voltage given by $V_{\rm od} = (V_{\rm dd} - V_{\rm th})$[1] . Because $I_{\rm dsat}$ for a transistor depends only on the W/L ratio, the drive current for a cluster of transistors can be found easily by modeling each one of them as resistors [34] (where the resistance is proportional to L/W) and finding the effective resistance of the cluster using the series-parallel formula for resistors. For example, the current through a stack of two devices can be modeled as

$$I_{\rm dsat} \propto \frac{1}{\frac{L_1}{W_1} + \frac{L_2}{W_2}},$$

while for parallel devices we can write

$$I_{\rm dsat} \propto W_1/L_1 + W_2/L_2.$$

However, all modern transistors exhibit the so called "high field effects" which make this simple modeling very inaccurate. As L is reduced the lateral field increases and the relationship between the carrier velocity and electric field starts to saturate [88]. Carriers are velocity saturated to a $v_{\rm sat}$ of around $10^7 cm/s$ in silicon. With short channels, the drain depletion region forms a larger part of the channel length and voltage on the drain also can lead to changing the barrier to inversion, thus affecting the $V_{\rm th}$. This effect is known as Drain Induced Barrier Lowering or DIBL [88, 65]. Finally, the low-lateral-field mobility is affected by the vertical gate field [82]. We use the Meyer saturation current model [89] to effectively capture all these important effects.

In this model, $I_{\rm dsat}$ of a MOS transistor is given by

$$I_{\rm dsat} = \frac{W v_{\rm sat} C_{\rm ox} V_{\rm od}^2}{V_{\rm od} + E_{\rm c} L}, \tag{2.2}$$

where $v_{\rm sat}$ is the saturation velocity and $E_{\rm c}$ is the critical lateral electric field that sets the

[1] As we are talking about digital circuits with rail to rail signal transitions, we use $V_{\rm dd}$ as our $V_{\rm gs}$.

onset of velocity saturation.

$$E_{\mathrm{c}} = \frac{2v_{\mathrm{sat}}}{\mu_{\mathbf{eff}}}.$$

Here, the effective mobility $\mu_{\mathbf{eff}}$, is itself a function of V_{dd} and V_{th} as it depends on the vertical electric field [89]. The magnitude of $E_{\mathrm{c}}L$ relative to V_{od} determines the extent of velocity saturation and therefore the short channel behavior. When $E_{\mathrm{c}}L >> V_{\mathrm{od}}$, the lateral field is small, so velocity saturation is small and I_{dsat} obeys the square law relationship. The influence of a strong horizontal field in short channel transistors keeps the device in saturation well beyond the $(V_{\mathrm{gs}} - V_{\mathrm{th}})$ level of the long channel device. With velocity saturation, the drain saturation voltage is given by [89]

$$V_{\mathrm{dsat}} = \frac{(V_{\mathrm{gs}} - V_{\mathrm{th}})E_{\mathrm{c}}L}{(V_{\mathrm{gs}} - V_{\mathrm{th}}) + E_{\mathrm{c}}L}.$$

For devices below 130nm, this is much lower than $(V_{\mathrm{dd}} - V_{\mathrm{th}})$ as devices remain in velocity saturated mode for a substantial portion of their drain voltage swing.

2.1.2 Dynamic energy model

As inputs to a logic circuit change, signals at different nets transition to new values, charging and discharging the corresponding capacitances. If a capacitance C swings by a voltage V_{swing} through a supply of V_{dd}, the total energy spent by the supply is the product of the charge placed on the capacitor and the supply voltage, given as $CV_{\mathrm{swing}}V_{\mathrm{dd}}$. Usually, the circuit swings are equal to V_{dd}, therefore average total dynamic energy per operation can be calculated as a sum of the dynamic energy dissipated at each of the n nets in the circuit.

$$E_{\mathrm{dyn}} = V_{\mathrm{dd}}^2 \sum_{i=1}^{n} \alpha_i C_i \tag{2.3}$$

Here, C_i is the sum of the wire capacitance, gate capacitance of the fan-out gates and parasitic capacitance of the driving transistor stage on net i. The switching activity factor α_i measures the average transition frequency of net i. For a given transition frequency at the inputs, α_i is calculated by fast logical switch level simulation of the circuit.

Crow-bar current also only occurs during transitions and is generally lumped in with the dynamic power. While it depends on rate of change of the input, it usually is a small fraction of the dynamic energy and causes a small error if ignored in energy estimation. The accuracy of the energy model really lies in estimating the different device and wire capacitances accurately. The MOS gate and diffusion capacitances depend on the applied voltage. However, for a rail to rail switching transition, the average capacitance gives a good estimate for power calculation.

2.1.3 Leakage energy model

The current model explained in Section 2.1.1 predicts that there is no drain current when $V_{\rm od} \leq 0$. However, CMOS transistors do have leakage currents when they are "switched off". CMOS gates leak all the time as this leakage current flows through the switched off transistors. The leakage energy per cycle is the sum of the energy in all the gates in a circuit.

$$E_{\rm leak} = V_{\rm dd} T_{\rm cycle} \sum_{i=1}^{m} I_{{\rm gate},i} \tag{2.4}$$

Here, m is the number of gates, $I_{gate,i}$ is the weighted sum of the leakage currents of the gate i over its various logic states and $T_{\rm cycle}$ is the clock cycle time.

$$I_{{\rm gate},i} = \sum_{j} p_j I_j(V_{\rm dd}, V_{{\rm th},j}), \tag{2.5}$$

where p_j is the probability of state j and I_j is the leakage current in that state, which depends on the width and $V_{\rm th}$ of leaking transistors. Leakage current reduces significantly with stacking [39], so we consider only those logic states that have a single leaking transistor between the power rails. The leakage in a gate is distributed between its pull-up and pull-down stacks according to the proportion of time the other stack is active. In other words, it depends on the average time the output remains in state 0 or 1 respectively. This is calculated by measuring the duty factor simultaneously with the switching activity factors to obtain a good first order estimate of p_j. For a single leaking transistor of width W, I_j can be modeled as [63]:

$$I_j = I_0 W \exp(\frac{-(V_{{\rm th},j} - \gamma_{\rm D} V_{\rm dd})}{n_{\rm bf} V_{\rm T}}) \tag{2.6}$$

where I_0 is a constant, $\gamma_{\rm D}$ is the DIBL coefficient, $V_{\rm T}$ is the thermal voltage equal to $k_{\rm B}T/q$, $k_{\rm B}$ being the Boltzman constant and $n_{\rm bf}$ accounts for the body-factor describing the efficiency of gate to channel coupling [88]. The magnitude by which $V_{\rm th}$ must change to cause an order of magnitude change in I_j is called sub-threshold slope and can be calculated as $\ln(10) n_{\rm bf} V_{\rm T}$. For most CMOS devices, this value is around 80-100mV/decade.

2.1.4 Modeling gate delay

We have extended the Meyer velocity saturated current model [15, 89] for a transistor to obtain the delay of CMOS gates. To calculate this delay correctly, it is necessary to consider the entire path from the power rails to the charging load. Thus for modeling the delay of a transistor stage we define a stage (gate) as a Channel Connected Component (CCC) [21, 12] where inputs are only connected to gate terminals. A CCC is defined as the largest group

of transistors having their source/drain terminals connected through conducting channels[2]. Most common logic gates like inverter, NAND and NOR gates fall in this category. As we shall see below, tearing circuits into CCCs makes it possible to model the delay of CMOS gates analytically.

For sake of explanation, we will consider the process of a load capacitance being discharged by an NMOS pull-down stack. The analysis for PMOS pull-up is similar. The fall delay for discharging a load capacitance $C_{\rm load}$ to $V_{\rm dd}/2$ by applying a rising step input to the driver NMOS transistor is given by

$$\tau_{\rm step} = \frac{C_{\rm load} V_{\rm dd}}{2 I_{\rm d}},$$

where the discharging current $I_{\rm d}$ is given by Equation 2.2. However this model greatly underestimates the delay in real circuits because of two reasons. Firstly, the input is never a step but has a finite rise and fall time. This turns on the driver transistor slowly and contributes to the stage delay. Usually the stage delays are comparable, so the output does not cross $V_{\rm swing}/2$ until the input is well beyond the $V_{\rm dd}/2$ point. In such cases the input slope just adds a simple delay term [35]. With this assumption, we can estimate the fall delay $\tau_{\rm d}$ by

$$\tau_{\rm d} = \frac{C_{\rm load} V_{\rm dd}}{2 I_{\rm d}} + f(\tau_{\rm in}), \tag{2.7}$$

where $\tau_{\rm in}$ is the input transition time and $f(\tau_{\rm in})$ is the added delay. By approximating the shape of input transition and the driver current build up as linear, an input with finite transition time can be considered as a delayed step. This delay, $f(\tau_{\rm in})$ is given as [35]

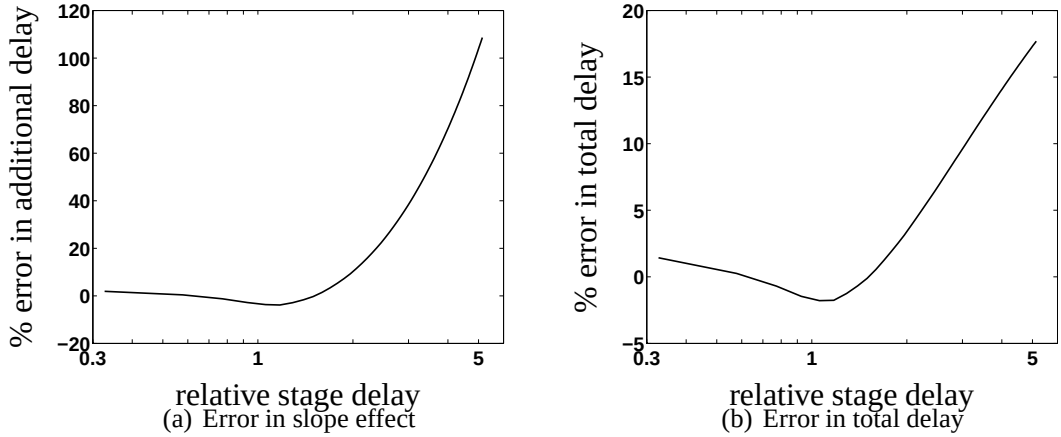
$$f(\tau_{\rm in}) = 0.5 \frac{V_{\rm th}}{V_{\rm dd}} \tau_{\rm in}.$$

[2]Some transistors are connected to power rails (supply and ground)

To assess the accuracy of this model, consider a two inverter delay path driven by a step input and driving a fixed load. The size of the second stage is fixed while the size of the first is changed to present inputs with different transition times to the second stage. Figure 2.3(a) shows the relative error in the estimating the extra delay added to the second due to its non-zero input transition time. The X axis represents the delay of the first stage relative to that of the second stage. If the input to the second stage transitions relatively quickly (i.e. the delay of the first stage is relatively small), the error is less, while for very slowly changing inputs, the error is large. However, if we measure the delay of the entire path, we can see that the error in estimating the total delay due to non-zero input transition time is significantly reduced.

While this error may still be important in timing "analysis", we care about how it affects optimal "design". Usually, optimal sizing ensures that stage delays and therefore the input and output transition times, are comparable. A light loading condition can occur during sizing if the transistors in a lightly loaded side path hit their minimum size constraint while the critical path is heavily loaded. However, in such cases the delay of the entire side path itself is already small compared to the critical path delay and so does not affect the sizing. In addition, we constrain the delay of every stage to avoid slowly changing signals for signal integrity reasons. This reduces the error in delay estimation. The second issue in accurate delay modeling is that during switching, the gate and drain voltages of the transistor are changing simultaneously and because of DIBL, $V_{\rm th}$ is also changing. Therefore the discharging current is actually changing throughout the output transition and $I_{\rm d}$ should represents an average discharging current. Hence for improved accuracy, we modified the expression of $I_{\rm dsat}$ in Equation 2.2 by adding two additional fitting parameters a and b to give

$$I_{\rm d} = a\frac{W v_{\rm sat} C_{\rm ox}(bV_{\rm dd} - V_{\rm th})^2}{(bV_{\rm dd} - V_{\rm th}) + E_{\rm c}L}.$$



(a) Error in slope effect (b) Error in total delay

Figure 2.3: Effect of loading conditions on the accuracy of our additive slope delay model

Parameter a is the averaging coefficient and b accounts for DIBL[3] As we are optimizing for $V_{\rm dd}$ and $V_{\rm th}$, we choose these parameters to best fit the entire range of desirable $V_{\rm dd}$ and $V_{\rm th}$.

The above equations can easily model the delay of an inverter, but complex gates are made of multiple series and parallel stacks of transistors. To model their delay we need to model the current of a stack of transistors. As was already mentioned, modern devices are velocity saturated and so cannot be combined as resistors. The issue is clear from Equation 2.2. Unlike the long channel transistor, increasing L of a velocity saturated device does not reduce the current proportionately. Therefore, a stack of two identical transistors behaves like a transistor with the same width but twice the length, as opposed to a transistor with same length with half the width as is true in quadratic transistors. This can be easily accounted for if all devices in the stack are of equal width, because then we can model that as a long transistor with the same width. However, in custom design, transistors in a stack can have different sizes depending on the delay at their gate inputs. For such cases, the

[3]The values of a and b used for our 90nm technology are respectively 0.9 and 1.12 for PMOS and 0.7 and 1.1 for NMOS.

current model needs to be generalized to estimate the drive current of a stack of transistors with unequal widths. The velocity saturated flow of carriers in a stack of transistors can be thought of as water flowing under pressure in connected pipes of different diameters. Here the flow of water is restricted mostly by the thinnest pipe, while the length of a thicker pipe has a relatively small effect on the flow rate. Using this idea, we model a stack of n transistors as an equivalent transistor with an effective width W_{eff} and effective length L_{eff} given as

$$\begin{aligned} W_{\text{eff}} &= \min(W_1, \ldots, W_n), \\ L_{\text{eff}} &= W_{\text{eff}} \sum_{i=1}^{n} L_i/W_i, \end{aligned}$$

where W_i and L_i are the width and length of the i^{th} series transistor. As devices in a digital circuit are usually of minimum length $L_{\min}$, we can rewrite L_{eff} as

$$L_{\text{eff}} = L_{\min} W_{\text{eff}} \sum_{i=1}^{n} 1/W_i.$$

Thus the equivalent width is set by the most velocity saturated device and the length is set by the averaged length of all transistors weighted by their widths. This allows us to size different transistors in the stack. If all widths are equal, the equations result in an equivalent transistor with the same width and length $nL_{\min}$ as expected.

The delay of a stack also consists of discharging[4] the intermediate capacitances. If the switching input is at the bottom of the chain (closer to power rails), then it has to discharge all the intermediate nodes. In this case we decompose the fall delay as a sum of fall delays where each intermediate capacitor is discharged by the chain below it, similar to the Elmore

[4]charging for PMOS

[23] delay calculation as shown in Figure 2.4. Note that the capacitors that are in the same

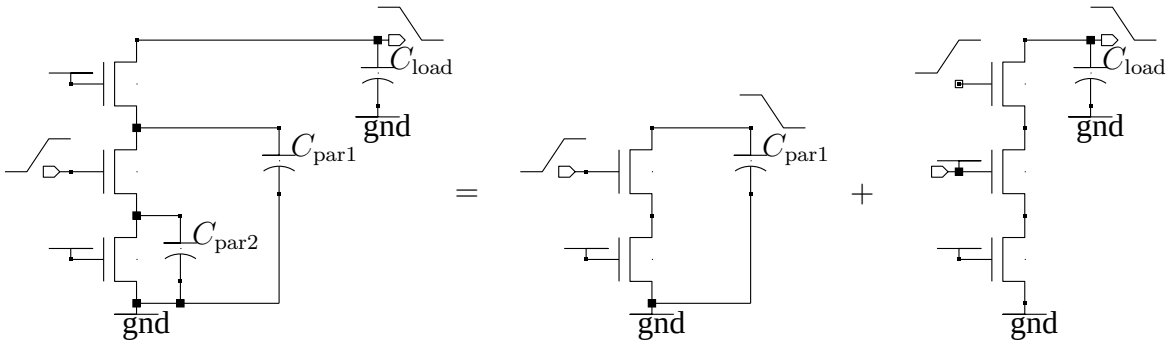


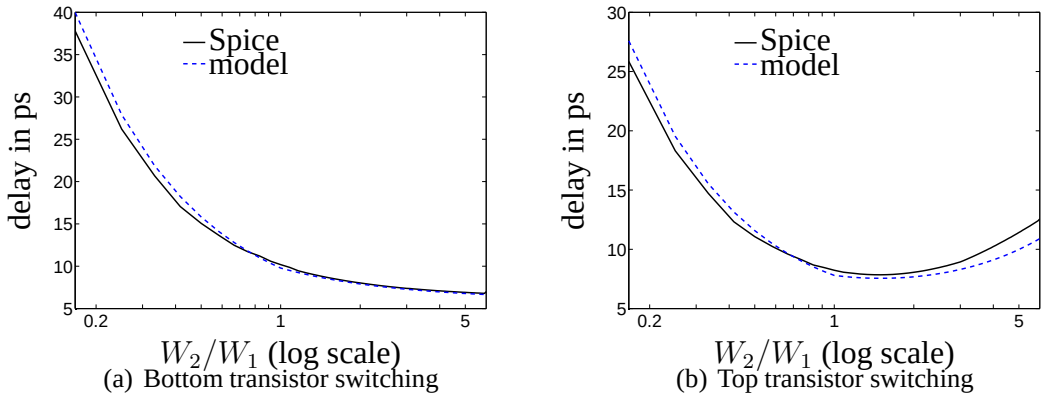
Figure 2.4: Delay of a stack of NMOS transistors activated at the intermediate input

state before and after the transition are not included in calculating the delay. Therefore we have no term for capacitor C_3 in the figure.

Figure 2.5 shows the validation of our model against spice simulations for step input to a stack of two NMOS devices as in a NAND gate. Let the width of the NMOS closer to the output be W_1 and the width of NMOS closer to ground be W_2. The models are more accurate near the point where $W_1 = W_2$. The error in the estimate shown in Figure 2.5(a) is due to our assumption that the intermediate node is at V_{dd} prior to discharge. In Figure 2.5(b), the error in modeling the delay from top transistor switching comes mainly due to ignoring charge build up in the capacitor at the intermediate node during the transition. If the intermediate capacitor is relatively large for the bottom transistor (width W_2) and comparable to the load capacitance, it can act as a virtual ground causing a smaller delay[5] and longer transition tail for the output [34].

An interesting effect of velocity saturation is that as V_{dd} is reduced, the relative reduction in the current of a stack of transistors is higher than that of a single transistor. This is expected because with reducing V_{dd}, the single transistor is less velocity saturated and therefore has a larger current relative to the stack of transistors, which suffered less velocity

[5]as calculated as the delay between the input reaching half-swing to output reaching half-swing



(a) Bottom transistor switching (b) Top transistor switching

Figure 2.5: Step delay model validation for a stack of 2 NMOS transistors

saturation to begin with. This is important for sizing, as it means that for the same drive capability, a stack of transistors (as in a NAND or NOR gate) has to be sized bigger relative to the single transistor stage (as in an inverter) as $V_{\rm dd}$ is reduced. This effect is neatly captured by the $(V_{\rm od} + E_{\rm c}L_{\mathbf{eff}})$ term of the drive current model.

The accuracy of the delay model remains well within 10% for chains of up to four transistors for reasonable $C_{\rm load}$ and intermediate parasitic capacitors and input signal transition times ($\tau_{\rm in}$) are comparable to the output signal transition times. Structures with up to four transistor stack include almost all the usual CMOS logic gates. In larger gates where there are many parallel stacks of transistors connected to the output, the model may underestimate the overall delay of a particular transistor stack as it does not consider the intermediate parasitic capacitance from other partially turned on stacks.

2.2 Technology Scaling

Over the years, the IC industry has successfully implemented Dennard scaling to improve performance. In 1974 Dennard proposed constant field scaling [19], where the electric fields in a MOS device are kept constant by scaling voltages with lithographic dimensions.

Scaling the feature size by the scaling factor α ($\alpha \approx 0.7$) per generation improved the delay per switching event by $1/\alpha$. The gate capacitance per micron remained constant as both the device length L and the oxide thickness $t_{\rm ox}$ scaled by α. The device current per micron also remained constant because the scaling of L and $t_{\rm ox}$ was offset by the scaling of $V_{\rm dd}$ and $V_{\rm th}$. This follows from Equation 2.1. This trend continued even in the velocity saturated device regime as can be observed by implementing scaling in Equation 2.2. Consequently, the CV/I figure improved by $1/\alpha$ in every generation. This improvement in switching delay enabled a $1/\alpha$ increase in the clock frequency $f_{\rm clk}$. Improved circuit techniques like deeper pipelining allowed $f_{\rm clk}$ to be increased further. With the gate capacitance per unit area $C_{\rm area}$ increasing by $1/\alpha$ due to scaling of $t_{\rm ox}$, the dynamic energy per switching event per unit area given as $C_{\rm area}V^2$ decreased by $1/\alpha$. Combined with slightly greater than $1/\alpha$ increase in $f_{\rm clk}$, the dynamic power per unit area $C_{\rm area}V^2 f_{\rm clk}$ thus increased only slightly in this period.

During this time the contribution of sub-threshold leakage to the overall power was negligible as $V_{\rm th}$ values were large. Thus Dennard scaling kept the power density under control while improving the processing speed. However, as described in Section 2.1.3 sub-threshold leakage current grows exponentially with $V_{\rm th}$, and non-scalability of thermal voltage $k_{\rm B}T/q$ with device dimensions means that for a given operating temperature, the sub-threshold slope would remain constant at around 80-100mV/decade [88]. This means that as $V_{\rm th}$ reduces during scaling, the leakage current will increase exponentially and at some point cannot be ignored, because it would contribute substantially to the overall chip power. This happened around $L = 130nm$ and consequently $V_{\rm th}$ scaling slowed down. Because the delay is related inversely to $(V_{\rm dd} - V_{\rm th})$, and increases significantly as $V_{\rm dd}$ approaches $V_{\rm th}$, $V_{\rm dd}$ scaling too slowed down to maintain adequate performance. Figure 2.6 shows the deviation from ideal scaling for some design parameters from industrial data

collected over the past two decades [66]. While the benefit of constant power density from

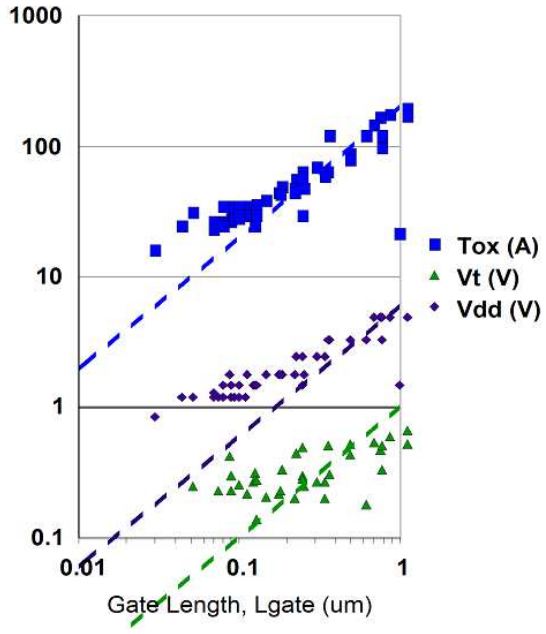


Figure 2.6: Scaling of $t_{\rm ox}$, $V_{\rm dd}$, $V_{\rm th}$ with gate length

Dennard scaling was no longer available, device dimensions continued to scale, increasing density and lowering cost, but also increasing power density per generation [88]. Power delivery and heat removal costs constrain the overall chip power in air cooled systems to around 100W. To be within this limit, frequency scaling has slowed down considerably this decade. Systems are running slower than the fastest allowed by technology and performance improvement is sought in other ways, like processing many instruction threads in parallel [75, 47, 71] rather than processing each one quickly. Foundries have also stepped up to provide technology that enables energy-aware design by offering multiple $V_{\rm th}$ devices [85]. In addition, designers are also trying to use multiple supply voltages [56, 77, 91, 14] on a single chip to achieve the timing in critical areas while saving leakage energy at other places.

To effectively use all these device and design options, the way we approach design needs to change. It is not enough to build the fastest circuit and then slow it down by changing a single design variable like the supply voltage. Optimization is required to correctly choose among these different options. The next chapter addresses this issue.

2.3 Process variability

The structures fabricated on silicon never exactly replicate the intended layout due to inevitable mechanical and lithographic variations. Variations arise from stepper misalignment, error in defining exact boundaries as the lithographic wavelength is higher than the etching dimensions, layout-pattern-dependent ion-implantation changes, fluctuations in the countably finite number of active dopant atoms in the scaled body of the MOSFET and so on. For a circuit to meet the specs amidst these variations, it is traditionally designed to work under different **corner** conditions of the fabrication technology. These represent the best and worst case devices for varying parameters like $V_{\rm th}$, $V_{\rm dd}$, temperature, mobility and so on. The device model corner files are provided by the foundry for every technology node. If a design meets the specification in simulation in the worst process corner, then it is guaranteed that almost all the fabricated designs will meet the timing.

With scaling, the relative magnitude of variations is increasing significantly [5]. If not accounted properly, the chip may be extremely under-designed making it hard to meet its design specifications and in some cases, to fail functionally as well. It is important to understand these variations to design robust circuits. From a design point of view, we classify these variations into three categories as shown in Figure 2.7, based on the amount of correlation among the devices on a chip .

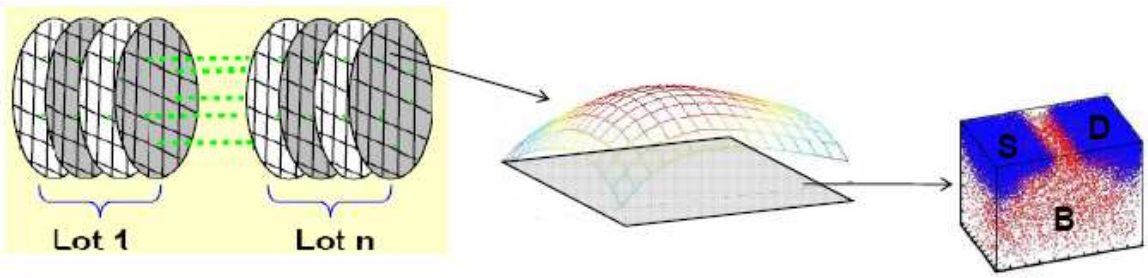


Figure 2.7: Types of process variations

1. **Global/Chip-to-chip**: Variations caused by factors external to the chip, wherein parameters for all the devices in the chip change in the same way fall in this category. In other words the variations in all devices on the chip are completely correlated. It includes all the lot-to-lot, wafer-to-wafer and on-die chip-to-chip variations. Because they affect all devices on the chip in the same way, they can be compensated by designing aggressively, leaving an appropriate guard band from the design spec, depending on the desired yield. If these are the only kind of variations, the appropriate value can be obtained by doing the traditional corner based simulations. Another solution for small perturbations in one of the process parameters is to compensate for it by changing some other design variable post fabrication. For example, V_{dd} of the chip can be set after fabrication to make the chip meet the desired specs.

2. **Correlated within chip:** Variations in which the varying parameter for different devices is correlated due to their proximity or similarity in layout fall in this category. The correlation distance can span from inter-device spacing to the chip dimensions. These variations cause parts of the chip to run slower or have a higher power density. One way to compensate for them is to assume they are part of the global variations. Better solutions include making a uniform layout to minimize layout dependent variations or using adaptive post fabrication tuning mechanisms for different blocks on

the chip to individually get them into spec.

3. **Local independent:** These variations have extremely short correlation distances. The parameter varies independently from device to device irrespective of their distance or layout. The main causes for these variations are the fluctuation in number of dopant atoms and randomness in photo etching causing length variations due to line edge roughness. As technology scales, the effect of these variations is growing significantly [64]. A small absolute change in the countably finite number of active dopant atoms can cause a large relative variation in V_{th}. As the feature size approaches the resolution of the photo-chemical process in resist, line edge roughness becomes more important. Clearly it is not feasible to compensate for such variations for every transistor using any of the methods used for the other types of variations. However, due to their short correlation distances, these variations tend to average out as the device area is increased. This is the key result used in designing circuits tolerant to these variations. A lot of research has been done to model such variations. Pelgrom's model [70], which says that the variance of a parameter is inversely proportional to the device area (LW) is a universally accepted model for these variations. To the first order this can be extended to express the variation in drive current and consequently, the transistor stage delay as function of the size of the driving transistor.

$$\sigma(I_{\mathrm{d}})^2 \propto \frac{1}{\sqrt{LW}} \tag{2.8}$$

More sophisticated models will be presented in Section 4.6.

Variations can also occur at run time due to changes in the environment, like fluctuations in V_{dd} and temperature. However, from a design point of view, they are like correlated within chip variations. So they can either be included in global variations, which means

designing in their worst case corner, or circuit level adaptive techniques can also be used to regulate them at run time [90, 16]. Variations can also occur due to aging of a device, like the Negative Bias Temperature Instability (NBTI) that affects PMOS devices. As before these variations can either be assumed global or dealt with using adaptive techniques. In this thesis, we shall focus on design optimization for circuits with local random variations using Pelgrom's equations to model them.

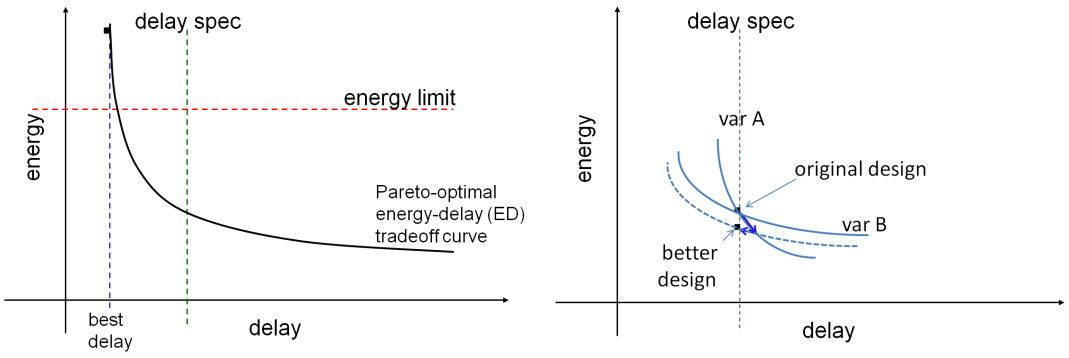
Variations in physical parameters result in variations in drive and leakage current, which lead to variations in circuit delay and energy. It is extremely unlikely that variations will improve both delay and energy of the chip as that means all parameters of all devices improved. Instead, variations cause many fabricated chips to not meet the energy-delay specification of the original design. We want to create an optimization method that optimizes the efficiency of circuits that are actually produced. The only way of doing this is to include the information about process variations in the design flow and generate a "robust" design, which would better tolerate the uncertainties in manufacturing. Incorporating variations exactly is usually a very hard problem [8], so Chapter 4 describes approximate but effective solution for generating robust circuits.

Chapter 3

Optimization for energy-efficiency

In this chapter we describe the energy-efficient circuit design problem without considering variations. After setting up the problem as a convex optimization problem we briefly talk about the Stanford Circuit Optimization Tool (SCOT) and describe experiments done with it to explore the relation between energy-efficiency and topology of 32bit adders.

Consider the "energy-delay" space of a typical digital circuit block as shown in Figure 3.1(a). For a given application specification, the fastest possible circuit may already exceed



(a) Pareto-optimal energy-delay tradeoff curve (b) Trading marginal costs to improve design

Figure 3.1: Energy-efficiency tradeoff space of a digital circuit block

the maximum energy limit, while the design may just have to meet the desired delay spec as

shown by the vertical dotted line on the right. If we gather all the possible slower designs built by tweaking the sizing, $V_{\rm dd}$, $V_{\rm th}$, topology and so on, we would see that there is a curve that bounds these designs on the lower side. Each point on this curve represents an energy-efficient design at that energy-delay specification, since none of the other designs are better in both. This curve is called the Pareto-optimal tradeoff curve and designs on this curve have a number of interesting properties. Most importantly, all their design variables must be in balance, i.e the marginal cost in energy for change in delay is same [10] for all the variables that the designer can adjust[1]. This must be true, else we can "sell" the expensive variable, buy back on the cheaper one, and get a design better in energy with the same delay or vice versa. Figure 3.1(b) shows this situation for a hypothetical two variable design. If design variable A has a higher marginal energy cost than variable B, we can make the design slower using A, reducing the energy and then speed it back to the original delay using B, increasing the energy by a lesser amount. Overall we would get a design with the same delay but lower energy, which is not possible if the original design was energy-efficient.

This chapter creates a mathematical framework to obtain this Pareto-optimal Energy-Delay (E-D) curve for a given circuit topology. These E-D curves can then be used at the higher level to choose the right design for the energy and delay constraints of the system.

3.1 Digital circuit sizing

To understand the energy-efficient circuit design problem, lets look at how digital circuit sizing is done today using circuit sizing tools. A typical digital circuit can be thought of as pools of combinational logic sandwiched between flip-flops. The clock edges driving

[1]This condition may not hold if the design variable has reached the end of its allowable range.

the flip-flops set the timing constraints on the combinational blocks. As every block has to meet the cycle time, the clock period is set by the longest path in the slowest block. The goal of a circuit sizer is to set the sizes of devices in these combinational blocks so that they meet the cycle time, while obeying area and energy constraints. Simultaneously, they have to meet other design constraints like maximum and minimum device size, input and output loading, signal rise/fall time constraints and so on. Most circuit sizers [17] model the gate delay in a static or data independent fashion. In this, the output transition time of a gate is set to be the worst case transition time for all input combinations. For example, the delay

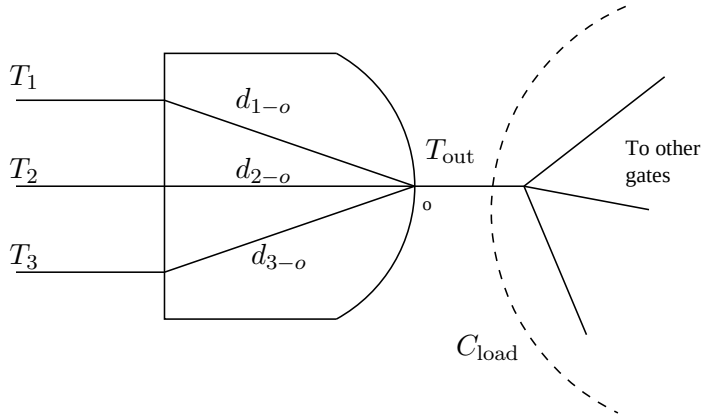


Figure 3.2: Gate delay constraints for the circuit sizing problem

of a typical three input gate shown in Figure 3.2 is given by

$$T_{\text{out}} = \max_{i=1,2,3}(T_i + d_{i-o}). \tag{3.1}$$

where T_i is the signal arrival time of input i and d_{i-o}, the typical gate delay from input i to the output o, is a function of the load capacitance C_{load}, transistor sizes W, channel length

L, supply voltage $V_{\rm dd}$, threshold voltage $V_{\rm th}$, oxide thickness $t_{\rm ox}$, and mobility μ:

$$d_{i-o} = f(C_{\rm load}, W, L, V_{\rm dd}, V_{\rm th}, t_{\rm ox}, \ldots). \tag{3.2}$$

Different circuit sizing programs use different methods to estimate d_{i-o}, including circuit simulation, analytical formulations or table look up. Irrespective of how the gate delays are estimated, the resulting sizing problems looks very similar as they are primarily governed by Equation 3.1.

These gates are then connected in a Directed Acyclic Graph (DAG) to form a combinational logic netlist as shown in Figure 3.3. The Primary Inputs (PIs) are typically assumed

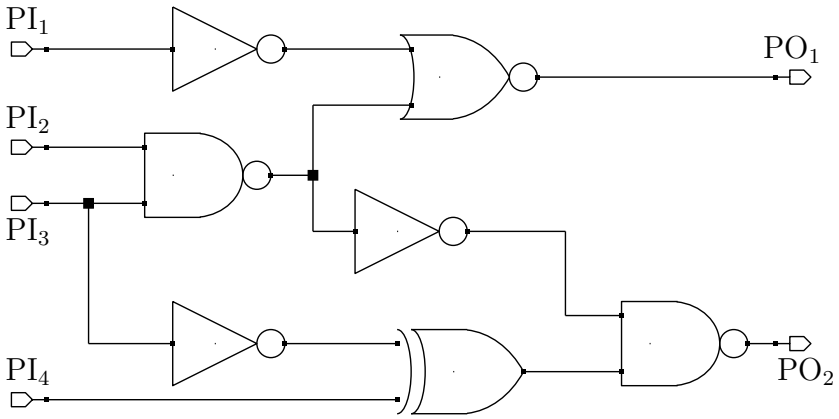


Figure 3.3: An example circuit netlist with boundary signals

to arrive at time $T_{\rm in} = 0$[2], and the delay of the circuit $T_{\rm d}$ is given as the maximum of signal arrive times $T_{\rm out}$ at any of the Primary Outputs (POs) that go to the flip-flop inputs. The static delay formulation is convenient because it allows us to obtain the circuit delay in terms of simple sum and $\max$ operations. The goal of the sizer is to set the widths of the transistors in the gates (or sizes of the standard cells) to optimize the delay, power and area of the circuit. In some cases one wants to minimize the delay, while meeting the power and

[2]or some known time as specified by the previous block

area constraints. At other times, one wants to meet the timing constraint, while minimizing the area. Without loss of generality, in this work, we chose the optimization problem as minimizing the delay under energy (or area) constraints. Each of the gate delays depends on the sizes of the driving gate itself, and the sizes of the gates it is driving. We can express d_{i-o}s as a function of the the widths of the transistors in the circuit (or the sizes of standard cells) and other factors like wire capacitances, $V_{\rm dd}$, $V_{\rm th}$, as explained in Section 2.1.4.

$$d_{i-o} = \mu(W, V_{\rm dd}, V_{\rm th}, C_{\rm load}, \ldots)$$

In the deterministic sizing problem, the expression for a particular d_{i-o} is assumed to return a number that represents the delay of that gate. We called the function for the d_{i-o}, $\mu()$, signifying the average delay, since if there was random variation in the gate delay one would use some form of average value for sizing. Using this gate delay model, one possible circuit sizing problem can easily be stated as - minimize the cycle time $\mathbf{T}_{\rm d}$, while keeping widths, slew rates, area within their specified limits. In the following sections we shall describe the mathematical framework for solving such optimization problems.

3.2 Optimization framework

The canonical representation of an optimization problem is as follows:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 1, \quad i = 1, \ldots, m, \\ & g_i(x) = 1, \quad i = 1, \ldots, p, \end{array} \tag{3.3}$$

where, $f_0(x)$ is the objective and $f_i(x)$ and $g_i(x)$ represent the inequality and equality constraints. The problem is convex if the objective is convex and constraints represent a convex set. For example, if $f_0(x)$, $f_i(x)$ and $g_i(x)$ are linear in x, the resulting problem is a linear program. Convex problems have the nice property that any local minimum is also the global minimum. Besides, many specific forms of convex problems have an efficient solution algorithm, thereby allowing one to solve large problems.

3.2.1 Digital circuit design problem as a Geometric Program

If the objective and constraint functions in Eq. 3.3 are posynomials, the resulting optimization problem becomes a Geometric Program (GP). The detailed description of a posynomial and other aspects of a GP are explained in Appendix A. Geometric Programs are not convex in their original form but can be converted into a convex optimization problem by change of variables and constraints using logarithmic transformation.

The models of performance metrics of digital and analog circuits are very amenable to posynomial modeling [33, 1, 11]. Section 3.3 explains how we model gate delay and energy as posynomials, creating a GP from the circuit optimization problem. In addition, every gate in a digital circuit connects to a relatively small number of other gates. Hence, barring global constraints like area and energy, most of the delay constraints are local and involve only a small number of design variables. Therefore the matrices involved in the optimization are sparse. Exploitation of sparsity leads to further improvement in solution efficiency.

For circuit sizing purposes, $f_0(x)$ in Eq. 3.3, becomes the overall circuit delay T_{d} while the constraints describe the boundary conditions and design constraints [7, 6]. Some examples are given below.

1. Area/Energy: While exact formulation of area is difficult, as it is dependent on placement and wire count, it is usually formulated as the sum of all the transistor widths or standard cell footprint areas. In case of standard cells, the areas are curve fitted as a posynomial function of their sizes. In any case, area is a weighted sum of the sizing variables, with positive weights. Energy is also the sum of energies dissipated in the different switching capacitors and leaking idle gates. It is a function of device widths, $V_{\rm dd}$ and $V_{\rm th}$.

2. Input capacitance and Output load constraints: Output load is the capacitance the circuit has to drive, while input capacitance is the maximum allowed capacitance on any of the primary inputs as seen by the previous block in the signal path.

3. Device width, $V_{\rm dd}$ and $V_{\rm th}$ bounds: Fabrication limits place bounds on device width. $V_{\rm dd}$ is bounded by reliability on the upper end and subthreshold region on the lower end, while $V_{\rm th}$ bounds are typically given by the device manufacturers to bound it between high leakage on the lower side to subthreshold operation on the upper end.

4. Slope constraints: For signal integrity reasons the rise and fall time of digital signals are constrained on every net to be within a given limit throughout the netlist.

5. Transistor ratios inside a gate: Pre-charge and keeper transistors in dynamic logic are sized in ratio to the NMOS pull down stack, so that they can track the pull down strength.

To formulate this optimization problem and facilitate design entry and analysis, we built a tool at Stanford, called the Stanford Circuit Optimization Tool. This tool has leveraged the rich prior work in circuit sizing. Some of the key results that we used are described next.

3.2.2 Prior work in performance-energy optimization

Circuit sizing is an old problem and designers have developed many simple rules to manually size custom circuits. If the circuit is symmetric for all inputs, like a memory decoder, it can be reduced to a single critical path, and can be sized for highest speed using simple equations [87]. However, for most real circuits with many re-converging paths, these manual techniques are only a rough guide, because with global area/energy constraints, optimal sizing is difficult – it requires optimizing many variables simultaneously. To solve these complex circuits more effectively, many circuit sizing tools have been developed.

As early as 1985, Fishburn and Dunlop showed using simple delay equations that the path delay is a posynomial and the circuit sizing problem for minimum delay is a GP, which meant that the problem had a single global minimum. They developed a sizing tool called TILOS (TImed LOgic Synthesizer) [24], which used path enumeration in every iteration to find the gate with the largest sensitivity to the overall delay and change its size to improve timing. This took many iterations and the convergence to the optimal design was extremely slow. At that time there were no good algorithms to solve large scale GP.

More recently, in 1999, IBM developed a circuit sizing tool called Einstuner [17]. Einstuner used static timing formulation to avoid path enumeration and used fast simulation to calculate the sensitivity of the overall delay to gate sizes. This enabled more accurate modeling of the sensitivities using Spice simulations, at the cost of losing the guaranteed convexity of the problem and adding computation time. Einstuner used a generic non-linear solver.

Since TILOS, many efficient interior point algorithms were developed to solve convex problems like GPs [10]. These not only made sizing quicker, but now energy constraints could be included to do energy constrained sizing. Researchers have combined the static timing formulation of the sizing problem and the simple delay equations [87] to solve the

energy-delay optimization problem and obtain the E-D tradeoff of circuits.

But in this era of velocity saturated transistors, the simple delay equations are not accurate enough, especially over a range of $V_{\rm dd}$ and $V_{\rm th}$ and in custom design scenario where it may be desirable to size all devices in a gate individually for maximum energy-efficiency. Some work has been done to address this issue, including accurately modeling the gate delay by fitting a posynomial of the design variables to simulation results [42]. The choice of multiple $V_{\rm th}$ devices and possibility of using multiple supply voltages led many researchers to explore energy reduction techniques through simultaneous $V_{\rm dd}$, $V_{\rm th}$ assignment [36, 2, 84, 44, 83]. One commonly followed idea is to size low $V_{\rm th}$ gates to meet the cycle time and then convert the non-critical ones to high $V_{\rm th}$ [2]. The $V_{\rm dd}$ assignment algorithms allowed multiple $V_{\rm dd}$s on the chip and ensured that only the high $V_{\rm dd}$ gates can drive the low $V_{\rm dd}$ ones to avoid leakage in PMOS [99, 92]. Use of level converters was also explored [37, 48].

Researchers in our research group and at University of California at Berkeley had used the above ideas in an ad hoc manner to explore the E-D tradeoff of a specific circuit, by generating a GP and solving it in matlab [54]. This motivated us to build a generic tool for getting E-D tradeoffs of other digital circuits. For this we needed a clean and easy design entry to explore different topologies, automatic generation of gate delay and energy models, efficient automatic solution of the resulting GP and provision for back annotation in spice or schematics for seeing the results. As presented in the previous chapter, we decided to use analytical modeling based on physics for calculating the gate delays d_{i-o}. An analytical model provides better circuit intuition as it clearly shows the sensitivity of various parameters to the gate delay. This also helps in changing the model as the technology changes. Variation in delay due to variations in different device parameters can also be obtained easily by taking derivatives. To avoid the path enumeration problem, we use block

based static formulation for delay propagation. The tool also allowed us to extend the circuit analysis by adding Monte Carlo simulations for statistical timing analysis in presence of local variations as we will explain in the next chapter. Our algorithms for robust design were also implemented easily with our own tool.

3.3 GP compatible models

Except for dynamic energy, the analytical models presented in the previous chapter are not in posynomial form in general. Posynomials can be thought of as modified polynomials where exponents are allowed to be real but the coefficients are restricted to be positive real numbers. Our modeling needs to be accurate in a certain range of sizing, $V_{\rm dd}$, $V_{\rm th}$, signal rise and fall times etc. With some rearrangement and posynomial transformation in some parts of the original equations, we can make them GP compatible in that range.

3.3.1 Modeling gate delay

If we expand the step delay equation for a stack of transistors using Eq. 2.2, we obtain

$$\tau_{\rm step} = \frac{C_{\rm load} V_{\rm dd}}{2 W_{\rm eff} v_{\rm sat} C_{\rm ox}} (\frac{1}{V_{\rm od}} + \frac{E_{\rm c} L_{\rm eff}}{V_{\rm od}^2}). \tag{3.4}$$

While formulating the gate delay constraints, the added delay due to $\tau_{\rm in}$ is absorbed in the delay of the fan-in gate. The expressions $1/V_{\rm od}$ and $1/V_{\rm od}^2$ can be expanded to posynomials with arbitrary accuracy using

$$\frac{1}{1-x} = 1 + x^2 + x^3 + x^4 + \ldots, x < 1,$$

or by fitting a posynomial over the range of $V_{\rm dd}$ and $V_{\rm th}$ used in design. To capture the true nature of mobility change with $V_{\rm th}$ and $V_{\rm dd}$, we expand $\mu_{\mathbf{eff}}$ so that $E_{\rm c}$ becomes a function of the design variables. The empirically derived FET carrier mobility equation (for electrons) is [15]

$$\mu_{\mathbf{eff}} = \frac{540}{1 + (\frac{V_{\rm dd}+V_{\rm th}}{5.4t_{\rm ox}})^{1.85}}.$$

Even with this transformation the analytical model remains a posynomial, although it becomes complex. If the range of $V_{\rm dd}$ and $V_{\rm th}$ is small, this complexity can be removed by assuming $\mu_{\mathbf{eff}}$ and therefore $E_{\rm c}$ to be some fixed average value in the range of interest. Having included the effect of $V_{\rm dd}$ and $V_{\rm th}$, the d_{i-o} can be obtained by considering all chains activated by the input i that can contribute to drive the output o and taking the maximum of these delays, for static problem formulation. The delays for a CCC thus obtained are a generalized posynomials [10] of its transistor widths and other design variables.

While the model allows each transistor to be sized individually, in standard cell based designs, all the transistors in the cell are sized together using one sizing variable. If the width of each transistor in the cell can be obtained as a function of the cell size, then the delay equations can be written for standard cells too. However, it is far more convenient to fit a posynomial [10] on the tables in the standard cell library to express cell delay, area, input capacitances, etc., as a function of the cell size. This simplifies the problem of choosing the correct standard cell sizes after optimization from a few discrete values available in the standard cell library.

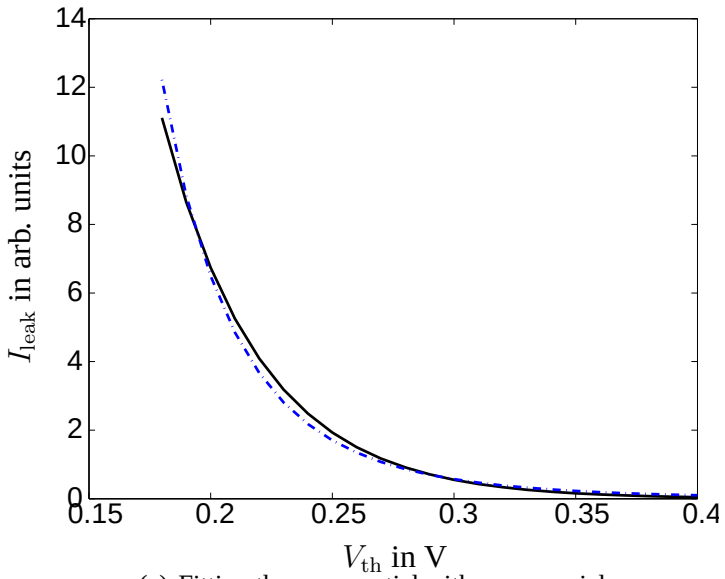
3.3.2 Modeling leakage energy

In order to include the leakage energy in the energy constraint, it should be expressed as a posynomial. A GP becomes convex by logarithmic transformation of its variables and

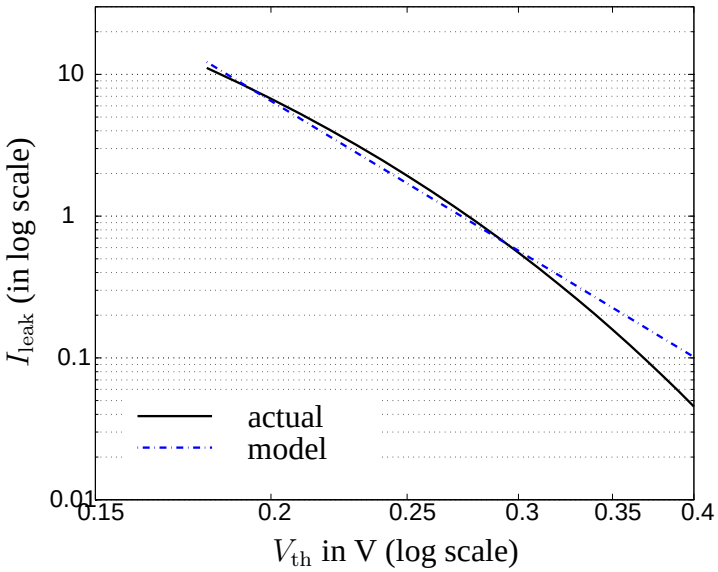
constraints [10]. Therefore, for any function to be compatible with GP, i.e. to be modeled accurately as a posynomial, it has to be convex in the log domain – also called log-log convex [9]. While the function $\exp((-\mu(V_{\rm th})/nV_{\rm T}))$ in the leakage energy equation (Eq. 4.17) is convex, it cannot be accurately modeled using posynomials over a large range of $I_{\rm leak}$ as $\log(I_{\rm leak})$ is not a convex function of $\log(V_{\rm th})$. However, note that leakage current varies over three orders of magnitude for the desired range of $V_{\rm th}$. Once the leakage of a gate falls below a certain value, its contribution to the overall energy is very small and its precise estimate is not crucial. Hence we only need to accurately model the leakage for the low $V_{\rm th}$ range, where the leakage current can contribute significantly to the overall energy. Figure 3.4 shows how fitting a monomial to leakage current causes large relative errors in the low leakage region while showing good accuracy in the high leakage portion. The log-log plot clearly shows that the best convex fit to a concave function is to model it with a straight line. A straight line in log-log domain is a monomial in the normal co-ordinates. We use this monomial to model the transistor leakage current.

3.3.3 Stanford Circuit Optimization Tool (SCOT)

The circuit design problem is formulated using analytical delay, area and energy models and solved using a commercial solver called MOSEK [61], which is a suite of routines for solving convex optimization problems. Wrapper programs are used around this package to enable the design entry as a Spice netlist, assimilation of the device model data into equations, conversion of netlist schematic into the canonical optimization problem format and post analysis of the results, including back-annotation for validation in Spice or visualization in schematic editor. The models and equations are in the form of generalized posynomials (Appendix A). The details of the tool are presented in Appendix B. Many non-convex



(a) Fitting the exponential with a monomial



(b) Modeling concave function with straight line in the log domain

Figure 3.4: Modeling the negative exponential of I_{leak} with a monomial

and hard-to-model circuit scenarios that occur during design (for instance while using special logic families or non-conventional circuit topologies) are described in the appendix. It also explains how they are handled in SCOT. The next section describes how we used

SCOT on 32-bit adders to derive the overall E-D tradeoff of CMOS 32-bit addition.

3.4 Circuit case study: 32-bit adder

Adder structures are ubiquitous in modern digital systems and are often present in critical timing blocks. Consequently, designers have studied adder topologies extensively and have developed many techniques to improve addition algorithms [96]. While initial papers [62, 46, 55, 30] focused mainly on performance, recent research [100, 67, 54] has also focused on energy, now a critical issue in digital design. Using SCOT, we extended this work by systematically exploring 32-bit custom designed adder topologies for their energy-efficiency [68]. We developed a relation between topological choices and energy-efficiency by comparing the Pareto-optimal E-D tradeoff curves of selected adder topologies in different logic styles, based on sizing, supply voltage and threshold voltage optimization.

Many architectures in different logic styles have been proposed, so adders are a rich area to explore. In addition, their layout is well known, so we can estimate the wire capacitance in different topologies and logic styles with good confidence. To run this experiment, we made schematics of several 32-bit adders with proper wire load estimates, and generated their E-D tradeoff curves. The lower bound of all curves gives the overall E-D tradeoff curve for 32-bit addition and indicates which topology and logic style is most energy-efficient for each region of the E-D space. These curves are discussed in Section 3.4.3.

In order to distinguish different adder topologies for energy-efficiency, before discussing these results, the next section describes the topological parameters that affect energy and delay. Next we describe the design constraints that we include in the optimization. The results and insights obtained using this tool are described in Section 3.4.3, where we compare the Pareto-optimal E-D tradeoff curves of various adders for a common set of

design constraints.

3.4.1 Adder topologies

We focus on circuits that add two N bit numbers to produce an $N + 1$ bit result. This operation basically consists of "propagating" the carry "generated" at any of the bit positions to all the higher positions. The final carry at each bit is then XORed with the bit sum to produce the sum at that bit position. Thus adder topologies can be separated into their Sum Generation Logic (SGL) that produces the bit sums and their Carry Propagation Logic (CPL) [94]. As CPL dominates the adder delay and energy, we use the following four parameters of the CPL (based on Harris's work [28]) to describe adder topologies.

1. Radix (R): In tree adders, we define R as the average number of bits combined at each logic stage[3] of the CPL. In linear carry-skip or carry-select adders, R refers to the average number of bits combined per stage to generate a block Propagate-Generate (PG) term.

2. Logic depth (L): L indicates the total number of stages in the CPL, and is at least $\log_R N$ for an N-bit adder. Note that the number of logic stages in the adder can be more than L.

3. Fanout (F): F represents the maximum logical branching seen by any stage in the CPL.

4. Wiring tracks (T): T measures the maximum number of wires running across the bit pitch between any successive levels of the CPL.

[3]To avoid confusion between a logic operation and the number of transistor stages needed to implement it, we consider every stage, including simple inversion, as a logic stage

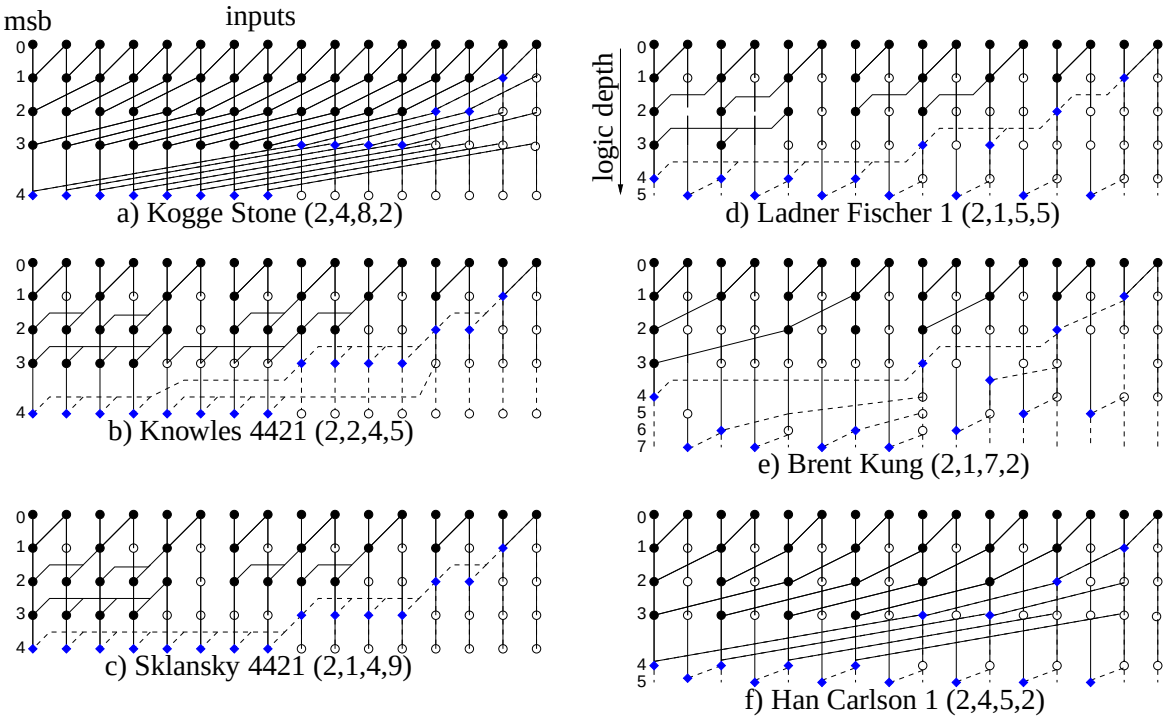


Figure 3.5: CPL dot diagram of selected adders with their (R, T, L, F). Solid lines and circles are PG signals and PG combine cells, dashed lines and diamonds are carry signals and carry generate cells, and empty circles represent wires or buffers.

For ease of visualization, CPL is usually represented using dot diagrams, where the dots mean logical operations on PG signals. The inputs are bit PG signals and the outputs are the final carries to the different bit locations. Figure 3.5 shows the CPL dot diagrams of selected 16-bit radix 2 adder topologies with different T, L and F numbers. For carry generation, R, L, F, and T are inter-dependent [28]. Brent-Kung, Sklansky [80] and linear ripple-carry adders have the least number of wire tracks ($T = 1$). Kogge-Stone and Sklansky adders are examples of minimum logic depth adders [46]. Kogge-Stone and linear ripple-carry adders have $F = 2$, while a 32-bit Sklansky adder has $F = 17$. For a given radix, Kogge-Stone (KS), Brent-Kung (BK) and Sklansky[4] maximize one of the three parameters T, L, and

[4]The Sklansky CPL design belongs to the Ladner-Fischer [49] family of adders. However Sklanky's adder [80] predates Ladner-Fischer adders and can be easily reduced to Ladner-Fischer design with removal

F respectively while minimizing other two. The Knowles, Ladner-Fischer (LF) and Han-Carlson (HC) topologies [46] tradeoff two of these three parameters keeping the third fixed [28]. Simple linear adders can also be described using R, T, L and F. For example, the 32-stage ripple-carry adder can be said to have $(R, T, L, F) = (1, 1, 32, 2)$. A carry skip and/or sum-select scheme in linear adders can be similarly described based on how they modify the topology.

For complete analysis, we consider the following frequently occurring design scenarios.

1. **External buffering:** Inverters can be added at all the inputs or all the outputs of the adder to best match the load it is driving, without changing its R, T, and L numbers, though with a possible inversion of the resulting sum. Hence for a given adder topology and design constraints, we optimize with all possible external buffering and choose the best of the tradeoff curves. This allows for a fair comparison between adders with different L.

2. **Internal buffering and restructuring:** Logic functions can be evaluated using a single complex high fanin (higher valency) gate or a series of smaller low fanin (lower valency) gates. The radix R is smaller in the latter case[5]. In our definition, R depends on the number of stages needed to do a particular logic operation on a given number of input bits. Thus, inverters appended to a complex gate in the middle of the critical path (as opposed to external buffering) reduces the overall radix. For example, a domino gate that consumes four bits and generates a 4-bit PG term has a radix of 2, because the dynamic gate and the inverter are two separate stages of computation.

of conditional-sum logic. Hence in this paper, we will use refer the Ladner-Fischer adder with highest F as Sklansky design.

[5]In adder literature, radix typically refers to the logic operation and "valency" refers to its implementation. However, from circuits point of view a higher radix CPL implemented with lower valency gates behaves similar in energy-delay to a lower radix CPL. Hence, for extracting the E-D tradeoff, we consider valency and radix as equivalent and use radix in this research.

A radix 4 domino design would consist of alternating 4 input dynamic and 4 input static stages. Even for the same radix, however, one can have two different designs. For example, a 4 input gate followed by an inverter and a two level tree of 2 input gates are both radix 2, but use different implementations. In these cases, we pick the best of the two designs for comparison.

3. **Sum selection:** The SGL usually consists simply of XOR logic. However, in sum select adders, the SGL generates a pair of speculative sums, to be correctly chosen when the respective carry arrives. Sum selection is a very common technique used for high performance adders today [62, 100]. Because only every k^{th} carry needs to be generated and fanned out to *k* muxes, a *k*-bit sum selection scheme re-distributes the logical fanout of the CPL by increasing the fanout of the final carry to *k*. This can potentially change the F of the CPL, independent of its R,T and L, creating multiple adders with the same R,T and L numbers. Because the CPL is a more critical part of the adder, we first find the most energy-efficient CPL structure and then explore the related sum selection techniques. The CPL and SGL, being in parallel, are almost independent for optimization purposes. This sequential procedure should therefore give optimal results.

4. **Ling adders [51]:** These use a reformulation of the Propagate-Generate (PG) equations of tree adders. Because Ling's equations are also associative and fall into a tree structure, they can be described using R, T, L and F. In this work, we look at the best PG adder structures and then compare them with similarly constructed Ling adders. Sum selection schemes are separately explored in both cases.

3.4.2 Design Constraints

In our analysis we focus on three principal design metrics: energy, delay, and output load (Cout). We optimize adders for sizing, supply (V_{dd}) and threshold voltages (V_{thN} and V_{thP}). For a given Cout, we generate E-D tradeoff by optimizing the delay at different total energy constraints.

Most of the critical wires in an adder run across bits and thus their lengths are set by the bit pitch. Wires along the bit pitch generally have lengths set by the number of transistors in that bit pitch, hence we capture these using fixed capacitances. This assumption is invalid in cases of high energy, when gate sizes become very large. However, in this region, transistor capacitances dominate the wire capacitances anyways, so the resulting sizing errors should be small.

We constrained the input capacitance (Cin) at any input to be less than 25fF, or roughly 15μm of transistor width. This is a reasonable load within the driving capability of library flip-flops in a 90nm CMOS technology. Except at the high energy points, this constraint does not come into play.

For small loads, if the Cin constraint is active, the adder has already entered the region of diminishing performance returns for added energy. For large loads, external buffering at the output of the adder is always more efficient than increasing the sizes of the gates in the adder. Adding inverters at the output generally reduces the required input capacitance to fall within the specified Cin constraint. To check the effect of a Cin constraint we optimize a few adders without the input constraint and show that it makes little difference. Reasonable signal slopes are maintained at every net, by limiting the delay of every logic stage. The minimum transistor width is constrained to $0.25\mu m$. V_{dd} ranges from 0.5V to 1.3V, while V_{th}s range continuously from about 0.2V to 0.4V. Both are common for all gates in the netlist. All dynamic gates have footers, keepers and intermediate precharge transistors,

which are used for stacks of three or more transistors. The sizes of all these peripheral devices are ratioed to their respective NMOS pull-down transistors in order to track their sizes under optimization. The final XOR gate for sum generation (or sum select mux) is static in all cases except for dual-rail domino circuits.

The activity factor of nets can differ by orders of magnitude especially in domino adders. This means the gates on low activity nets can potentially be sized larger and still not dissipate too much dynamic energy. The sizes however, do not blow up because they are still constrained by the Cin.

Because different topologies have different logic depth, fanout, and internal loading and hence can be optimal under different load conditions, we generate the pareto-optimal E-D tradeoff curves for different values of Cout.

3.4.3 Results and analysis

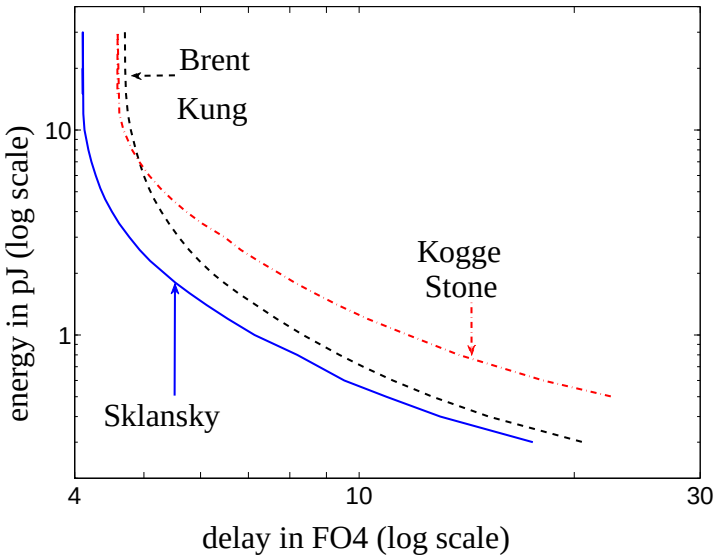
The delay of an N-bit adder primarily depends on how fast the carry reaches each bit position. Parallel prefix logic networks [46], which use tree structures to compute the carry, are very efficient for large N [87]. Hence, for a systematic traversal of adders, we start with the three corner radix 2 parallel prefix adders based on L, T and F (Section 3.4.1) built in static CMOS logic. We will show later in this section that 2 is the optimal radix.

We first consider the corner adders – Kogge-Stone, Brent-Kung and Sklansky – designed in 90nm static CMOS logic. We specify the delay in FO4[6], which is 31ps[7] for our technology. Figure 3.6 shows the pareto-optimal E-D tradeoffs of these adders for Cout = 25fF and Cout = 100fF. The adder designs include external buffering if necessary. Figure 3.6(b) also shows the E-D curve for a Sklansky adder with no input capacitance constraint.

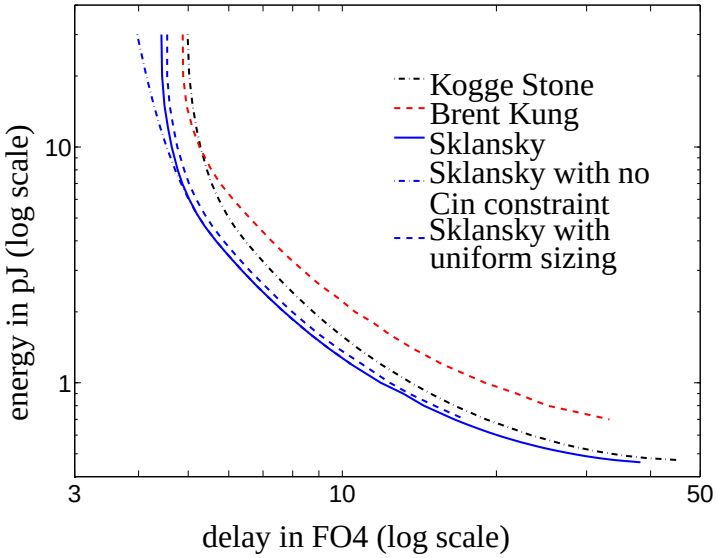
[6] An FO4 delay is the delay of single inverter driving 4 copies of itself from a step input. The FO4 delay is typically used to characterize the technology speed.

[7] While the FO4 delay changes with $V_{\rm dd}$ and $V_{\rm th}$, the unit used here is measured at nominal $V_{\rm dd}$ and $V_{\rm th}$.

We can see that this constraint is active only in high energy regions.



(a) Cout = 25fF



(b) Cout = 100fF, with a Sklansky adder E–D curve for uniform sizing and one for no Cin constraint.

Figure 3.6: E-D curves for the three radix 2 corner adders.

Figure 3.7 shows how the supply and threshold voltages change across the E-D curve

for a Sklanksy adder. When V_{dd} reaches its upper bound, the threshold voltages continue to decrease. At the lower bound of V_{dd}, the design becomes infeasible due to signal slope constraints, even though V_{th}s have not hit their upper bounds. If the input activity factor is increased (decreased), the supply and threshold voltages both decrease (increase), increasing (decreasing) the leakage power in relation to the increase (decrease) in active power.

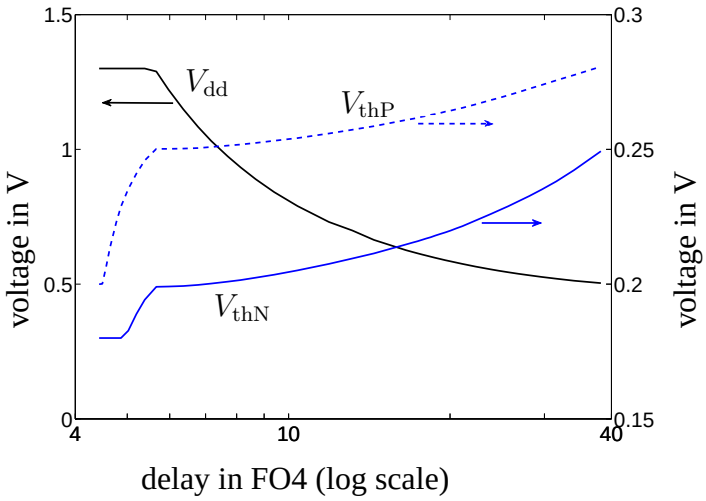


Figure 3.7: Change in V_{dd} and V_{th}s across the E–D space for a Sklansky adder.

The E-D curves show diminishing returns as we go towards either higher performance or lower energy. At higher energies, as device sizes continually increase, the effect of wires and Cout decreases. After a certain point, the gates would largely be driving their own parasitic capacitance and further improvement in delay would not be possible. With Cin constrained, after a certain energy, the design becomes identical to one based solely on logical effort [87], for which the marginal cost of energy for improvement in delay is infinite. Similarly, as energy is lowered, the supply and threshold voltage both change (see Figure 3.7) until the design enters a region where, analogous to the minimum delay solution, the marginal cost in delay for lowering the energy is very high.

The three radix 2 adders in Figure 3.6 each minimize two parameters out of L, T and F, at the cost of the third. Our experiments indicate that the Sklansky adder, which has the highest F, but smallest L and T, is the most energy efficient. Clearly, these three parameters do not trade equally with each other.

A large logical fanout at a particular stage does not necessarily imply that that stage will be slow. What matters to the delay is the *electrical* fanout (due to capacitive loading). In a Sklansky adder, at every n^{th} stage of the CPL tree, the carry drives $2^n + 1$ gates. After optimal sizing, we find that of these $2^n + 1$ PG combine gates at the $(n+1)^{\text{th}}$ stage, the one gate that drives $2^{n+1} + 1$ PG combine cells at the next stage (or the largest load in general) is sized much larger than the others, resulting in an overall electrical fanout closer to 2. This optimization of electrical-vs-logical fanout arises due to the possibility of differential sizing of the gates at the same stage. With its highest F of 17, Sklansky adder can take maximum advantage of differential sizing.

Unlike F, L has a real cost. A Brent-Kung adder has the same T, but almost twice the L as the Sklansky adder. This may seem useful for driving a large Cout, but inverters are far more efficient than P/G gates and can always be padded to a lower logic depth design to make up for the required gain at lower energy cost.

Like L, T also has real costs. A larger T means a higher portion of the total energy consumed in wires. With its smallest T of 1, the Sklansky design spends most of the energy budget in driving useful logic, with the least amount wasted in wires. A Kogge-Stone adder, having the largest T, suffers from high energy loss to result in poor energy efficiency. Figure 3.8 shows the percentage of total adder energy[8] consumed in wires for the three adders. Note that wire energy changes with V_{dd}, which changes with the optimal E-D point.

[8]This excludes the energy consumed in output loads (Cout).

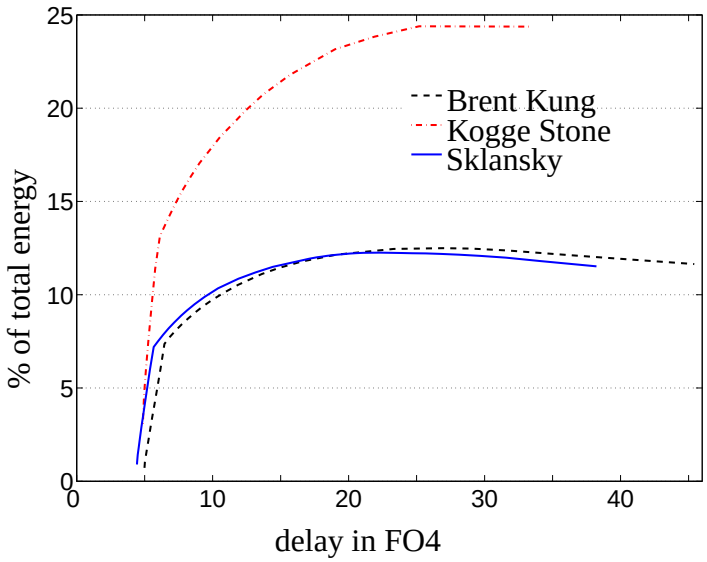


Figure 3.8: Energy consumed in wires.

At lower energies, resources are constrained and adders with fewer gates have an advantage. The Brent Kung adder is most economical in gate count. Hence it comes closer to Sklansky at lower energies, unlike Kogge-Stone, which has a comparable number of gates to Sklansky, but more wires.

To confirm our inferences about the Sklansky design we optimized its two closest radix 2 adders – a LF adder with an extra logic level ($(R, T, L) = (2, 1, 6)$) and a Knowles adder with an extra wire track ($(R, T, L) = (2, 2, 5)$). We also optimized a 2-bit sum select Sklansky adder to check if reduced fanout ($F = 8$) in Sklansky CPL at the cost of increased fanout for the final carry gives any benefit. We found that the cost of generating the conditional sum in SGL was more than the advantage of having CPL and SGL in parallel. The results shown in Figure 3.9 confirm that Sklansky was better than its three closest relatives. Due to lower gate count, the selected Ladner-Fischer adder tends to compete with Sklansky adder at lower energies.

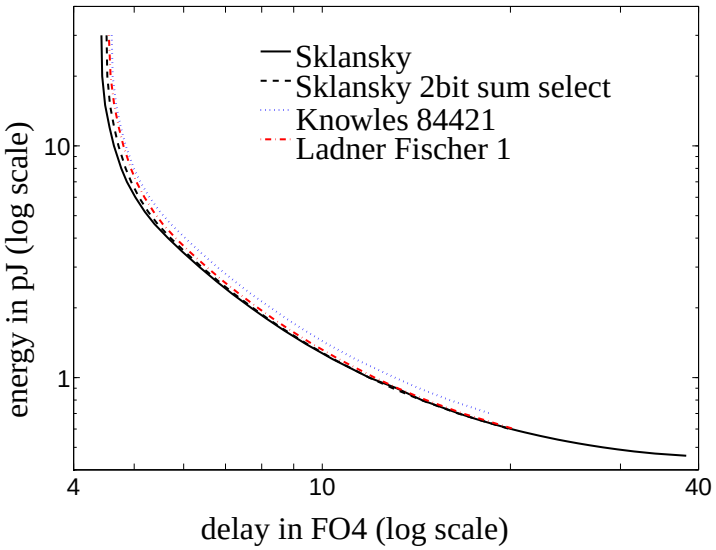


Figure 3.9: Comparison of Sklansky adder E–D curves to its closest neighbors and to a 2-bit sum select scheme.

Other logic styles and topologies

Given that the effect of loading and parasitics is similar in all logic styles, the topology that is the most energy-efficient in one, will be the most energy-efficient in the other as well. Following are the results of our experiments on other logic styles designed on the basis of the results from static CMOS logic.

1. **Domino and dual rail designs:** We made Sklansky designs in radix 2 domino and dual rail domino logic[9]. Similar to the static case, a fully dynamic 2-bit sum select scheme does not give any benefit. However, a 2-bit sum select using static SGL improves the energy-efficiency due to lower activity factor in the SGL. On the other hand, dual-rail designs consume almost twice the energy compared to domino designs with the only benefit that the XORs in the SGL are faster. Hence they are better

[9]Because the fanin of the domino gate (dynamic gate and inverter) is 4, some researchers [100] call this radix 4.

than the domino designs only at high energy by a small amount. Figure 3.10 shows the E-D curves of selected Sklansky domino/dual-rail topologies.

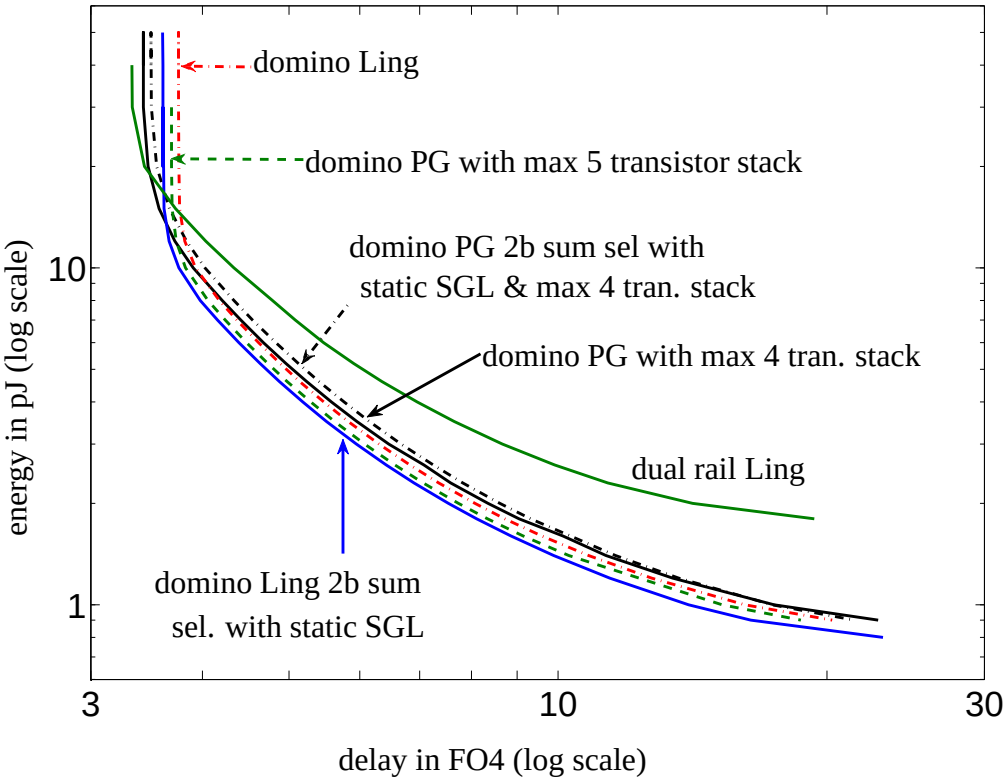


Figure 3.10: Comparison of Sklansky domino and dual-rail domino tree adders.

Ling adder performs about 5% better in delay than the PG adder if only 4 transistor stacks are allowed, but it is worse than an equivalent PG adder (which will have a 5 transistor stack in the first dynamic gate). However, as mentioned before in Section 2.1.4, our modeling of the parasitic delay is most optimistic in the 5 stack gate, so we expect the two designs to be comparable in practice. Due to large fanin right at the inputs, the Cin constraint becomes active in these adders pretty early on, which is why the 4 stack PG adder looks better at higher energy.

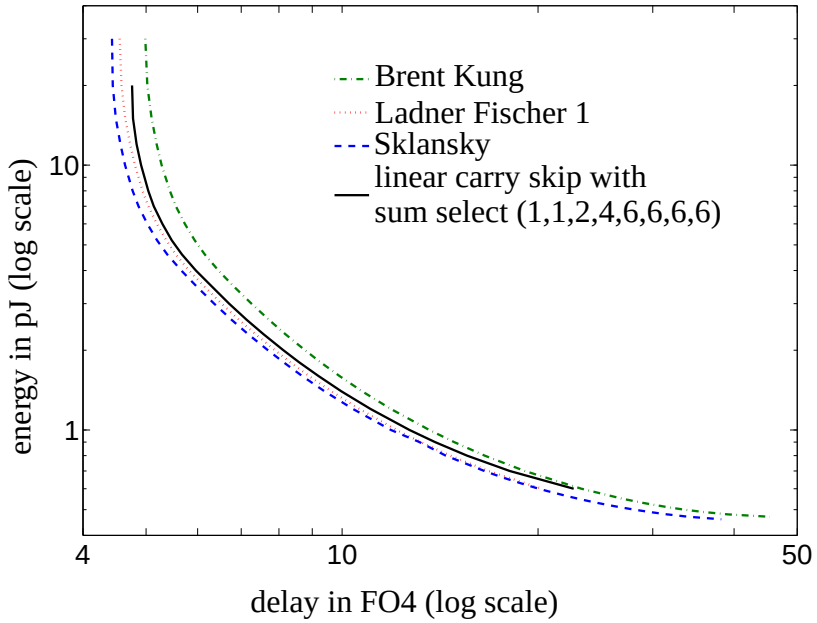


Figure 3.11: Comparison of the linear adder to closest tree adders. The Brent Kung adder is also shown for reference.

2. **Linear adders:** While a linear ripple-carry adder is extremely inefficient due to its large logic depth, a full 32-bit Manchester carry chain adder [96] suffers from excessive parasitic delay. Carry skip and sum selection techniques exploit parallelism by overlapping the PG generation of a block of bits with the ripple carry inside the block. If, using similar gates, linear adders can be designed to have similar number of stages and wire tracks as the best tree adders, they should be equally energy-efficient. We designed such a linear carry-skip sum-select adder, with block sizes of 1,1,2,4,6,6,6 and 6, resulting in 9 logic stages, similar to LF1 adder. Figure 3.11 clearly shows that not surprisingly, with a T equal to that of a Knowles 84421 adder, this linear adder compares well with the tree designs.

While their L is comparable for $N = 32$ bits, the logic depth of linear adders increases faster than that of a tree structure, where it is logarithmic with N. For example, for $N = 64$, L for Sklansky adder increases by unity, whereas for the best linear adder, it increases by 3. Hence linear adders are less efficient for higher N.

Optimal radix

From our definition of the radix, a logical operation performed on N inputs in p stages has a radix of $\log_p N$ regardless of the kind of gates used. For delay, p is optimal ($p_{\rm opt}$) when each stage has a delay of about a FO4 [87]. The value of $p_{\rm opt}$ depends on the load and the kind of gates used in the design. A gate doing more logic than simple inversion requires more effort and hence is inherently slower. Thus gates in higher radix adders are inherently slower. Accounting for this logical effort, calculation shows that for all reasonable adder loads (where Cout is at least about Cin), the number of stages in a minimum logic depth radix 2 adder is about equal to or less than their respective $p_{\rm opt}$. The logic depth of higher radix adders may fall short of $p_{\rm opt}$ but that can be made up by external buffering. Even for bigger adders (i.e. $N \geq 32$), $p_{\rm opt}$s for $R \geq 2$ are about the same [29] and big enough to accommodate the logic depth of all adders with $R \geq 2$.

Although for minimum delay, the delay per stage should be equal, the inherent slowness of the higher fanin gates (in higher radix adders) coupled with higher logical fanout per stage restricts the delay per stage to be always higher than a certain amount[10]. Also, because parasitic delay grows at least as the square of the number of inputs [96], the parasitic delay of higher fanin gates dominates any gain from the reduced L, given that inverters need to be padded to reach $p_{\rm opt}$ stages. Complex high fanin gates like the ones that produce the group generate signals grow in the number of parallel transistor stacks as well, significantly

[10] for reasonable output load (Cout) on the adder.

increasing their parasitic delay. Therefore a radix of 2 is optimal for delay.

From the energy point of view, using higher fanin gates does not save on switching activity. Adders being multiple output structures, the intermediate signals generated by using trees of lower fanin gates are typically used for other computation. Also, while the propagate function benefits from having a large fanin gate, as it is an AND function and activity factor reduces with more inputs, switching simulations show that the generate operation maintains the switching factor. Hence overall, higher fanin PG cells have only a marginal reduction of switching activity over trees of lower fanin cells. For domino designs however, because NMOS is faster than PMOS, it is better to have series NMOS stacks in the dynamic gate than having PMOS stacks in the following static stage. Hence the gates with four input dynamic gates followed by an inverter are more efficient than gates with two input dynamic stage followed by a two input static stage. Note that the radix is 2 in both cases.

To validate our intuition, we designed static CMOS 32-bit Sklansky adder, one with radix 3 and another with radix $\sqrt{6}$ (using alternate 3 bit and 2 bit combine stages). To avoid the irregularity of a 32-bit radix 3 adder, we also designed 27-bit adders with radix 2 and 3. Figure 3.12 compares the E-D tradeoff and confirms our understanding. The results remain unchanged for Cout = 25f.

Effect of buffers on energy-efficiency

External buffers increases the number of stages in the adder. Hence one might expect that while they are useful for adders with small logic depth driving large loads, they would be inefficient for an adder with L that is already at or bigger than the optimal L. However, inverters are the most efficient drivers. Hence, in addition to buffering, padding inverters at the output leads to a reduction in the size of the complex gates that precede them, thus

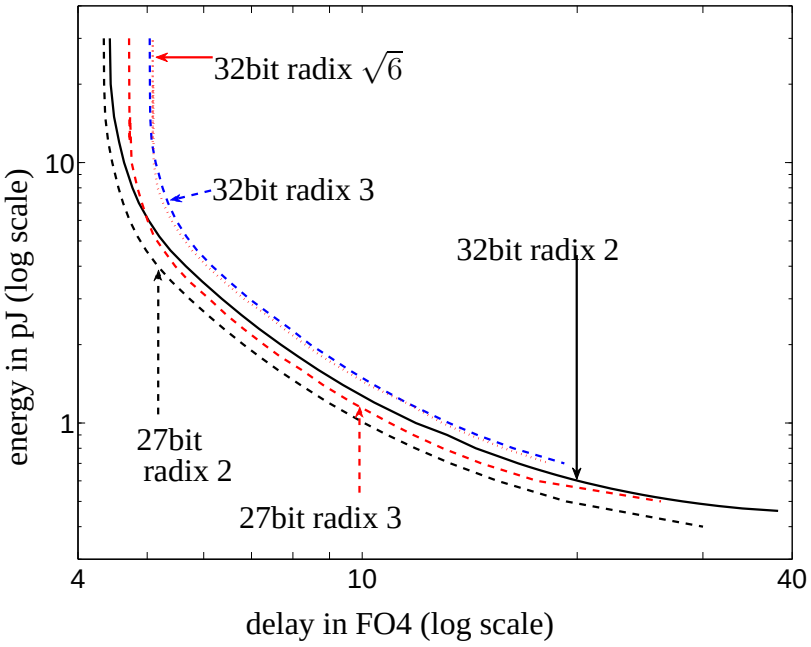


Figure 3.12: E–D tradeoff curves of 32-bit and 27-bit Sklansky adders of different radices.

saving energy. The reduction of gate sizes on the side paths helps to reduce the load on the critical path and more than compensates for the increased delay due to the extra logic stage. Figure 3.13 shows that while buffering is inefficient in the high energy region due to the delay added by the extra inverter stage(s), as the energy budget is reduced, the same design padded with a single inverter stage does much better than the original. The potential increase in delay by adding an extra logic stage is more than compensated by the energy benefit from smaller SGL gates, even in the Brent Kung adder, which already has a large L. In fact, all the E-D curves shown in this paper are with a single inverter padding, except for the dual rail ling adder design, which was more efficient without external buffering. This is expected, because there the outputs are already driven by the inverter of the dual rail domino gate. Selective padding of different paths can possibly bring more gains, but would affect the logical functionality of the design.

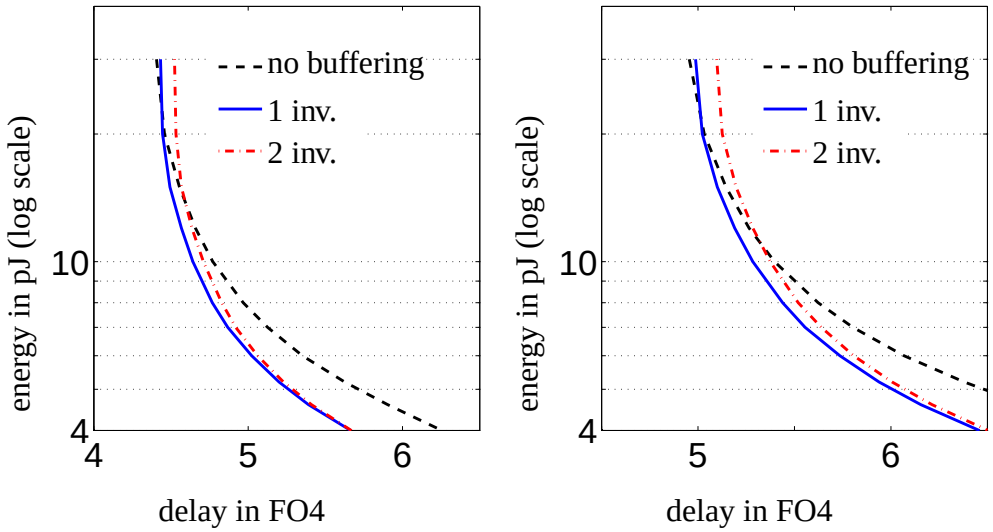


Figure 3.13: Effect of external buffering on radix 2 Sklansky (left) and Brent Kung (right) adders.

3.4.4 Adder design space

The E-D tradeoff curves of the different adders across different logic styles intersect at points where one adder achieves better energy efficiency than the other. Static adders are good at low performance regions because, with lower switching activity factor and no clock load, they consume lower energy; but they saturate quickly as higher performance is desired, due to inherently slower gates (large logical effort). With lower logical effort gates, dynamic adders have potentially higher performance and at higher energy they perform much better than their static counter parts. The overall Pareto-optimal curve is the lower bounding curve of all these curves. It gives an indication of the E-D space of optimally designed adders. Figure 3.14 shows the complete 32-bit adder E-D space. Incidently, at or before the point where Cin constraint becomes active, the next logic family takes over the pareto-optimal curve. We can also observe that sensitivity of energy-efficiency to R,T, and L depends on the location on E-D curve. At low energy side, wire tracks T have a large

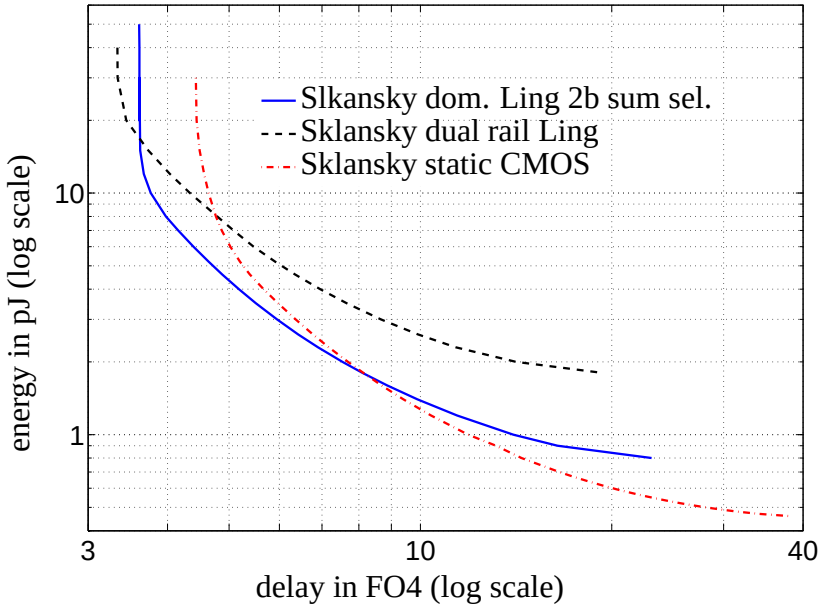


Figure 3.14: 32-bit adder E-D space. $C_{\text{load}} = 100fF$

weight because they consume a greater portion of the energy budget. At low delay points logic depth L matters more because every added stage adds a minimum delay, which causes external buffering to be inefficient at high energies and enables the Kogge-Stone E-D curve to cross the Brent-Kung one.

3.5 Summary

The goal of a circuit sizer is to optimally allocate the given area/energy among the different devices in the netlist for maximum possible performance. The delay and energy of a gate can be accurately modeled as posynomials. While the total energy is just a sum of many posynomial terms, the overall delay can be easily obtained using sum and $\max$ operations on gate delays, with static signal timing propagation in the circuit netlist. Using these

elements, the deterministic circuit optimization problem can be cast as a Geometric Program, which can be solved efficiently. Although some important circuit issues like slope constraints and transmission gate timing constraints are either difficult to model and/or inherently non-convex, suitable work-around can be obtained in a convex framework, to drive the optimizer to do correct sizing.

We developed this optimization framework in a tool called SCOT. SCOT helped us to efficiently obtain the tradeoff curves for different 32-bit adder topologies. This information not only helped us to gain valuable insights regarding the factors that make circuits energy-efficient, but also gave us an overall picture of the energy-delay design space of 32-bit adders.

The optimization shown in this chapter did not include variations as one of the parameters. In the next chapter, we will show how variations can significantly reduce the yield of optimized designs and then describe the solutions we have come up with to tackle the statistical design problem.

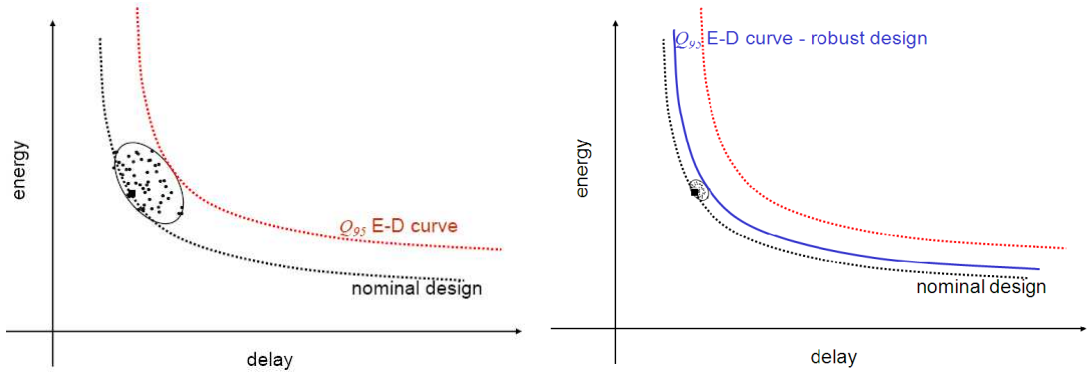
Chapter 4

Optimization for robustness

In Chapter 2, we described the different process variations that can occur during fabrication. Variations cause the delay and energy of the fabricated designs to have a distribution around the original specifications. Thus variations cause the fabricated chips designed for one design point to scatter over a region in the energy-delay space as shown in Figure 4.1(a). Some of these designs still lie on the Pareto-optimal curve and are therefore energy-efficient. These can be binned for selling them with different energy-delay specs. It is very unlikely that variations may result in a design that beats the Pareto-optimal curve, as it is extremely unlikely that all the parameters in the chip are improved in the right direction. Most of the scattered designs are on the upper-right of the Pareto-optimal curve, and therefore not energy-efficient. The parametric yield[1] of a design is defined as the percentage of functional fabricated designs that meet or exceed the specs. If we desire a certain yield in the scenario in Figure 4.1(a), the specification of the product will have to be relaxed. This gives us a new energy-delay tradeoff curve for the fabricated designs that meet the new specs. This curve can be greatly inferior to the curve we designed for. We really want to

[1] as opposed to functional yield which is the percentage of designs that have no functional defects. Unless there is ambiguity, henceforth in this thesis, we will refer to parametric yield simply as yield

optimize the efficiency of the circuits that are actually produced (i.e the yield curve). Including the process variation information in the circuit design process can lead to design of robust circuits that would tolerate variations better. Such a design methodology where variation information is incorporated in the design phase itself is called "statistical design", as opposed to the traditional "deterministic design" where all device parameters are assumed to be fixed to some average value or are assumed to be completely correlated with each other. The design resulting from the use of only the nominal or average parameter values is referred to as "nominal design", while the design resulting from statistical design methodology will be termed as "robust design". As shown in Figure 4.1(b), a robust design has a tighter distribution and hence suffers a smaller degradation of the original Pareto-optimal curve.



(a) Q_{95} E–D curve of nominal design (b) Improved E–D curve for robust design

Figure 4.1: Possible improvement in Q_{95} E–D curve with design for robustness

Under variations, the delay and energy of a circuit become random variables with a Probability Density Function (PDF) which depends on the PDF of the individual gate delays and energies. The robust Pareto-optimal tradeoff curve for a particular yield, say, α can be obtained by optimizing such that α percentile of the implementations have specs that are better than the one we designed for. In this chapter we will attempt to solve this

statistical design problem.

The techniques to statistically handle delay and energy are different due to the way delay and energy relate to the design variables. We shall first focus on delay variations and then focus on energy. In order to understand how individual gate delay variations contribute to the overall delay distribution, we describe different methods of estimating the circuit timing under gate delay uncertainty [45]. These analysis techniques will be useful as a guide to statistical design. Using statistical analysis, we show how local independent process variations adversely affect the delay of deterministically optimized designs from the previous chapter. It is feasible to solve the statistical design problem exactly in some special simple circuits. We discuss the formulation of the exact solution of statistical design problem. The exact solution is extremely tedious to apply to typical circuits. To overcome this problem we have derived some effective heuristics based on insights from the exact solution and statistical analysis, which we shall describe next.

4.1 Estimating performance bounds

Consider the overall circuit delay $\mathbf{T}_\mathrm{d}$, given by

$$\mathbf{T}_\mathrm{d} = \max\{d(p) \mid p \in \mathcal{P}\},$$

where $\mathcal{P}$ denotes the set of all paths and $d(p)$ represents the delay of path p. We will consider energy estimates in Section 4.6. The PDF of $\mathbf{T}_\mathrm{d}$ depends on the extent and nature of process variations. Several scalar performance measures can be used to characterize the delay metric of the circuit. Examples include:

- Expected value: $\mu(\mathbf{T}_\mathrm{d})$.

- Probability of missing a target T_0: $\mathbf{prob}(\mathbf{T}_\mathrm{d} \geq T_0)$.
- Expected tardiness: $\mu(\max\{\mathbf{T}_\mathrm{d} - T_0, 0\})$.
- α-percentile of $\mathbf{T}_\mathrm{d}$: $Q_\alpha(\mathbf{T}_\mathrm{d}) = \inf\{t : \mathbf{prob}(\mathbf{T}_\mathrm{d} \leq t) \geq \alpha\}$,

where $\inf\{\}$ is the infimum[2] of a set of numbers.

Exact Statistical Static Timing Analysis (SSTA) which extends the traditional Static Timing Analysis (STA) by propagating PDFs instead of numbers is very difficult, except in a few special cases. The problem is that the evaluation of gate delay as given in Equation 3.1 involves both, addition and maximum of random variables. While there are common families of distributions that are closed under addition, and others that are closed under maximum (or minimum), no practical family of distributions for gate delay is closed under both.

We can, however, say many things about the distribution of $\mathbf{T}_\mathrm{d}$. For example, if gate delay $\mathbf{D}$ is Gaussian, then delay of each path through the netlist is a Gaussian and $\mathbf{T}_\mathrm{d}$ is the maximum of a number of correlated Gaussian random variables. There are many bounds and asymptotics known for such distributions [78]. A general and common approach is based on approximating or bounding the distributions of the gate output timings, by recursively bounding (or approximating) the delay distribution of gate i in combination with bounds (or approximations) of the distributions of the signal arrival times at its input [20, 22, 26, 52, 58, 74, 79, 95]. Let T_nom represent the delay of the nominal design where the gate delays are assumed to be fixed to their mean values. When the distribution of $\mathbf{D}$ is very tight, *i.e.*, $\mathbf{D}$ is very close to its mean $\mu(\mathbf{D})$ with high probability, we expect T_nom to give a good approximation of $\mathbf{T}_\mathrm{d}$.

[2]The infimum of a subset of some set is the greatest element, not necessarily in the subset, that is less than or equal to all other elements of the subset.

In fact, $T_{\rm nom}$ is always an underestimator of $\mathbf{T}_{\rm d}$, in the following sense.

$$T_{\rm nom} = T(\mu(\mathbf{D})) \leq \mu(\mathbf{T}_{\rm d}) = \mu(T(\mathbf{D})), \tag{4.1}$$

where $T()$ represents the overall circuit delay as a function of gate delays. This holds for any distribution on $\mathbf{D}$. One interpretation of this inequality is that by adding zero-mean statistical variation to any gate delay, we can only increase the expected value of the overall delay. This inequality is a direct application of Jensen's inequality [10], along with the observation that $T()$ is a convex function of the gate delays. Convexity follows since $T()$ is the maximum of a set of sums – a linear function – of gate delays [10].

Equality holds in (4.1) if and only if there is a unique path that is always critical [32]. The criticality of a path is defined as the fraction of manufactured instances it has the highest delay. Similarly the criticality of a gate can be defined as the fraction of manufactured circuits in which it appears on a critical path. If there are relatively few paths with high criticality, the difference between the left and righthand sides of (4.1) can be relatively small. In other cases, *e.g.*when all gates uniformly have low criticality, the difference can be relatively large as we shall see in Section 4.2.

4.1.1 Performance bounds via stochastic dominance

Let $F_{\mathbf{U}}(t) = \mathbf{prob}(\mathbf{U} \leq t)$ denote the Cumulative Distribution Function (CDF) of the scalar random variable $\mathbf{U}$. A scalar random variable $\mathbf{X}$ is said to be *stochastically less than or equal to* another scalar random variable $\mathbf{Y}$ (denoted by $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$) if $F_{\mathbf{X}}(t) \geq F_{\mathbf{Y}}(t)$ holds for all t. Stochastic inequality can also be expressed in terms of percentiles. Let $Q_\alpha(\mathbf{U}) = \inf\{t : \mathbf{prob}(\mathbf{U} \leq t) \geq \alpha\}$ denote the α-percentile of $\mathbf{U}$. Then, $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$ if and only if $Q_\alpha(\mathbf{X}) \leq Q_\alpha(\mathbf{Y})$, for all $\alpha \in (0, 1)$. All of the performance measures described

in the beginning of Section 4.1 are monotone with respect to stochastic dominance. For example, if $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$, the expected tardiness of $\mathbf{X}$ is no more than the expected tardiness of $\mathbf{Y}$.

A basic result is that for any random variables $\mathbf{X}_1, \ldots, \mathbf{X}_p$, no matter what their joint distribution is, their maximum is always stochastically greater than or equal to each of them:

$$\mathbf{X}_i \leq_{\mathbf{st}} \max\{\mathbf{X}_1, \ldots, \mathbf{X}_p\}, \quad i = 1, \ldots, p.$$

Since $\mathbf{T}_{\mathrm{d}}$ is the maximum of all path delays, it is always stochastically greater than or equal to the delay of any path, no matter what the duration distributions are, and whether or not they are independent. We conclude that for each of the performance measures described in Section 4.1, the maximum of the performance measure over all paths is a lower bound on the performance measure of $\mathbf{T}_{\mathrm{d}}$. For example, the α-percentile of $\mathbf{T}_{\mathrm{d}}$ satisfies

$$\max_{p \in \mathcal{P}} Q_\alpha(\mathbf{D}_p) \leq Q_\alpha(\mathbf{T}_{\mathrm{d}}), \tag{4.2}$$

where $\mathcal{P}$ is the set of all paths, and $\mathbf{D}_p$ is the delay of path p. This gives us a method for obtaining a lower bound on a performance measure when the durations are Gaussian. In this case, each path delay is Gaussian, and its performance measure can be calculated exactly as a function of its mean and variance. By taking the maximum of these measures over all of the paths (or a subset) we obtain a lower bound on the performance measure for $\mathbf{T}_{\mathrm{d}}$. (Unfortunately, there is no simple recursion, like in static timing analysis to calculate T_{nom}, for calculating the maximum of the α-percentile over all paths.)

For future use, we give a very simple lower bound on percentiles of $\mathbf{T}_{\mathrm{d}}$. By Jensen's inequality, T_{nom} is a lower bound on $\mu(\mathbf{T}_{\mathrm{d}})$. Since gate delays are non-negative and have close to Gaussian or otherwise well behaved distributions, we can argue that for every path,

its expected delay is less than or equal to its α-percentile for values of α of interest, such as $\alpha = 0.95$.

$$\mu(\mathbf{D}_{i_1} + \cdots + \mathbf{D}_{i_k}) \leq Q_\alpha(\mathbf{D}_{i_1} + \cdots + \mathbf{D}_{i_k}), \qquad \forall\, p = (i_1, \ldots, i_k) \in \mathcal{P}. \tag{4.3}$$

With this assumption, T_{nom} is a lower bound on the α-percentile of $\mathbf{T}_{\mathrm{d}}$:

$$T(\mu\mathbf{D}) = \max_{p=(i_1,\ldots,i_k)\in\mathcal{P}} \{\mu(\mathbf{D}_{i_1}) + \cdots + \mu(\mathbf{D}_{i_k})\} \leq Q_\alpha(\mathbf{T}_{\mathrm{d}}). \tag{4.4}$$

4.1.2 Performance bounds via surrogate netlists

A hypothetical netlist where the delay of every gate i, d_i is fixed and given by $\tilde{d}_i = \mu(\mathbf{D}_i) + \kappa_i\sigma(\mathbf{D}_i)$, where $\sigma(\mathbf{U})$ denotes the standard deviation of a scalar random variable $\mathbf{U}$, is called a surrogate netlist of the original netlist. We call $\kappa_i \geq 0$ the *margin coefficients*. Note that the original netlist is the surrogate netlist with all margin coefficients zero. We can derive some bounds on the percentiles (or other measures) of $\mathbf{T}_{\mathrm{d}}$ from the STA of its surrogate netlist for proper choice of margin coefficients.

We consider the case in which the gate delay distributions are independent and Gaussian. The delay of path $p = (i_1, \ldots, i_k)$ is also Gaussian, and its α-percentile can be expressed as

$$Q_\alpha(\mathbf{D}_p) = \mu(\mathbf{D}_{i_1}) + \cdots + \mu(\mathbf{D}_{i_k}) + \Phi^{-1}(\alpha)\left(\sigma(\mathbf{D}_{i_1})^2 + \cdots + \sigma(\mathbf{D}_{i_k})^2\right)^{0.5}$$

where

$$\Phi(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-t^2/2}\, dt \tag{4.5}$$

is the CDF of a unit Gaussian variable.

We will now relate $Q_\alpha(\mathbf{D}_p)$ to the delay of a path in a surrogate netlist, using the Cauchy-Schwartz inequality

$$(u_1^2 + \cdots + u_k^2)^{0.5} \geq a_1 u_1 + \cdots + a_k u_k, \tag{4.6}$$

provided $a_1^2 + \cdots + a_k^2 \leq 1$. This gives

$$Q_\alpha(\mathbf{D}_p) \geq \mu(\mathbf{D}_{i_1}) + \cdots + \mu(\mathbf{D}_{i_k}) + \Phi^{-1}(\alpha)(a_1\sigma(\mathbf{D}_{i_1}) + \cdots + a_k\sigma(\mathbf{D}_{i_k})),$$

provided $a_1^2+\cdots+a_k^2 \leq 1$. Note that the righthand side here is the delay of the path p, in the surrogate netlist with $\kappa_i = \Phi^{-1}(\alpha)a_i$. This operation essentially allows us to break open the root mean square (RMS) calculation of the standard deviation of a path and convert it into additive terms.

We can make several simple choices of the a_i so that the requirement $a_1^2 + \cdots + a_k^2 \leq 1$ holds for every path. One simple choice is

$$a_i = l_{\max}^{-1/2},$$

where $l_{\max}$ is the maximum length of any path in the netlist. Another choice is

$$a_i = l_i^{-1/2},$$

where

$$l_i = \max\{l(p) \mid \text{path } p \text{ contains gate } i\}$$

is the length of the longest path that contains gate i.

The quantity $l_{\max}$ is readily computed using recursion as used in STA where $l_{\max} =$

$T_{\rm nom}$ with all gate delays equal unity. A variation on this recursion can be used to efficiently calculate the quantities l_i. We use a STA recursion to compute the length of the longest path from inputs to gate i, and another recursion, starting at the outputs and working backward, to find the length of the longest path from each gate i to any output. We add these two quantities at each node to obtain l_i.

The bounds above, along with (4.2), imply that the delay of the surrogate netlist $\tilde{T}_{\rm nom}$, with the choice of margin coefficients

$$\kappa_i = \Phi^{-1}(\alpha) {l_{\max}}^{-1/2}, \tag{4.7}$$

or the more sophisticated choice

$$\kappa_i = \Phi^{-1}(\alpha) l_i^{-1/2}, \tag{4.8}$$

is a lower bound on α-percentile of $\mathbf{T}_{\rm d}$. Note that the timing of a surrogate netlist can be computed efficiently using STA.

The same lower bound on the $Q_\alpha(\mathbf{T}_{\rm d})$ holds with correlated Gaussian duration distributions, provided that the covariance of any two gate delays is nonnegative. When this is the case, the standard deviation of the delay of any path is less than or equal to the standard deviation when the gate delays have the same standard deviations, but are uncorrelated.

It is also possible to obtain *upper bounds* on a performance measure, such as the α-percentile of $\mathbf{T}_{\rm d}$, that have a similar form. As an example, we consider the case where the gate delays are Gaussian and can be correlated. Let $|\mathcal{P}|$ be the total number of paths from sources to sinks and $\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}$ denote the delays of all such paths, *i.e.*,

$$\mathbf{T}_{\rm d} = \max\{\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}\}.$$

Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_{|\mathcal{P}|}$ denote *independent* random variables whose distributions are identical to those of $\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}$, respectively, that is,

$$\tilde{\mathbf{T}}_d = \max\{\mathbf{Z}_1, \ldots, \mathbf{Z}_{|\mathcal{P}|}\}.$$

A basic result on stochastic comparison between random variables shows that

$$\mathbf{T}_\mathrm{d} \leq_\mathrm{st} \tilde{\mathbf{T}}_\mathrm{d}.$$

It follows that the α-percentile of $\mathbf{T}_\mathrm{d}$ is less than or equal to that of $\tilde{\mathbf{T}}_d$

$$Q_\alpha(\mathbf{T}_\mathrm{d}) \leq Q_\alpha(\tilde{\mathbf{T}}_\mathrm{d}).$$

Now the idea is to compute $Q_\alpha(\tilde{\mathbf{T}}_d)$, which is relatively easy. Since $\mathbf{Z}_i$ are independent of each other, the righthand side can be expressed as

$$Q_\alpha(\tilde{\mathbf{T}}_d) = \inf\left\{t \mid \prod_{i=1}^{|\mathcal{P}|} F_{\mathbf{Z_i}}(t) \geq \alpha\right\}. \tag{4.9}$$

We do not know what $\inf\{t\}$ is, but we are looking for an upper bound. So let us choose t that is high enough to satisfy Eq. 4.9 as

$$t = \max_{i=1,\ldots,|\mathcal{P}|}(\mu(\mathbf{Z}_i) + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i)).$$

Since $\mathbf{Z}_i$ are Gaussian, we have $F_{\mathbf{Z_i}}(t) \geq \alpha^{1/|\mathcal{P}|}$, and hence $\prod_{i=1}^{|\mathcal{P}|} F_{\mathbf{Z_i}}(t) \geq \alpha$. This along with (4.9) leads to the inequality

$$Q_\alpha(\tilde{\mathbf{T}}_d) \leq \max_{i=1,\ldots,|\mathcal{P}|}(\mu(\mathbf{Z}_i) + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i)).$$

We now show how to relate the righthand side of the inequality above to the delay of a path in a surrogate netlist. Let $\mathbf{Z}_i$ be the delay of a path consisting of gates $i_1, \ldots, i_k$:

$$\mathbf{Z}_i = \mathbf{D}_{i_i} + \cdots \mathbf{D}_{i_k}.$$

Since $(x_1^2 + \cdots x_k^2)^{0.5} \leq x_1 + \cdots + x_k$ for $x_i \geq 0$, we have

$$\begin{aligned} \mu(\mathbf{Z}_i) + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i) &= \sum_{j=1}^{k} \mu\mathbf{D}_{i_j} + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\Big(\sum_{j=1}^{k} \sigma_{i_j}(\mathbf{D}_{i_j})^2\Big)^{0.5} \\ &\leq \sum_{j=1}^{k} \Big(\mu(\mathbf{D}_{i_j}) + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_{i_j}(\mathbf{D}_{i_j})\Big). \end{aligned}$$

The righthand side is the delay of the path $p = (i_1, \ldots, i_k)$ in the netlist with gate delays $\mu(\mathbf{D}_i) + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{D}_i)$. Now, we have the bound

$$Q_\alpha(\mathbf{T}_\mathrm{d}) \leq \gamma,$$

where γ is the delay of the surrogate netlist with gate delays $\mu(\mathbf{D}_i) + \kappa\sigma_i(\mathbf{D}_i)$, where

$$\kappa = \Phi^{-1}(\alpha^{1/|\mathcal{P}|}).$$

The values of κ required to obtain the upper bound taper off roughly as the log of the number of paths as shown in Figure 4.2. For example, for a netlist where the number of all paths is 3000, the choice of $\kappa = 4.14$ gives an upper bound for its 0.95-percentile. Of course, the upper bound thus obtained on the α-percentile is tighter if paths in a netlist are mostly independent.

To summarize, we have described some computationally efficient and relatively simple performance bounds. These bounds require only the means and variances of the gate delay,

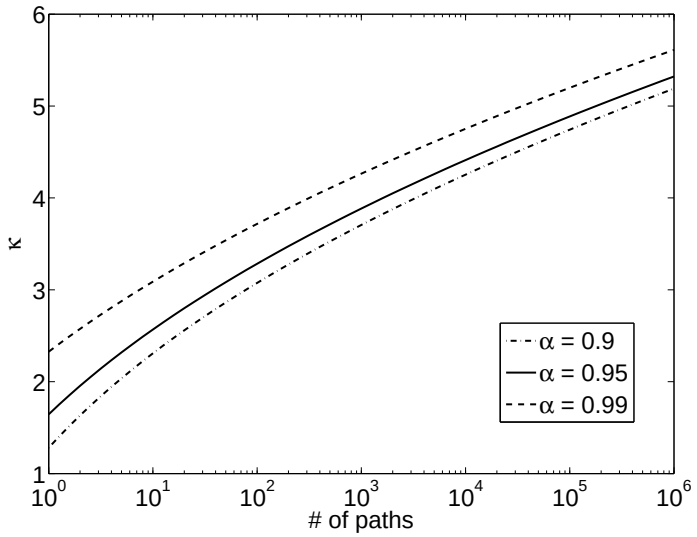


Figure 4.2: κ required for estimating the upper bound of $Q_\alpha(\mathbf{T}_\mathrm{d})$

and may not be as accurate as others that have been proposed, *e.g.*, the one given in [4], which rely on the same information. But the main purpose of the bounds given in this section is not statistical *analysis* which can be done efficiently and accurately by Monte Carlo analysis. The bounds given in this section can be used to provide bounds on suboptimality of a statistical *design* and develop heuristics for robust sizing. We describe this in Section 4.4. In Section 4.1.4, we shall see how these bounds perform on some representative circuits.

4.1.3 Monte Carlo analysis

While exact analysis of the distribution of $\mathbf{T}_\mathrm{d}$ is very difficult, Monte Carlo methods can be used to approximately compute the distribution, along with the performance measures or other quantities of interest. In this section we describe basic Monte Carlo analysis of a netlist; more sophisticated methods can be used to get higher accuracy with fewer samples,

or combine Monte Carlo simulation with bounding methods [13, 81, 86].

In basic Monte Carlo analysis, we draw M independent samples $d^{(1)}, \ldots, d^{(M)}$ from the distribution of the random duration vector $\mathbf{D}$, and create M distinct netlist instantiations. For each instance, we can efficiently evaluate its delay using STA, to obtain $T_{\mathrm{nom}}^{(1)}, \ldots, T_{\mathrm{nom}}^{(M)}$. The sampled delays $T_{\mathrm{nom}}^{(1)}, \ldots, T_{\mathrm{nom}}^{(M)}$, are, of course, independent samples from the distribution of $\mathbf{T}_{\mathrm{d}}$.

Percentile estimation

To estimate $Q_\alpha(\mathbf{T}_{\mathrm{d}})$ from the samples $T_{\mathrm{nom}}^{(1)}, \ldots, T_{\mathrm{nom}}^{(M)}$, we first re-order them so that

$$T_{\mathrm{nom}}^{(1)} \leq \cdots \leq T_{\mathrm{nom}}^{(M)}.$$

Once re-ordered, $T_{\mathrm{nom}}^{(k)}$ is called the k^{th} *order statistic* of $\mathbf{T}_{\mathrm{d}}$. A simple estimate $\hat{Q}_\alpha(\mathbf{T}_{\mathrm{d}})$ of the percentile is given by

$$\hat{Q}_\alpha(\mathbf{T}_{\mathrm{d}}) = T_{\mathrm{nom}}^{(\lceil M\alpha+1 \rceil)}$$

where $\lceil x \rceil$ denotes the integral part of $x \in \mathbf{R}$. This estimate is asymptotically consistent and its variance is inversely proportional to M under the mild assumption that the PDF of $\mathbf{T}_{\mathrm{d}}$, $f_{\mathbf{T}_{\mathrm{d}}}(Q_\alpha(\mathbf{T}_{\mathrm{d}}))$, is always positive [93].

Criticality index estimation

Under mild assumptions, any instance of the circuit has a unique critical path, with probability one[3]. The criticality index of a gate or a path from an input to an output can be estimated by counting the fraction of the realizations in which it is critical. These estimates

[3]For instance, if the distribution of $\mathbf{D}$ is continuous, then the joint distribution of path delays is continuous, and so the probability that the delays of any two paths are the same is zero [93].

are consistent, and the variance of the estimation error is inversely proportional to M [25]. Criticality indexes are computed during the Monte Carlo simulations.

4.1.4 Bound estimates for representative circuits

Table 4.1 shows the results of the our statistical analysis experiments on different circuits. Figure 4.3 shows the same information in a bar graph. The circuits starting with "c" are ISCAS'85 benchmark circuits [27]. The delays are in FO4[4]. These circuits represent different building blocks of a digital system and have different topological characteristics that will affect the tightness of the bounds we obtain. For example, dec8-256 has all its paths almost identical while add32BK has a unique path with highest number of gate stages. We assume independent gate delay variations and use Pelgrom's model described in Section 2.3 with parameters such that the standard deviation of the drive current ($\sigma(I_{\rm d})$) for 1μ minimum length device width is 15%. For calculating $\sigma(I_{\rm d})$ for a stack of transistors, $L_{\rm eff}$ and $W_{\rm eff}$ as modeled in Equation 2.8 are used. As our focus is first on robust *sizing*, for these analyses, we use a simplified version of our delay model by fixing $V_{\rm dd}$ and $V_{\rm th}$ to nominal values. The circuits used here are sized to be roughly in the middle of their area-delay tradeoff curves for reasonable loads. We use area constraint (instead of energy) as we are focusing only on delay variations and area given as the sum of the widths is relatively invariant with process variations.

The results show that accuracy of the bounds is highest for a single inverter chain and reduces as the circuits become more complex. The gross inaccuracy of the upper bounds primarily results from two reasons. First, contrary to our assumption for evaluating the upper bound, paths in a typical circuit share many common gates and therefore are highly correlated. Second, as the paths get longer, the error between taking the sum of the gate

[4] 1FO4 = 31ps in our technology.

delay standard deviations vs. taking their root means square value increases. The latter explains why the upper bound is inferior even in the inv5 benchmark, which is a single chain of inverters. The lower bound estimate is worse when only a few gate stages dominate the delay variation of the dominating paths, because the lower bound underestimates their effect by dividing their sum by the square root of the path length[5] (Eq. 4.8). Observing the sizing in inv5 reveals that the relatively smaller fanout near the input of the chain compensates for the higher $\sigma(I_{\mathrm{d}})$ due to relatively smaller devices there, resulting in comparable variances for all gate stages. This explains the accuracy of the lower bound estimate.

Table 4.1: Estimates of $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ on different combinational digital circuits. Delay values are in FO4.

Name	detail	#gates	#paths	$Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ Monte Carlo	T_{nom}	lower bound	upper bound
inv5	5 inv. chain	5	1	7.5	7.4	7.5	7.8
inv52	2 chains of 5,2 inv.	7	2	4.6	4.3	4.5	4.9
dec8-256	8b decoder	680	2560	17.3	15.4	16	20.5
sh32	32b 5 stage shift	206	3008	18.5	16	17	23.4
add32KS	Kogge Stone adder	709	7260	15.5	14.2	14.3	23.7
add32BK	Brenk Kung adder	431	6216	15.1	13.1	13.4	22.4
add32Sl	Sklansky adder	475	6428	15.8	13.6	13.8	22.1
c1355	32b SEC	558	303000	22.8	20.2	20.8	37.3
c1908	16b SEC/DED	430	839000	30.6	26.1	26.8	55
c2670	12b ALU/control	963	17500	21.7	17.5	17.8	32.3
c3540	8b ALU w/ bcd & shift	961	3655000	33.3	27.8	28.3	58.1
c432	27channel interrupt ctrl	165	25000	21.5	18.4	18.7	33.8
c499	32b SEC	518	12300	18	15.9	16.5	28
c5315	9b ALU	1626	291000	31.1	26.8	27.2	51.3
c7552	32b add/cmp	1993	239000	32.3	30.6	31	61.6
c880	8b ALU	389	9000	21.3	18.1	18.5	34.4

Equipped with tools for analyzing the statistical behavior of a netlist, in the next section

[5]The worst case occurs when one gate stage significantly dominates the delay variation of a statistically critical path with many stages.

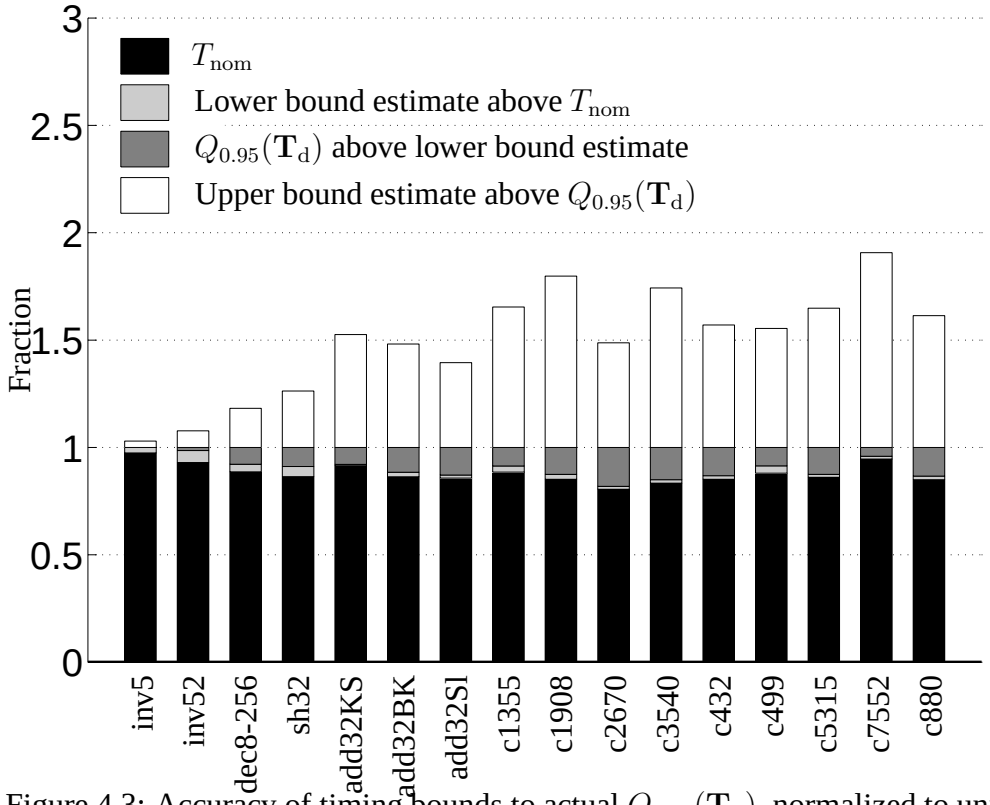


Figure 4.3: Accuracy of timing bounds to actual $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$, normalized to unity

we will see how optimization ignoring uncertainty is inadequate in producing good designs after fabrication.

4.2 Effect of ignoring process variations

The optimization in previous chapter assumed that there are no local process variations. Figure 4.4 shows the result of doing Monte Carlo simulations for delay of a 32-bit static Sklansky adder assuming the same variation model as in the previous section. If all the gate delay variations were fully correlated, all devices would change in the same way, causing the PDF to be symmetrical around the nominal delay T_{nom}. However, with independent

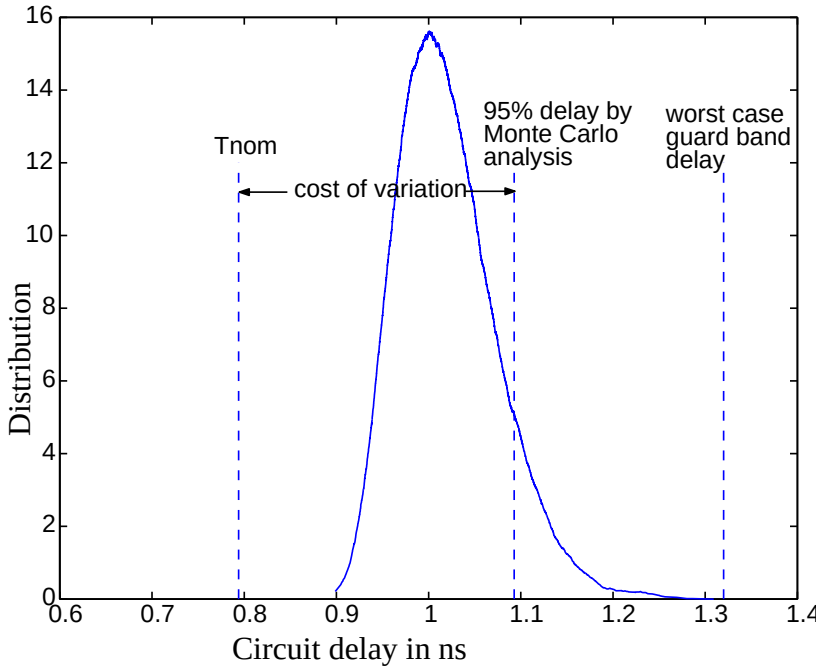


Figure 4.4: Monte Carlo analysis on a deterministically sized 32-bit adder

gate delays, the figure shows that interestingly, no part of the delay PDF even touches the nominal value. If we want 95% percent of the chips to meet the cycle time, then the cycle time specification has to be pushed out by almost 30%. This pushing out of delay PDF with variations can be explained by observing the mean-standard deviation ($\mu - \sigma$) scatter plot of the adder as shown in Figure 4.5. In this plot, the X axis represents the mean and the Y axis represents the standard deviation (σ) of the path delays. Each dot in the graph represents one path. Since the optimizer is unaware about variations, it has no information about the Y axis. From its point of view, as long as the delay is less than or equal to the max delay, the overall cycle time will not get worse. This assumptions leads to two problems. The first is that there is a wall of equally critical paths on the right edge of the graph. Since the cycle time is the max of these path delays, with variations, every additional critical path does hurt, since it means that it is more likely that one of those paths will be slower than

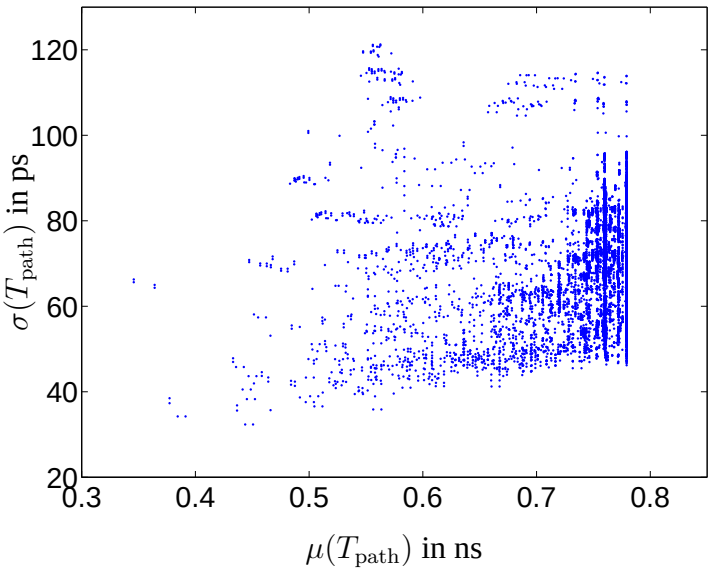


Figure 4.5: $\mu - \sigma$ scatter plot for deterministic sizing

nominal.

The second problem is that the sizer may downsize gates in critical paths, unaware of the fact that smaller gates have larger delay variations. It does not take steps to reduce the uncertainty of the path delay even in cases where it might take only a slight sizing perturbation to reduce the variation of a path.

These effects combine to make the design sensitive to variations, which is exactly what we are trying to avoid. This problem with conventional sizing programs is the one that we will address in the rest of the chapter.

4.3 Exact statistical sizing

In order to get a yield of α at our specified design point, the correct objective for delay of the design would be $Q_\alpha(\mathbf{T}_\mathrm{d})$. It is possible to express it analytically and hence solve

the statistical sizing problem exactly for some simple circuits. Figure 4.6 shows the inv52 circuit with two inverters chains C1 and C2, connected to the same input and having two outputs with timings $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively. Assuming that the input arrives at time $t = 0$,

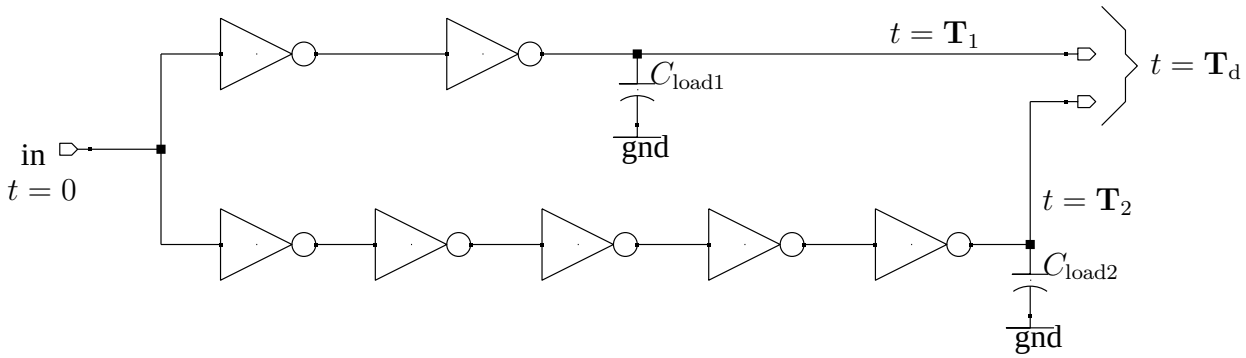


Figure 4.6: Example circuit for exact solution of the statistical design problem

the overall timing is given by $\mathbf{T}_\mathrm{d}$. With deterministic sizing, we would have $\mathbf{T}_1 = \mathbf{T}_2 = T_{\mathrm{nom}}$ and C1 having fewer stages will be sized smaller with more delay per stage. Now assuming Gaussian delays with Pelgrom's model for variations, C1 has a larger variance due to smaller devices and less averaging. This will result in a long tail in the delay PDF of the fabricated inv52 designs, which is what we want to tighten.

With Gaussian gate delays, the path delays are also Gaussian. Assuming that $\mathbf{T}_1 \sim N(\mu_1, \sigma_1)$ and $\mathbf{T}_2 \sim N(\mu_2, \sigma_2)$, $\mathbf{T}_\mathrm{d}$ will have a distribution given by the $\max()$ of two Gaussian random variables. Suppose that our goal is to size the circuit to minimize the 95^{th} percentile delay $Q_{0.95}(\mathbf{T}_\mathrm{d})$. This will ensure that 95% of the fabricated designs will make the delay we specify. We know that $P(\mathbf{T}_\mathrm{d} \leq Q_{0.95}(\mathbf{T}_\mathrm{d})) = 0.95$. Let $P(\mathbf{T}_1 \leq Q_{0.95}(\mathbf{T}_\mathrm{d})) = p_1$ and $P(\mathbf{T}_2 \leq Q_{0.95}(\mathbf{T}_\mathrm{d})) = p_2$, where p_1 and p_2 represent the percentiles

for individual paths. As the two paths are structurally independent,

$$\begin{aligned} P(\mathbf{T}_{\mathrm{d}} \leq Q_{0.95}(\mathbf{T}_{\mathrm{d}})) &= P(\mathbf{T}_1 \leq Q_{0.95}(\mathbf{T}_{\mathrm{d}}) \cap \mathbf{T}_2 \leq Q_{95}(\mathbf{T}_{\mathrm{d}})) \\ &= P(\mathbf{T}_1 \leq Q_{0.95}(\mathbf{T}_{\mathrm{d}})) \cdot P(\mathbf{T}_2 \leq Q_{0.95}(\mathbf{T}_{\mathrm{d}})) \\ &= p_1 \cdot p_2 = 0.95. \end{aligned} \tag{4.10}$$

This forms our first constraint. To meet the $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ spec, the distributions of $\mathbf{T}_1$ and $\mathbf{T}_2$ have to be tighter than $\mathbf{T}_{\mathrm{d}}$ so that we have the following constraints

$$\begin{aligned} Q_{p_1}(\mathbf{T}_1) &\leq Q_{0.95}(\mathbf{T}_{\mathrm{d}}), \\ Q_{p_2}(\mathbf{T}_2) &\leq Q_{0.95}(\mathbf{T}_{\mathrm{d}}). \end{aligned} \tag{4.11}$$

As $\mathbf{T}_1$ and $\mathbf{T}_2$ are Gaussian, $Q_{p1}(\mathbf{T}_1)$ and $Q_{p2}(\mathbf{T}_2)$ can be obtained by adding a suitable number of σ_1, σ_2 to μ_1, μ_2 respectively. This suitable number of σ deviations can be obtained in terms of p_1 and p_2 using the inverse CDF function (Eq. 4.5). For Gaussian distribution, σ is a log-log convex function of p for $p > 0.71$. Figure 4.7 shows modeling of inverse CDF with max of monomials $f(p)$, around our desired point of 0.95. Thus $f(p)$ represents the number of σ deviations for a particular probability. Now we can write the percentile delay as

$$Q_{p_1}(\mathbf{T}_i) = \mu_i + f(p_i)\sigma_i.$$

Assuming that μ_1,σ_1,μ_2 and σ_2 are posynomial function of the device sizes, the GP can

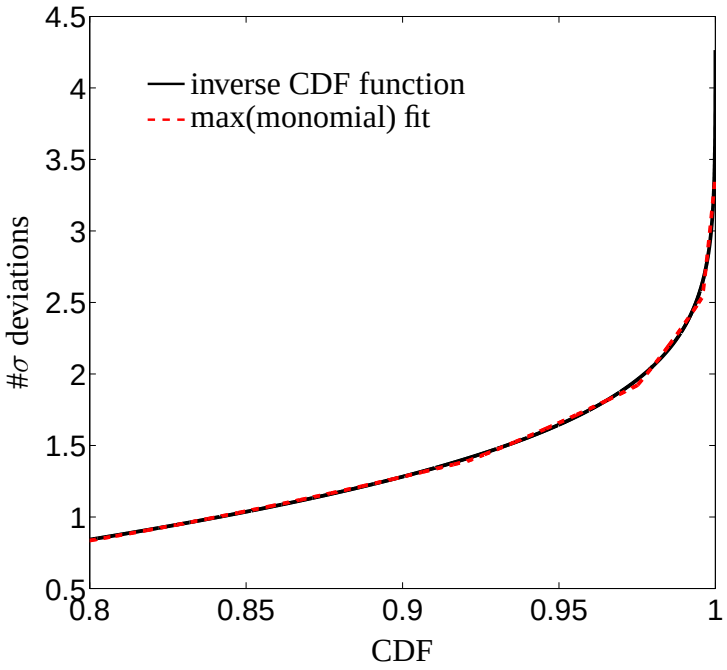


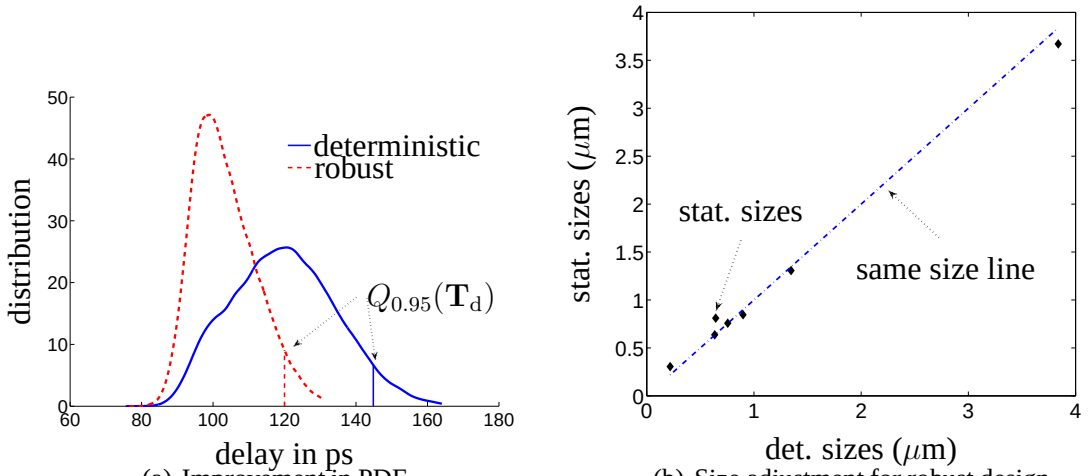
Figure 4.7: Modeling inverse CDF with max of 4 monomial terms

now be written as

$$\begin{aligned} \text{minimize} \quad & Q_{0.95}(\mathbf{T}_\mathrm{d}) \\ \text{subject to} \quad & p_1 \cdot p_2 = 0.95, \\ & \mu_1 + f(p_1)\sigma_1 \leq Q_{0.95}(\mathbf{T}_\mathrm{d}), \\ & \mu_2 + f(p_2)\sigma_2 \leq Q_{0.95}(\mathbf{T}_\mathrm{d}), \\ & \text{Area/Energy constraints}, \\ & \text{Other constraints}. \end{aligned} \tag{4.12}$$

Solving this sizing problem results in tightening of the inequalities in Equation 4.12. The two paths C1 and C2 get optimized to include their vulnerability to variations. Including the standard deviation as in the Pelgrom's model makes the optimizer size C1 bigger than it would have been in deterministic sizing as shown in Figure 4.8(b). Some area from the

bigger chain C2 is given to the smaller chain. The push back on $\mu(\mathbf{T}_1)$ and $\sigma(\mathbf{T}_1)$ more than compensates for the slight increase in $\mu(\mathbf{T}_2)$ and $\sigma(\mathbf{T}_2)$ to give a better $Q_{0.95}(\mathbf{T}_\mathrm{d})$. Figure



(a) Improvement in PDF (b) Size adjustment for robust design

Figure 4.8: Comparison of robust design to nominal design

4.8(a) shows the improvement in the PDF of $\mathbf{T}_\mathrm{d}$ after statistical design. While the circuit is now sized sub-optimally from a deterministic point of view, in presence of variations, the statistics of the overall delay are dominated only by one of the paths. In circuits with many outputs, more paths would be pushed back from being critical.

4.3.1 Balance of sensitivities

In the previous section, improvement in the mean delay of the smaller chain makes up for its relatively worse standard deviation to result in overall $Q_{0.95}(\mathbf{T}_\mathrm{d})$ improvement. While there are many combinations of means and standard deviations that can give the same $Q_{0.95}(\mathbf{T}_\mathrm{d})$, under area constraints, the optimal choice is the one where the ratio of area sensitivity to sensitivity of the delay metric (in this case, $Q_{0.95}(\mathbf{T}_\mathrm{d})$), is the same for all the variables. We

can re-arrange the condition of optimality as

$$\frac{\frac{\partial Q_{0.95}(\mathbf{T}_\mathrm{d})}{\partial \mu_1}}{\frac{\partial A}{\partial \mu_1}} = \frac{\frac{\partial Q_{0.95}(\mathbf{T}_\mathrm{d})}{\partial \mu_2}}{\frac{\partial A}{\partial \mu_2}} \Rightarrow \frac{\frac{\partial Q_{0.95}(\mathbf{T}_\mathrm{d})}{\partial \mu_1}}{\frac{\partial Q_{0.95}(\mathbf{T}_\mathrm{d})}{\partial \mu_2}} = \frac{\frac{\partial A}{\partial \mu_1}}{\frac{\partial A}{\partial \mu_2}} \tag{4.13}$$

where A is the total area of the circuit. We cannot write this condition for T_nom in nominal design, as T_nom is not differentiable in $\mu_{1,2}$[6]. Now, for $Q_{0.95}(\mathbf{T}_\mathrm{d})$, the sensitivity ratio changes with different combinations of means and standard deviations, but the equality always remains[7]. To show this, let us consider a simpler version of the above circuit in Figure 4.9. Assume that the loads on the two chains are unequal with some ratio $r, r > 1$

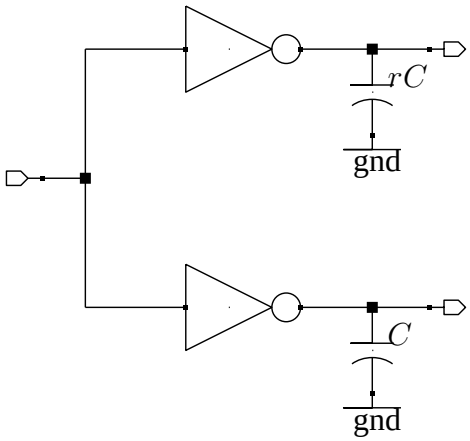


Figure 4.9: A simple circuit with symmetric structure but different loads

and for the gate delays $\sigma \propto \mu$[8]. We will change r to see how the optimal area-delay point moves with changing area distribution. Starting with nominal design we have $\mu_1 = \mu_2$ and so it follows that $\sigma_1 = \sigma_2$. The load on path 1 is the larger of the two. Therefore, we can redistribute the sizing between the two inverters such that μ_1 increases while μ_2 decreases

[6]This ratio is unity for increasing $\mu_{1,2}$, because if any path becomes slower, the delay is given by that path. The ratio is undetermined for decreasing $\mu_{1,2}$ because if any path becomes faster, then it no longer determines the circuit delay and its sensitivity becomes zero.

[7]assuming none of the variables hit their range limits

[8]This is done to make the problem simpler by eliminating two variables

keeping the $Q_{0.95}(\mathbf{T}_\mathrm{d})$ constant. Figure 4.10 shows the curve of constant $Q_{0.95}(\mathbf{T}_\mathrm{d})$ with changing means obtained by taking many Monte-Carlo samples in MATLAB. Clearly, increase in μ_1 causes greater and greater decrease in μ_2 until the $Q_{0.95}$ of the first path delay itself equals $Q_{0.95}(\mathbf{T}_\mathrm{d})$ at which point the curve becomes vertical.

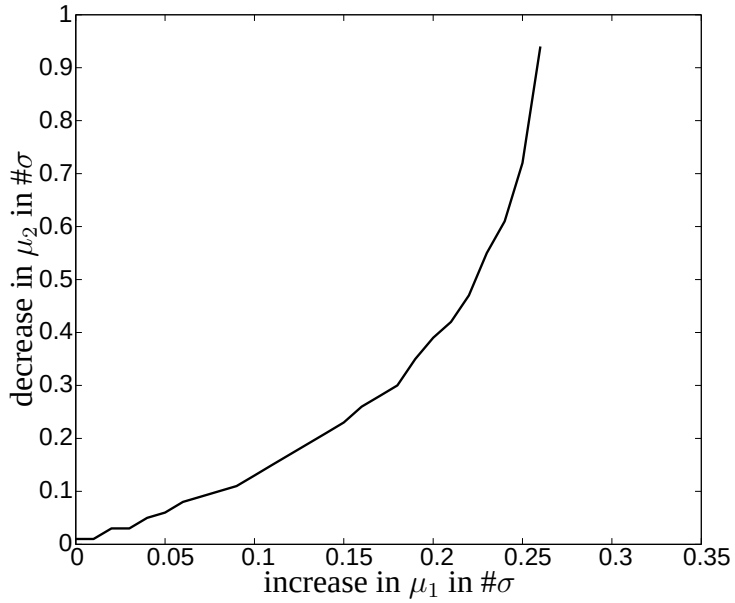


Figure 4.10: Tradeoff between mean delays for the same $Q_{0.95}(\mathbf{T}_\mathrm{d})$

While delay is inversely related to the size of the gates, area is directly proportional to it. So with only sizing as our variable, we can write the total circuit area as

$$A = K(r/\mu_1 + 1/\mu_2),$$

where K is some constant. Let us evaluate the area at every point on the constant $Q_{95}(\mathbf{T}_\mathrm{d})$ curve and take the minimum, for different values of load ratio r. Figure 4.11 shows how the r is same as the slope of the curve in Figure 4.10 at different points. If there are more

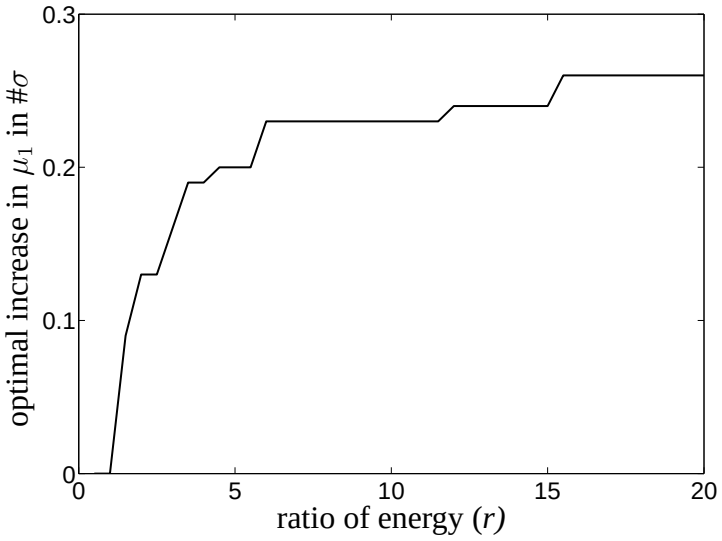


Figure 4.11: Change in the optimal mean with the ratio of energy between the two chains

such parallel paths, the ratio of the sensitivities to area and $Q_\alpha(\mathbf{T}_\mathrm{d})$ for all of them would be equal. As a result some paths would be pushed back from being critical.

From this principle, we can infer that if all paths in a circuit are structurally identical and drive the same load, then the optimal robust sizing will be no different from the deterministic one, because given a fixed overall area every path will have the same sensitivity to the overall $Q_{0.95}(\mathbf{T}_\mathrm{d})$, the same area cost, and therefore an equal share of the area as any other path. Therefore one path cannot be pushed back from the other critical one. Assuming that variations are size dependent, the only improvement can come from re-sizing devices within a path to make the path delay distribution tighter.

4.3.2 Non-scalability of the exact solution

While the exact solution easily solves the robust design problem for inv52, it is not suitable for typical circuits. For instance, for the circuit shown in Figure 4.12, which is a minor

modification on inv52 circuit in Figure 4.6, $\mathbf{T}_1$ and $\mathbf{T}_2$ are not independent and so the 95^{th} percentile boundary of their joint distribution cannot be given by the simple product of

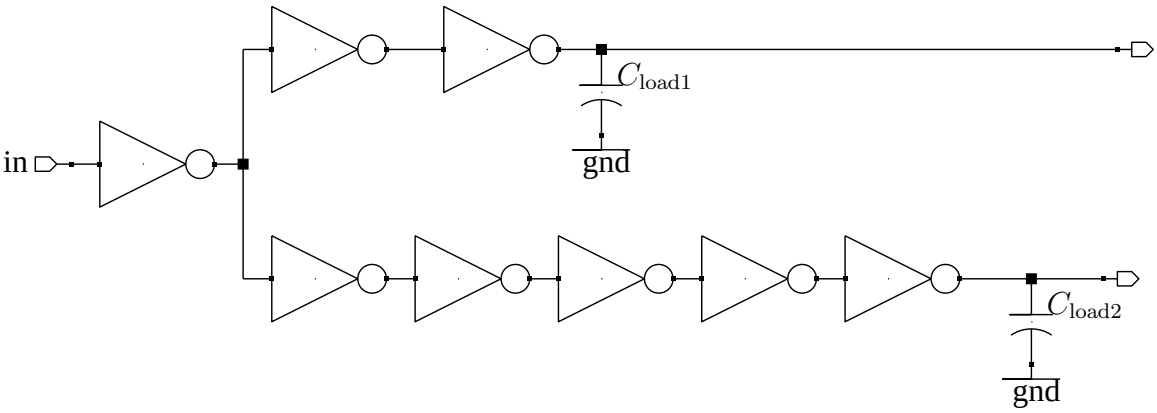
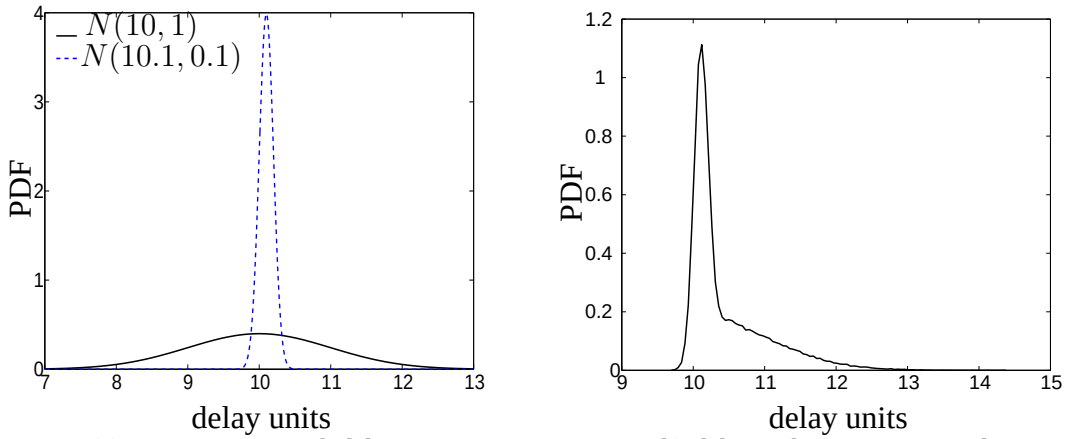


Figure 4.12: A simple circuit with two dependent paths

probabilities p_1 and p_2. A more complicated constraint describing an ellipsoidal region is needed. The situation gets even more complicated as the number of paths and the interconnections between them increase. We can see from Table 4.1 that the number of paths in circuit netlists grows exponentially with the size, making it computationally impractical to enumerate all the paths for robust sizing.

Another method to avoid enumerating all paths is to propagate delay PDFs throughout the netlist in the same fashion as the static timing formulation in deterministic design. However, as mentioned before, no practical family of distributions are closed under sum and $\max$ operations. Figure 4.13 shows an example of a nand gate with inputs having gaussian distributions with comparable means but very different variance. The PDF of the output timing shows a long tail which is hard to define with a set of few parameters, especially as we care about the 95^{th}, a point that lies on this tail.

A simpler solution is therefore needed to capture variations during optimization. In the following sections we describe our heuristics that get us most of the benefit of robust sizing



(a) Two Gaussian path delays (b) delay at the converging node

Figure 4.13: Delay PDFs with comparable means but different variances produce long tails at converging nodes

with only modest computational cost over deterministic sizing.

4.3.3 Previous work in robust sizing

The non-practicality of the exact solution for typical circuits led to a lot of research in the area of sizing for robustness under variations [38, 73, 98, 31]. One idea involved augmenting the objective function in the deterministic sizing problem to include a penalty for outputs coming closer to the critical output [98], thereby reducing the number of critical paths. The penalty coefficient had to be carefully chosen to prevent the optimizer from choosing incorrect sizing to reduce the objective function by focusing on the penalty instead of the overall delay. Another path based approach [73] identifies the non-critical paths responsible for the delay spread by defining a *disutility* function for gate and path delays that includes both means and variances of the delay random variables. Sizing is done to minimize the *disutility* of non-critical paths responsible for timing yield loss. As the algorithm has to enumerate paths, its complexity, and hence the solution time grows

quickly with the circuit size. A simulation based approach using static delay formulation is presented in [38] where approximate formulae are used to calculate the mean and variance of the max of Gaussian distributions at every iteration. These formulae are complex to solve and not amenable to analytical expressions. Besides, being non-convex, the convergence is slow and approximating a distribution with a long tail as Gaussian is error-prone as shown in Figure 4.13. Other algorithms [31] improve the delay statistics by adding area to selectively upsize smaller devices. While this improves the delay statistics, increasing area (energy) usually improves performance. The key is to improve performance with the same area or energy constraint as that for the deterministic design.

4.3.4 Basic idea for robust sizing

Sizing problems generally have a broad "flat" region around the timing minima. This means that in most cases small perturbations have small effects on the nominal circuit delay. In the deterministic method, the sizer puts most of the effort in getting to that lowest point in a relatively flat region. But this design point might be very sensitive to variations because at this point many paths are critical. In addition, many of these paths may be critical because they contain small devices, which can increase the delay uncertainty. The idea is to get robustness using these small sizing perturbations.

We can use two main insights from the exact statistical solution to formulate efficient heuristics. Given a deterministically sized inv52 circuit, we saw in Section 4.3 how robust sizing pushes back on one of the paths. This is the first thing we would like to achieve [98]. Secondly, if the uncertainty depends on the device size, then in addition to the pushing back on paths, we need to adjust the device sizes within a to reduce the uncertainty. By having a penalty for uncertainty that prevents the sizer from making bad choices about path delays and device sizes, we might be able to achieve much of the benefit of considering the

complete distribution of the exact solution [69].

4.4 Heuristic techniques for robust sizing

Given that propagating accurate PDFs is impractical and modeling them as Gaussian causes loss of information in the PDF tail, the key question is how to propagate $Q_{0.95}$ of net timings. In addition, as many efficient means of solving the deterministic sizing problem have been developed, is there a way to make the robust sizing look like the deterministic sizing so that we can leverage the already existing solutions? To answer these questions, we seek to modify the deterministic optimization problem to make it more variation aware, while maintaining the GP framework that allows efficient solutions. We propose the following two techniques based on the performance bounds estimation methods using surrogate netlists and the sensitivity analysis described in Sections 4.1.2 and 4.3.1 respectively.

4.4.1 Adding delay margins (ADM)

We propose to augment the gate delays d_{i-o} in (3.1) to obtain $\tilde{d}_{i-o}$ defined as

$$\tilde{d}_{i-o} = \mu(d_{i-o}) + \kappa_j \sigma(d_{i-o}). \tag{4.14}$$

The margin coefficients κ_j introduced in Section 4.1.2 help to account for process variations by adding to every gate j, a delay penalty term that is proportional to its delay uncertainty.

Margin coefficients provide the tradeoff between the mean and variance at the gate level. The choice of κ depends on what percentile of the distribution one is optimizing for, but only weakly. This added margin is just a hint to the optimizer to try to minimize uncertainty along critical paths. It is not a precise method. Section 4.1.2 described different

choices of κ_j that can be used for margining. Even if the $Q_{0.95}(\mathbf{T}_\mathrm{d})$ estimate at some given κ is not good, a scaled value might help. In general it is hard to predict what value will produce a result that is closest to the nominal delay (and hence to the optimal result as nominal delay is a lower bound on the $Q_{0.95}(\mathbf{T}_\mathrm{d})$). In fact, in our solution we try different values of κ_j and choose the value that yields the best result from SSTA using Monte Carlo simulations. Useful values of κ range from 1 to about 4. The region around minimum $Q_{0.95}(\mathbf{T}_\mathrm{d})$ is flat so that with a granularity of 0.5, we can easily cover the entire range with 7 optimizations and SSTAs. The results of using this technique are presented in Section 4.5.

4.4.2 Using soft-max (USM) for merging path delays

Since T_out in Equation 3.1 is a maximum of a set of input delays that are random, the distribution of T_{out} is shifted to the right of all the input delay distributions. This shift is more pronounced when several of the input arrival time distributions are close to the dominant one, and negligible when, say, only one of the inputs dominate the distribution. To take into account the right shift caused by taking the maximum of a set of random variables, we propose to use a *soft maximum* function smax_p defined as

$$\mathrm{smax}_p(x) = \left(\sum |x_i|^p\right)^{1/p},$$

where p is the exponent that represents the penalty for closeness of arguments and the sum accounts for increase in uncertainty with every extra input. USM steers the optimizer away from sizing many paths to be critical, a bad situation for delay statistics. While it approaches the $\max$ function asymptotically, the soft max retains in spirit the fact that under variations even a path with smaller nominal delay can contribute to the delay spread

at the converging node. To simulate the propagation of $Q_{0.95}$ of signal arrival times in the netlist, USM is applied in conjunction with ADM.

USM is not only an enhancement to ADM, in some cases where ADM is ineffective, it is necessary for robust sizing. For example, in the circuit presented in Section 4.3.1, with $\sigma(\mathbf{D}) \propto \mu(\mathbf{D})$, ADM by itself just results in delay scaling without any sizing improvement over the deterministic case. USM forces the paths to be unequal by modeling their pushing effect on the overall delay PDF and leads to a better (not necessary optimal) distribution of area between the two inverters.

Combining ADM and USM, we can write the equation for T_{out} for the gate in Figure 3.2 by modifying Equation 3.1 as

$$T_{\text{out}} = \left(\sum_{i=1,2,3} |T_i + D_{i-o}|^p \right)^{1/p}.$$

Using this relation as the delay propagation equation retains the computational merits of the deterministic sizing problem (like sparsity), making the algorithm scalable to larger circuits. Moreover, if the $\mu(d_{i-o})$ and $\sigma(d_{i-o})$ of gate delays are generalized posynomials then the problem can still be cast as a Generalized Geometric Program (GGP) [10], leading to efficient solutions. A crude search loop in the (p, κ) space around the basic optimization routine can easily be implemented to obtain the best statistical sizing (as validated by SSTA). In our experience the best p varies over a range of 30 for different circuits depending on their topology. Within this range, the sensitivity of $Q_{0.95}(\mathbf{T}_{\text{d}})$ is small so that a granularity of 10 suffices.

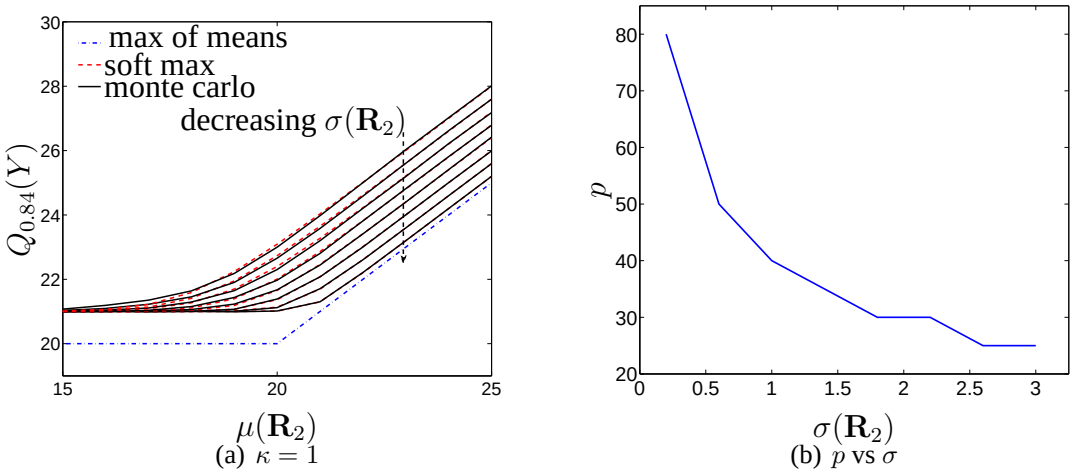
4.4.3 Validating USM and ADM with two Gaussian delay variables

To show how ADM and USM together simulate the statistical delay propagation, let us perform a simple MATLAB experiment. Consider two Gaussian random delay variables $\mathbf{R}_1$ and $\mathbf{R}_2$. To keep the probability of negative samples negligible, let $\mu(\mathbf{R}_1) = 20$ and $\sigma(\mathbf{R}_1) = 1$ while we sweep $\mu(\mathbf{R}_2)$ from 15 to 25 and $\sigma(\mathbf{R}_2)$ from 0.2 to 3. Let $\mathbf{Y} = \max(\mathbf{R}_1, \mathbf{R}_2)$. We seek to model $Q_{0.84}(\mathbf{Y})$ (*i.e.*the percentile corresponding to 1σ increase over the mean) and $Q_{0.95}(\mathbf{Y})$ of the random variable $\mathbf{Y}$. We first find these values using Monte Carlo simulation. These are plotted as solid curves of varying $\mu(\mathbf{R}_2)$ for changing values of $\sigma(\mathbf{R}_2)$ in Figures 4.14(a) and 4.15(a). We then define $\widehat{Q}_{0.84}(\mathbf{Y})$ and $\widehat{Q}_{0.95}(\mathbf{Y})$ as:

$$\begin{aligned} \widehat{Q}_{0.84}(Y) &= \mathrm{smax}_{p1}(\mu(\mathbf{R}_i) + 1\sigma(\mathbf{R}_i)), \quad i = 1, 2 \\ \widehat{Q}_{0.95}(Y) &= \mathrm{smax}_{p2}(\mu(\mathbf{R}_i) + 1.65\sigma(\mathbf{R}_i)), \quad i = 1, 2 \end{aligned} \tag{4.15}$$

and attempt to fit $Q_{0.84}(\mathbf{Y})$ and $Q_{0.95}(\mathbf{Y})$ by choosing the right p. The model fit is shown in dashed curves. The values of κ must be equal to $\Phi^{-1}(0.84) = 1$ and $\Phi^{-1}(0.95) = 1.65$ in order to fit the asymptotic regions where one distribution completely dominates the other. The values of p ($p1$ and $p2$ in Equation 4.15) change to represent how smoothly the percentile lines curve, as shown in Figures 4.14(b) and 4.15(b). The plots in Figures 4.14 and 4.15 show that our smax_p and $\kappa\sigma$ margins give close estimates of the $Q_{0.84}(\mathbf{Y})$ and $Q_{0.95}(\mathbf{Y})$ for specific values of p and κ. Here $\mathbf{R}_1$ and $\mathbf{R}_2$ represent the delay of two converging paths which can vary over a significant range from each other in their mean and standard deviation. The value of p decreases as the extent of variations increases, reflecting a higher penalty for paths with high variance coming closer.

For simplicity, we use uniform p and κ for all gates in the circuit. As mentioned before, in practice, a range of 30 suffices for p. Values of p exceeding the range are associated with relatively small variations that do not dominate the overall delay. This range is chosen from



(a) $\kappa = 1$ (b) p vs σ

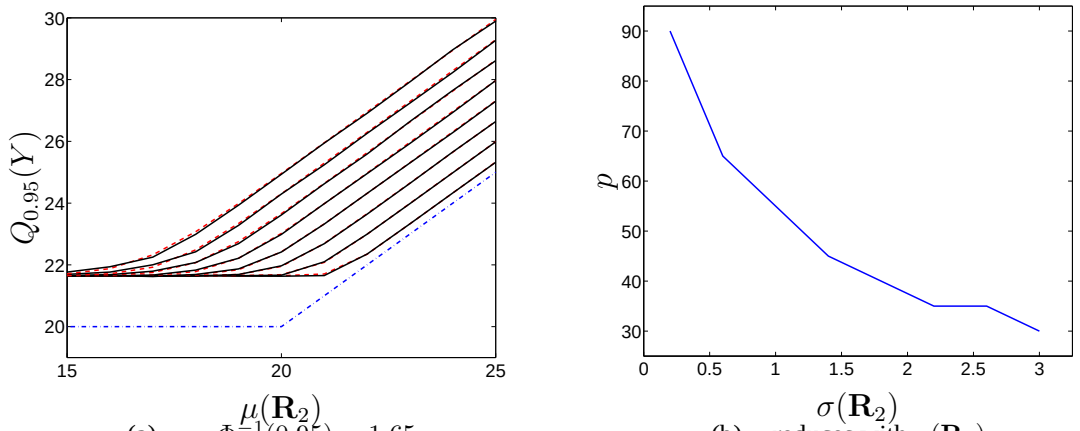
Figure 4.14: Validating USM and ADM for the 84^{th} percentile point

Figures 4.14(b) and 4.15(b) based on the extent of variations specified for the technology. For the extent of variations we chose, $p \in [20, 50]$ and $\kappa \in [1, 4]$ give good statistical sizing for $Q_{0.95}(\mathbf{T}_{\text{d}})$ in the representative circuits.

4.5 Applying robust sizing heuristics

The robust design algorithm simply consists of using the USM and ADM for different values of κ and p, and choosing the best design. The number obtained for the signal arrival time T at any net using the heuristics is certainly not the exact $\mathbf{Q}_{\alpha}(T)$ (for the specified α) of its timing distribution. It just represents a rough measure of the criticality of the arrival time to the overall delay. The timing results we present are always from SSTA[9] done after the robust optimization. SSTA is the only trustworthy method for comparing results. The soft max function, and the simple augmented delay expression are used only to *design* the circuit, and not to *analyze* it.

[9]This consists of performing STA on 10000 Monte Carlo netlist samples, leaving a residual error of $1/\sqrt{(10000)} = 1\%$ in our $Q_{0.95}(\mathbf{T}_{\text{d}})$ estimation.



(a) $\kappa = \Phi^{-1}(0.95) = 1.65$ (b) p reduces with $\sigma(\mathbf{R}_2)$

Figure 4.15: Validating USM and ADM for the 95^{th} percentile point

For testing the efficacy of our heuristics, we assume a general dependence of delay variation on transistor size given as

$$\frac{\sigma(\mathbf{D})}{\mu(\mathbf{D})} \propto \frac{1}{(LW)^{\alpha}},\ \alpha \geq 0, \tag{4.16}$$

where α is the degree of dependence on size. For Pelgrom's model, $\alpha = 0.5$. Here we show the results for $\alpha = 0.5$ and $\alpha = 0$. The variation in both cases is normalized to have 15% of relative standard deviation in the drive current ($\sigma(I_{\text{d}})/I_{\text{d}}$) for 1μ minimum length device.

Figure 4.16 shows the delay distribution improvement for 32-bit adder after using the ADM and USM techniques assuming Pelgrom's variation model. The $Q_{0.95}(\mathbf{T}_{\text{d}})$ improves by more than 20% over the deterministic design. The reason for this improvement becomes clear by observing the μ-σ scatter plot for this design shown in Figure 4.17. The high variance critical paths resulting from deterministic sizing are pushed back from dominating the overall delay, at the cost of modest increase in the delay (and variance) of the low variance paths, resulting in overall reduction of $Q_{0.95}(\mathbf{T}_{\text{d}})$. The total area is redistributed

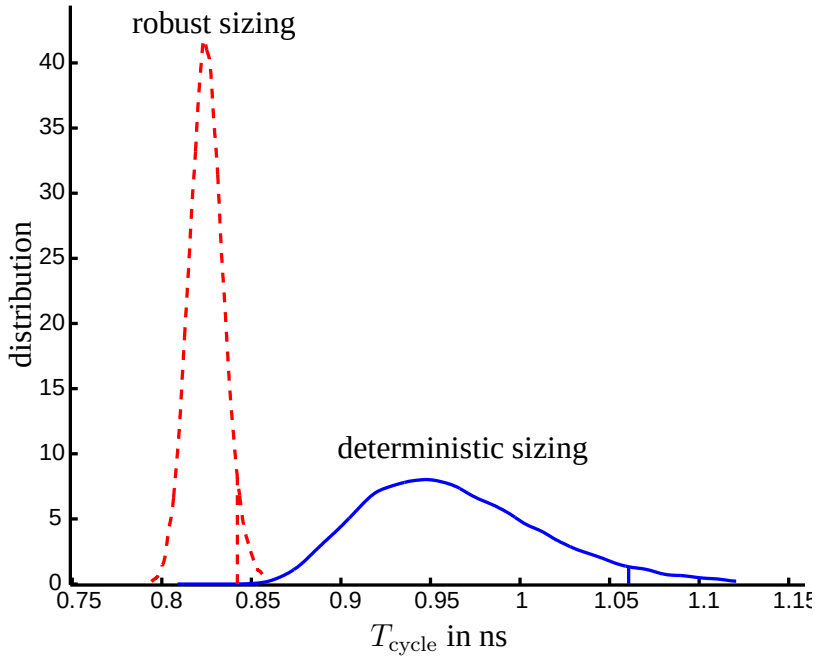


Figure 4.16: Improvement in $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ for a 32-bit LF adder using $\kappa = 1.5$ and $p = 30$

to upsize the high variance paths. The heuristics seek to achieve the optimality condition described in Section 4.3.1 by balancing the sensitivity of $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ to different path delays with their marginal cost of area. The delay of the circuit is now dominated by a fewer critical paths.

Table 4.2 shows the results of using our heuristics on the benchmark circuits introduced in Section 4.1.2. To get an upper bound on the sub-optimality of our solution, we use T_{nom} as our lower bound estimate on $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$. This upper bound is shown in the last column. Note that we cannot use the tighter lower bound described in Section 4.1.2 for sub-optimality calculation as we do not have the optimal sizing to begin with. Results shows that using ADM and USM heuristics combined give an improvement of anywhere from 0 to 24% of T_{nom} depending on the netlist topology. Due to inherent structural symmetry, the decoder dec8-256, 32 bit logarithmic shifter/rotator sh32 and single error correction (SEC)

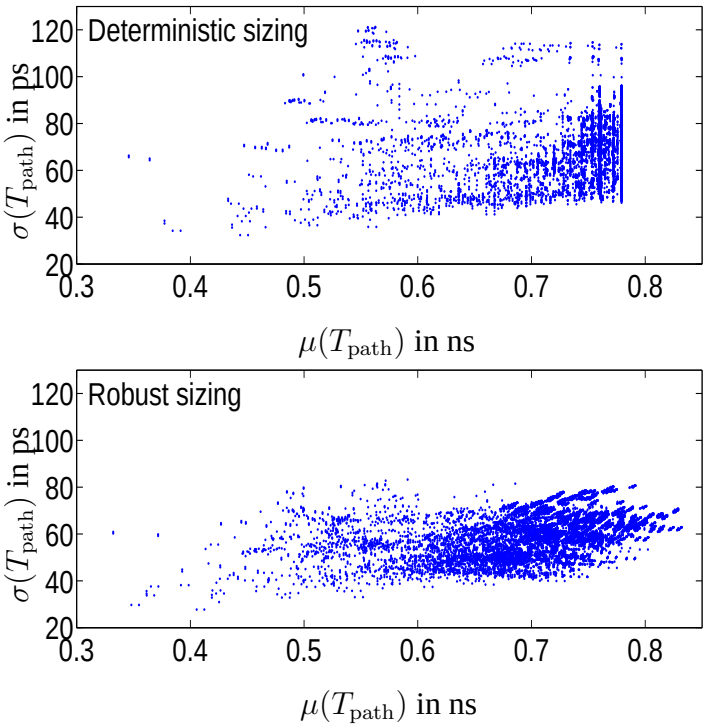


Figure 4.17: μ–σ scatter plot for all paths of 32bit LF adder ($T_{\rm path}$ = path delay)

circuits c1355 and c499 have many of their $Q_{0.95}(\mathbf{T}_{\rm d})$ equally sensitive to many of their paths and consequently show very small statistical improvement over deterministic sizing. In particular, the improvement in decoders is strictly due to redistribution of width within a single path (like the inv5 circuit). Naturally, lesser improvement over deterministic sizing means higher sub-optimal bound. Even circuits with identical logical function (add32KS, add32BK and add32Sk) can have different improvements based on their topologies. While add32KS has many paths of similar length, add32BK has a single long path with many small side paths. In deterministic sizing, the optimizer downsizes many of these side paths making them critical. These small paths suffer from large variability. Consequently, under identical design constraints, $Q_{0.95}(\mathbf{T}_{\rm d})$ degrades the most for add32BK under variations.

Table 4.2: SSTA of robust sizing using Pelgrom's model. Delays are in FO4. The original $Q_{0.95}(\mathbf{T}_\mathrm{d})$ is for deterministic sizing.

circuits	orig. $Q_{0.95}(\mathbf{T}_\mathrm{d})$	T_nom	unif. κ UK	UK + $\mathrm{smax}()$	$\kappa \propto \frac{1}{\sqrt{l_i}}$ KSq	KSq + $\mathrm{smax}()$	imprvmnt in $Q_{0.95}(\mathbf{T}_\mathrm{d})$ in % T_nom	max subopt in % T_nom
inv5	8.3	7.4	7.5	7.5	7.5	7.5	10	3
inv52	5	4.3	4.6	4.6	4.7	4.6	9	8
dec8-256	17.3	15.3	17.3	17.3	17.3	17.3	0	13
sh32	19	16	18.4	18.3	18.4	18.3	4	14
add32KS	17.7	13.3	16.1	15.4	15.7	15.4	18	15
add32BK	17.2	12.9	14.5	14.2	14.2	14.2	24	10
add32Sl	16.7	13.1	15.1	14.9	15.1	14.9	14	14
c1355	22.9	20	22.4	22.1	22.2	22.1	4	10
c1908	30.7	25.8	30	28.3	28.9	28.3	9	9
c2670	22.8	17.6	20.3	18.6	19	18.6	24	6
c3540	34.1	27.7	32.2	30.1	30.3	30.1	15	9
c432	22.6	17.8	20.2	19	19	19	20	7
c499	18.1	15.8	17.9	17.6	17.8	17.6	3	11
c5315	32.9	26.7	29.3	28.4	28.4	28.3	17	6
c7552	35.8	30.5	33	32.2	32.4	32.2	12	5
c880	23.2	18.3	19.8	19.3	19.5	19.3	21	5

However, this also means that add32BK stands to gain the most from statistical sizing. The heuristics prevent the side paths from being downsized to criticality while maintaining a reasonable overall delay. As expected, add32KS gains the least because of having a more uniform topology. The topology of add32Sk falls in the middle of the two extremes. However, under identical design constraints, Sklansky topology still wins as it achieves the smallest T_nom that makes up for its poorer gains from statistical design due to its topology

Table 4.3 show similar results for the case where $\sigma(I_\mathrm{d}) \propto \mu(I_\mathrm{d})$. In this case, because the variations for two stages with equal relative loading is the same regardless of the driver size, having small devices is not bad. The improvement in $Q_{0.95}$ thus comes only from sizing the paths relative to one another, which is clear from getting no improvement in the

inv5 benchmark. Exact statistical sizing on inv5 and inv52 also shows negligible improvement in robustness. In fact, using uniform margin coefficients for ADM without USM is ineffective as it just scales the delay model and hence results in deterministic sizing[10]. Here we rely largely on USM to get us the robust design. The statistical behavior of ALU

Table 4.3: Results of robust sizing techniques in case where $\sigma \propto \mu$. Delays are in FO4.

circuits	det. $Q_{0.95}(\mathbf{T}_\mathrm{d})$	T_nom	unif. κ UK	UK + $\mathrm{smax}()$	$\kappa \propto 1/\sqrt{l_i}$ KSq	KSq + $\mathrm{smax}()$	imprvmnt in $Q_{0.95}(\mathbf{T}_\mathrm{d})$ in % T_nom	subopt u. bound in % T_nom
inv5	8.3	7.4	8.3	8.3	8.3	8.3	0	12
inv52	5	4.3	5	4.8	5	4.8	3	14
dec8-256	18.8	15.3	18.8	18.8	18.8	18.8	0	23
sh32	19.6	16	19.6	19.4	19.6	19.4	1	21
add32KS	16	13.3	16	15.5	16	15.5	4	16
add32BK	15.3	12.9	15.3	14.5	15.3	14.5	6	13
add32Sl	16	13.1	16	15.3	16	15.3	5	17
c1355	22.3	20	22.3	22	22.3	22	1	10
c1908	29	25.8	29	28.1	29	28.1	3	9
c2670	21	17.6	21	18.9	20.9	18.9	12	7
c3540	31.9	27.7	31.9	30.3	31.9	30.3	6	9
c432	20.8	17.8	20.8	19.5	20.8	19.5	7	10
c499	17.8	15.8	17.8	17.6	17.8	17.6	1	11
c5315	31.4	26.7	31.4	28.9	31.4	28.9	9	8
c7552	34.8	30.5	34.8	32.7	34.8	32.7	7	7
c880	20.7	18.3	20.7	19.8	20.6	19.8	5	8

and control logic benchmark c2670 is governed mainly by having many really short paths along with really long ones[11]. Therefore it shows marked improvements over deterministic design in both variation cases .

We can see from Eq. 4.16 that the absolute variation in delay, represented by $\sigma(\mathbf{D})$, increases with lowering the size of the effective driver transistor or increasing the delay of

[10] In fact, this is nothing but using some worst case corner, as defined by the κ value, which naturally does no change the sizing.

[11] Here short and long is based on the number of stages

that stage. Therefore a simple method to curb variations is to limit the minimum device size in deterministic sizing so that all devices are sized away the portion where $1/\sqrt{size}$ is high. The disadvantage of this simple method is that it uniformly constraints all devices without considering the sensitivity of the overall delay to their delay variation. An alternative simple method is to constraint the delay per stage in deterministic sizing. This would keep the non-critical but otherwise highly variation-prone paths well sized. Again, this method has the disadvantage of constraining all stages uniformly to over size even statistically non-dominant stages. The ADM and USM techniques effectively include both these techniques by upsizing and improving the delay only the statistically dominant stages as indicated by the sensitivity of $Q_{0.95}(\mathbf{T}_{\mathrm{d}})$ to the delay variation of different stages.

4.6 Including variations in energy

While accounting for delay variations is made complicated by the sum and max operations, the total energy is simply the sum of energy dissipated in every gate and interconnect. As the number of these elements is usually quite large, the PDF of the total energy dissipation is quite narrow for local independent variations and its mean is a good estimate for the energy of the circuit. Dynamic energy is the sum of energy dissipated in all the switching capacitors (Equation 2.3). Assuming 10% local temporal variations in V_{dd}, we use $\mu(V_{\mathrm{dd}}^2)$ in our expressions. Note that this value is bigger than $\mu^2(V_{\mathrm{dd}})$.

$$\mu(V_{\mathrm{dd}}^2) = \mu^2(V_{\mathrm{dd}}) + \sigma^2(V_{\mathrm{dd}})$$

for Gaussian distribution of V_{dd}. The local variations in individual C_is are averaged out by summing over a large number of gates.

Just like calculating the correct $\mu(V_{\mathrm{dd}}^2)$ for dynamic energy, care should be taken to

calculate the correct average leakage energy. Assuming that $V_{\rm th}$ in Equation 2.6 has a Gaussian distribution, I_j has a log-normal distribution and its average value is greater than that obtained by using $\mu(V_{\rm th})$ in (2.6) as shown below [43].

$$\mu(I_j) = I_0 \exp(-\frac{\mu(V_{{\rm th},j}) - \gamma_{\rm D} V_{\rm dd}}{nV_T} + \frac{\sigma^2(V_{{\rm th},j})}{2(nV_T)^2}) \tag{4.17}$$

In other words, the mean of leakage current is increased by a factor $F_{\rm leak}$ given as

$$F_{\rm leak} = \exp(\frac{\sigma^2(V_{\rm th})}{2(nV_T)^2}).$$

Clearly, $F_{\rm leak}$ increases exponentially with the variation in $V_{\rm th}$. The effect of $V_{\rm th}$ variation on the transistor leakage current is shown in Figure 4.18. The dependence of $\sigma^2(V_{\rm th})$ on sizing through Pelgrom's model causes $F_{\rm leak}$ to shoot up as the devices get smaller. The same plot also compares the average leakage current in the presence of variations (4.17) to the nominal value using $V_{\rm th}$ fixed to its mean, as a function of transistor width. The currents are normalized to the nominal leakage current of a 1μ wide transistor. Interestingly, due to variations, as the width reduces, the average leakage does not get smaller, in fact, it can actually increase! At larger widths $F_{\rm leak}$ approaches unity and hence the average leakage asymptotically reaches the nominal value. This can be viewed as lowering of the effective threshold voltage for leakage in presence of variations. The DIBL factor term can be adjusted similarly for local variations in $V_{\rm dd}$. $F_{\rm leak}$ can be as high as 6 or 7 for smaller devices. In a deterministic approach, the optimizer, unaware of this factor, sets many devices to have small sizes and low $V_{\rm th}$ to obtain a good drive with lower input capacitance. This is bad as the high $\sigma(V_{\rm th})$ for small devices exponentially increases the leakage current, contrary to the assumptions made by the deterministic optimizer.

With the inclusion of $V_{\rm th}$ and $V_{\rm dd}$ as design variables, we can express $\sigma(I_{\rm d})$ as a function

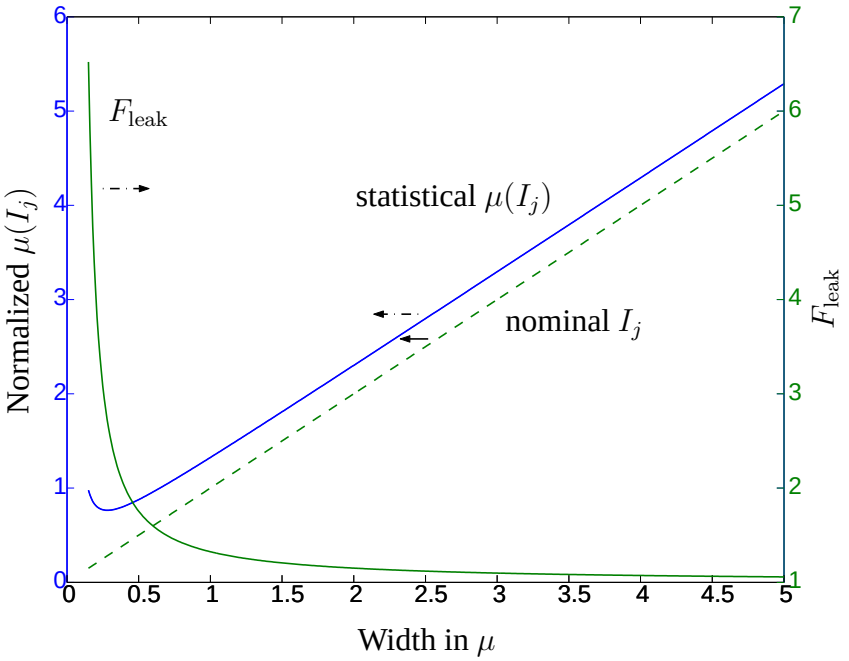


Figure 4.18: F_{leak} and normalized $\mu(I_j)$ (4.17) as a function of transistor width

of variations in V_{th} and the current factor β [70]. For simplicity we model $\sigma(I_{\text{d}})$ based on the alpha power law current model [76] as

$$\frac{\sigma^2(I_{\text{d}})}{I_{\text{d}}^2} = \frac{\sigma^2(V_{\text{th}})}{V_{\text{od}}^2} + \frac{\sigma^2(\beta)}{\beta^2}$$

where $\sigma^2(\beta)$ accounts for variation in β due to variations in channel length, mobility etc.. We assume that these these two sources of variations are independent. Using Pelgrom's model we can express the local variation in V_{th} for a stack of transistors as

$$\sigma^2(V_{\text{th}}) \propto \frac{1}{L_{\text{eff}} W_{\text{eff}}}.$$

The relative variation in β also has the same dependence. The quadratic model is made only to formulate $\sigma(I_{\text{d}})$ and produces a small error in calculating delay variation which is

calculated from Equation 2.7 as

$$\sigma(\tau_d) = \frac{d\tau_d}{dI_{\rm d}}\sigma(I_{\rm d}).$$

Using the above equations and $V_{\rm th}$, $V_{\rm dd}$ and sizing as design variables, we can obtain the

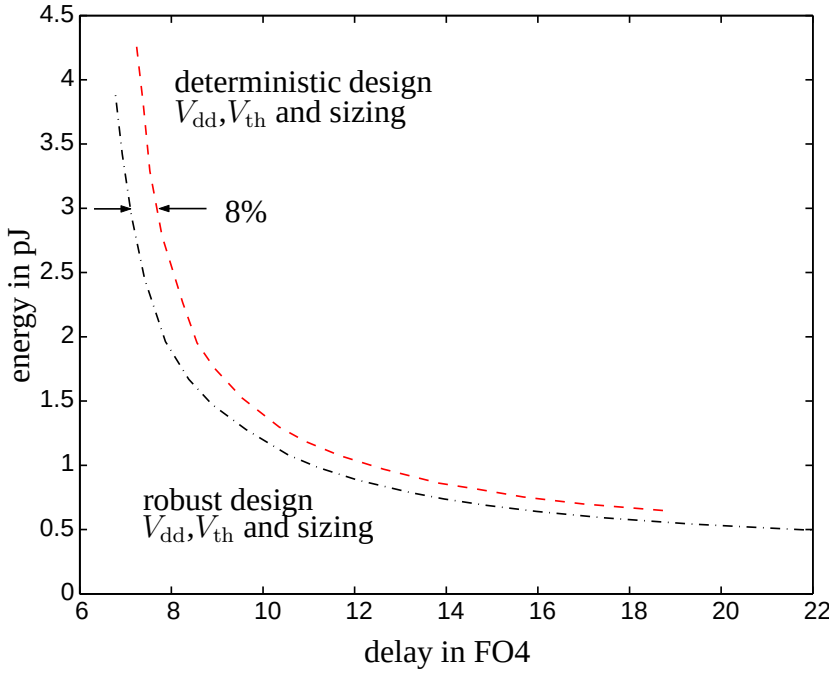


Figure 4.19: Improvement in the energy-delay tradeoff curve for the 95^{th} percentile delay due to statistical design

$Q_{0.95}$ energy-delay tradeoff curves for different circuits. Figure 4.19 shows a minimum of 8% improvement in the energy-delay tradeoff curve for $Q_{0.95}(\mathbf{T}_{\rm d})$ of a 32bit adder. The improvement at the higher delay side is mainly due to curbing delay variations from device sizing, while the gain at the higher energies is mainly from including $V_{\rm th}$ variations in estimating leakage energy.

4.7 Summary

Process variations cause uncertainties in gate delays which lead to uncertainty in the overall delay of the circuit. Calculating the complete PDF or a percentile point of the overall delay precisely is difficult, but upper and and lower bounds of a particular percentile point can be efficiently calculated by performing STA on surrogate netlists using additional margins on individual gate delays. Monte Carlo methods can also be used to estimate the PDF and percentile points.

Deterministic sizing results into many equally critical paths which is statistically worse for delay. The optimizer downsizes many of these paths to contain smaller gates which, according to Pelgrom's model, exhibit larger delay uncertainty and significantly increase the delay variance of that path and thus the overall delay. For robustness against process variations, the correct problem to solve is to minimize some α percentile of the overall delay ($Q_\alpha(\mathbf{T}_\mathrm{d})$) rather than minimizing the nominal delay. In this case the optimizer will distribute the resources (area or energy) between different paths, and different gates within a path, such that the ratio of sensitivity of $Q_\alpha(\mathbf{T}_\mathrm{d})$ to sensitivity of area or energy is the same for all gate delays.

While the exact formulation of this statistical design problem is possible for a few simple circuits, it is extremely tedious for most circuits. However, the optimizer just needs to be prevented from making bad choices of making many paths critical and downsizing gates in the process. The two heuristic techniques described in this chapter – of adding σ delay margins to gate delay expressions and using soft max function to combine path delays at converging nodes – are effective in designing robust circuits by steering the optimizer to push back on non-critical paths and avoid small devices in paths dominating the overall delay. Thus, no propagation of delay PDFs is required and similar to the deterministic sizing, the statistical sizing problem can be cast as a GP, with modest overhead.

Results of applying these heuristics show that the improvement in robustness depends strongly on the topology. Circuits with many structurally identical paths benefit less from statistical design as no path delay can be traded with another for sizing. The bigger the asymmetry among the logical paths in the netlist, the larger is the benefit of statistical design over deterministic one.

As energy is the sum of many terms, sum of the average energies for all gates and nets is a good estimate for the overall energy. However, leakage energy increases exponentially with decreasing $V_{\rm th}$ and hence is statistically dominated by the low $V_{\rm th}$ devices. Therefore, the average leakage is not the leakage of an average device, the variance of $V_{\rm th}$ – which depends inversely on device size –also affects the value. As this variation depends on sizing, we can steer the optimizer away from using highly leaky devices that will degrade the energy of the circuit. Using the proposed heuristic techniques of statistical sizing with the correct estimate of energy give a robust design having a much better energy-delay tradeoff curve under process variations.

Chapter 5

Conclusions

With high end microprocessors facing steep cooling costs and rising demands for longer battery life in portable devices, energy-efficiency is crucial in digital systems today. For energy-efficiency, the ratios of marginal cost (or sensitivity) of delay and energy to all the user tunable design variables – sizing, $V_{\rm dd}$, $V_{\rm th}$, logic style and topology should be equal, so that no variable can be traded off with another for a better design. For continuous variables, like sizing, $V_{\rm dd}$ and $V_{\rm th}$, the circuit design problem can be formulated as a Geometric Program, which can be solved very efficiently. The Stanford Circuit Optimization Tool facilitates schematic design entry, automatic problem formulation and post-optimization timing analysis. We used this tool to generate optimal designs for different topologies and logic styles of a 32-bit adder, which served as our case study. By overlapping the different tradeoff curves we generated the overall energy-delay tradeoff curve for 32-bit addition spanning a range of 60x in energy and 10x in delay. This curve not only allows designers to find the best design for given specifications, but also provides the information about the energy and delay costs (or gains) involved in changing to a different specification.

With shrinking feature size, local random process variations are becoming increasingly

detrimental to circuit performance. While the goal should be to optimize for the efficiency of circuits that are actually produced, in fact, ignoring process variations by using fixed parameters during optimization can actually make the design more vulnerable to process variations. Making all paths critical, which is the optimal condition for deterministic sizing, pushes out the overall delay under variations. At the same time, ignoring the exponential dependence of leakage energy on $V_{\rm th}$ results into designs that consume excessive static power after fabrication.

For an optimal statistical solution that results in a robust design, the ratio of sensitivities of a specified percentile delay point (as opposed to nominal delay) and energy should be made equal for all the design variables. We show that this problem can be solved exactly for certain circuit topologies for some distributions. However, it is impractical to solve for most typical circuits. Instead, simple heuristics can be used to get most of the benefit of statistical sizing. We have developed two heuristics that effectively include process variations during optimization and approximately solve the statistical design problem for all topologies. We add standard deviation margins to the mean delay model used in deterministic sizing and use "softmax", instead of $\max()$ to combine delays at converging nodes. Unlike deterministic sizing, these heuristics effectively target two aspects of a robust design – resizing stages with high delay uncertainty within the critical paths, and pushing back on paths that need not be critical and/or have a high delay variance.

Because the total energy is a sum of many varying terms, the average total energy is a relatively good estimate of the overall energy. However due to nonlinear dependence of dynamic energy on $V_{\rm dd}$ and leakage energy on $V_{\rm th}$, the average energy is not the same as the energy calculated with average values of $V_{\rm dd}$ and $V_{\rm th}$. It depends on the variations in these parameters. These variations add a factor that must be considered to determine the correct estimate of energy in statistical optimization.

The proposed heuristic method for designing robust energy-efficient circuits largely retains the original GP formulation for deterministic design problem so we can still leverage the benefit of efficient GP solutions. The extent to which a given circuit can be made robust is highly dependent on its topology. For structurally symmetric circuits like the decoder, the robust design is same as the nominal design, while for designs where there is a large disparity in sensitivities, either due to different number of stages in different paths, or different loads at different outputs, the improvement can be up to 30% assuming Pelgrom's model for variations.

5.1 Future work

The idea of balancing the ratio of marginal energy-delay cost across all design variables can be used at other levels of design hierarchy to create energy-efficient systems. In particular, at the micro-architecture level, it can be used to allocate the optimal amount of energy to different functional units, cache, load-store units and so on depending on their marginal costs to the overall performance and energy of the system. SCOT enables us to generate the Pareto-optimal E-D tradeoff curves of different circuit blocks. These curves can be used in another tool that takes in the architecture and optimizes for different performance-energy points. Work is ongoing in our group to develop architecture level energy-performance models that enable design of energy-efficient systems using the energy-delay models for different circuit blocks [3]. A similar approach can be used to design devices aimed for a particular circuit requirement. An example for BJTs is presented in Ref. [41].

Ensuring a particular overall parametric yield involves a statistical design methodology that deals with all the three different kinds of variations, namely, global chip-to-chip, correlated within-chip and local random variations. As discussed in Section 2.3, design

techniques exist for dealing with the first two. This thesis focused on designing robust circuits for local random variations. The next step is to generate a complete statistical solution by integrating the proposed algorithms in the existing design flow.

Appendix A

Geometric programming basics

The following material is taken from Ref. [9].

A.1 Monomial and posynomial functions

Let $x_1, \ldots, x_n$ denote n real positive variables, and $x = (x_1, \ldots, x_n)$ a vector with components x_i. A real valued function f of x, with the form

$$f(x) = cx_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \tag{A.1}$$

where $c > 0$ and $a_i \in \mathbf{R}$, is called a *monomial function*, or more informally, a *monomial* (of the variables $x_1, \ldots, x_n$). We refer to the constant c as the *coefficient* of the monomial, and we refer to the constants $a_1, \ldots, a_n$ as the *exponents* of the monomial. As an example, $2.3x_1^2 x_2^{-0.15}$ is a monomial of the variables x_1 and x_2, with coefficient 2.3 and x_2-exponent -0.15.

Any positive constant is a monomial, as is any variable. Monomials are closed under multiplication and division: if f and g are both monomials then so are fg and f/g. (This

includes scaling by any positive constant.) A monomial raised to any power is also a monomial:

$$f(x)^\gamma = (cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n})^\gamma = c^\gamma x_1^{\gamma a_1}x_2^{\gamma a_2}\cdots x_n^{\gamma a_n}.$$

The term 'monomial', as used here (in the context of geometric programming) is similar to, but differs from the standard definition of 'monomial' used in algebra. In algebra, a monomial has the form (A.1), but the exponents a_i must be nonnegative integers, and the coefficient c is one. Throughout this thesis, 'monomial' refers to the definition given above, in which the coefficient can be any positive number, and the exponents can be any real numbers, including negative and fractional.

A sum of one or more monomials, *i.e.*, a function of the form

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}}, \tag{A.2}$$

where $c_k > 0$, is called a *posynomial function* or, more simply, a *posynomial* (with K terms, in the variables $x_1, \ldots, x_n$). The term 'posynomial' is meant to suggest a combination of 'positive' and 'polynomial'.

Any monomial is also a posynomial. Posynomials are closed under addition, multiplication, and positive scaling. Posynomials can be divided by monomials (with the result also a posynomial): If f is a posynomial and g is a monomial, then f/g is a posynomial. If γ is a nonnegative integer and f is a posynomial, then f^γ always makes sense and is a posynomial (since it is the product of γ posynomials).

Let us give a few examples. Suppose x, y, and z are (positive) variables. The functions (or expressions)

$$2x, \quad 0.23, \quad 2z\sqrt{x/y}, \quad 3x^2y^{-.12}z$$

are monomials (hence, also posynomials). The functions

$$0.23 + x/y, \quad 2(1 + xy)^3, \quad 2x + 3y + 2z$$

are posynomials but *not* monomials. The functions

$$-1.1, \quad 2(1 + xy)^{3.1}, \quad 2x + 3y - 2z, \quad x^2 + \tan x$$

are not posynomials (and therefore, not monomials).

A.2 Standard form Geometric Program

A *geometric program* (GP) is an optimization problem of the form

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 1, & i = 1, \ldots, m, \\ & g_i(x) = 1, & i = 1, \ldots, p, \end{array} \tag{A.3}$$

where f_i are posynomial functions, g_i are monomials, and x_i are the optimization variables. (There is an implicit constraint that the variables are positive, *i.e.*, $x_i > 0$.) We refer to the problem (A.3) as a geometric program in *standard form*, to distinguish it from extensions we will describe later. In a standard form GP, the objective must be posynomial (and it must be minimized); the equality constraints can only have the form of a monomial equal to one, and the inequality constraints can only have the form of a posynomial less than or equal to one.

As an example, consider the problem

$$\begin{array}{ll} \text{minimize} & x^{-1}y^{-1/2}z^{-1} + 2.3xz + 4xyz \\ \text{subject to} & (1/3)x^{-2}y^{-2} + (4/3)y^{1/2}z^{-1} \leq 1, \\ & x + 2y + 3z \leq 1, \\ & (1/2)xy = 1, \end{array}$$

with variables x, y and z. This is a GP in standard form, with $n = 3$ variables, $m = 2$ inequality constraints, and $p = 1$ equality constraints.

We can switch the sign of any of the exponents in any monomial term in the objective or constraint functions, and still have a GP. For example, we can change the objective in the example above to $x^{-1}y^{1/2}z^{-1} + 2.3xz^{-1} + 4xyz$, and the resulting problem is still a GP (since the objective is still a posynomial). But if we change the sign of any of the coefficients, or change any of the additions to subtractions, the resulting problem is not a GP. For example, if we replace the second inequality constraint with $x + 2y - 3z \leq 1$, the resulting problem is *not* a GP (since the lefthand side is no longer a posynomial).

A.2.1 Simple extensions of GP

Several extensions are readily handled. If f is a posynomial and g is a monomial, then the constraint $f(x) \leq g(x)$ can be handled by expressing it as $f(x)/g(x) \leq 1$ (since f/g is posynomial). This includes as a special case a constraint of the form $f(x) \leq a$, where f is posynomial and $a > 0$. In a similar way if g_1 and g_2 are both monomial functions, then we can handle the equality constraint $g_1(x) = g_2(x)$ by expressing it as $g_1(x)/g_2(x) = 1$ (since g_1/g_2 is monomial). We can maximize a nonzero monomial objective function, by minimizing its inverse (which is also a monomial).

As an example, consider the problem

$$\begin{array}{ll} \text{maximize} & x/y \\ \text{subject to} & 2 \leq x \leq 3, \\ & x^2 + 3y/z \leq \sqrt{y}, \\ & x/y = z^2, \end{array} \tag{A.4}$$

with variables $x,\ y,\ z \in \mathbf{R}$ (and the implicit constraint $x,\ y,\ z > 0$). Using the simple transformations described above, we obtain the equivalent standard form GP

$$\begin{array}{ll} \text{minimize} & x^{-1}y \\ \text{subject to} & 2x^{-1} \leq 1, \quad (1/3)x \leq 1, \\ & x^2y^{-1/2} + 3y^{1/2}z^{-1} \leq 1, \\ & xy^{-1}z^{-2} = 1. \end{array}$$

It's common to refer to a problem like (A.4), that is easily transformed to an equivalent GP in the standard form (A.3), also as a GP.

A.3 Generalization

In this section we describe extensions of GP that are less obvious than the simple ones described in Section A.2.1. This leads to the idea of *generalized posynomials*, and an extension of Geometric Programming called *Generalized Geometric Programming*.

We say that a function f of positive variables $x_1, \ldots, x_n$ is a *generalized posynomial* if it can be formed from posynomials using the operations of addition, multiplication, positive (including fractional) power, and maximum.

Let us give a few examples. Suppose x_1, x_2, x_3 are positive variables. The function

$$\max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\}$$

is a generalized posynomial, since it is the maximum of two posynomials. The function

$$\left(0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7}\right)^{1.5}$$

is a generalized posynomial, since it is the positive power of a posynomial. All of the functions appearing in the examples of the two previous sections, as the objective or on the lefthand side of the inequality constraints, are generalized posynomials.

As a more complex example, the function

$$h(x) = (1+\max\{x_1, x_2\})\left(\max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\} + \left(0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7}\right)^{1.5}\right)^{1.7}$$

is a generalized posynomial. This can be seen as follows:

- x_1 and x_2 are variables, and therefore posynomials, so $h_1(x) = \max\{x_1, x_2\}$ is a generalized posynomial.
- $1+x_1$ and $2x_1+x_2^{0.2}x_3^{-3.9}$ are posynomials, so $h_2(x) = \max\left\{1 + x_1, 2x_1 + x_2^{0.2}x_3^{-3.9}\right\}$ is a generalized posynomial.
- $0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7}$ is a posynomial, so $h_3(x) = \left(0.1x_1x_3^{-0.5} + x_2^{1.7}x_3^{0.7}\right)^{1.5}$ is a generalized posynomial.
- h can be expressed as $h(x) = (1 + h_1(x))\,(h_2(x) + h_3(x))^{1.7}$ (*i.e.*, by addition, multiplication, and positive power, from h_1, h_2, and h_3) and therefore is a generalized posynomial.

Generalized posynomials are by definition, closed under addition, multiplication, positive powers, and maximum, as well as other operations that can be derived from these, such as division by monomials. They are also closed under composition in the following sense. If f_0 is a generalized posynomial of k variables, for which no variable occurs with a negative exponent, and $f_1, \ldots, f_k$ are generalized posynomials, then the composition function

$$f_0(f_1(x), \ldots, f_k(x))$$

is a generalized posynomial.

A very important property of generalized posynomials is that they satisfy the log convexity property that posynomials satisfy. If f is a generalized posynomial, the function

$$F(y) = \log f(e^y)$$

is a convex function: for any $y,\ \tilde{y}$, and any θ with $0 \leq \theta \leq 1$, we have

$$F(\theta y + (1-\theta)\tilde{y}) \leq \theta F(y) + (1-\theta)F(\tilde{y}).$$

In terms of the original generalized posynomial f and variables x and $\tilde{x}$, we have the inequality

$$f(x_1^\theta \tilde{x}_1^{1-\theta}, \ldots, x_n^\theta \tilde{x}_n^{1-\theta}) \leq f(x_1, \ldots, x_n)^\theta f(\tilde{x}_1, \ldots, \tilde{x}_n)^{1-\theta},$$

for any θ with $0 \leq \theta \leq 1$.

A.3.1 Generalized geometric program

A *generalized geometric program* (GGP) is an optimization problem of the form

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 1, & i = 1, \ldots, m, \\ & g_i(x) = 1, & i = 1, \ldots, p, \end{array} \tag{A.5}$$

where $g_1, \ldots, g_p$ are monomials and $f_0, \ldots, f_m$ are generalized posynomials. Since any posynomial is also a generalized posynomial, any GP is also a GGP.

While GGPs are much more general than GPs, *they can be mechanically converted to equivalent GPs* [10] As a result, GGPs can be solved very reliably and efficiently, just like GPs. The conversion from GGP to GP can be done automatically by a parser as it parses the expressions describing the problem. The GP modeler only needs to know the rules for forming a valid GGP, which are very simple to state. There is no need for the user to ever see, or even know about, the extra variables introduced in the transformation from GGP to GP.

Unfortunately, the name 'generalized geometric program' has been used to refer to several different types of problems, in addition to the one above. For example, some authors have used the term to refer to what is usually called a *signomial program*[1], a very different generalization of a GP, which in particular cannot be reduced to an equivalent GP, or easily solved.

Once we have the basic idea of a parser that scans a problem description, verifies that it is a valid GGP and transforms it to GP form (for numerical solution), we can add several

[1] A signomial is a function with the same form as a posynomial (i.e., (A.2), where the coefficients c_j are allowed to be negative. A signomial program (SGP) is a generalization of a geometric program, which has the form of a GP, but the objective and constraint functions can be signomials

useful extensions. The parser can also handle inequalities involving negative terms in expressions, negative powers, minima, or terms on the righthand side of inequalities, in cases when they can be transformed to valid GGP inequalities. For example, the inequality

$$x + y + z - \min\{\sqrt{xy}, (1 + xy)^{-0.3}\} \leq 0 \tag{A.6}$$

(which is certainly not a valid generalized posynomial inequality) could be handled by a parser by first replacing the minimum with a variable t_1 and two *upper bounds*, to obtain

$$x + y + z - t_1 \leq 0, \qquad t_1 \leq \sqrt{xy}, \qquad t_1 \leq (1 + xy)^{-0.3}.$$

Moving terms around (by adding or multiplying) we obtain

$$x + y + z \leq t_1, \qquad t_1 \leq \sqrt{xy}, \qquad t_1 t_2^{0.3} \leq 1, \qquad 1 + xy \leq t_2,$$

which is a set of posynomial inequalities. (Of course we have to be sure that the transformations are valid, which is the case in this example.)

The fact that a parser can recognize an inequality like (A.6) and transform it to a set of valid posynomial constraints is a double-edged sword. The ability to handle a wider variety of constraints makes the modeling job less constrained, and therefore easier. On the other hand, few people would immediately recognize that the inequality (A.6) can be transformed, or understand *why* it is a valid inequality. A user who does not understand why it is valid will have no idea how to modify the constraint (*e.g.*, add terms, change coefficients or exponents) so as to maintain a valid inequality.

Appendix B

Stanford Circuit Optimization Tool

The tool consists of five major components made up of perl scripts, C++ code and interface to other stand-alone tools like schematic editors (SUE), IRSIM, MOSEK, SPICE and so on. A brief description of the sections in the order of their use is as follows.

1. **Schematic Entry**: The netlist is entered in a schematic editor like SUE [59], with the design constraints specified as annotated comments. To facilitate automatic modeling, the user is restricted to using Channel Connected Components (CCCs) (defined in Section 3.3.1) as the basic logic gates while making circuits. The spice file generated from SUE is then modified with perl scripts to extract all the commands from the SUE comments and flatten the design hierarchy to the CCC level. SCOT uses the modified spice file as its main input.

2. **Generating Models and Switching Statistics**: Analytic delay and leakage models are generated for each CCC type in the modified netlist using equations for transistor current models for a chain of transistors as explained in Section 2.1.4. The activity and duty factors necessary for dynamic and leakage power expressions are generated by performing switch level simulations using IRSIM.

3. **Problem formulation**: Using the delay-energy models, the activity and duty factors, and netlist connectivity, this part of the software generates the sizing-$V_{\rm dd}$-$V_{\rm th}$ allocation problem as a Geometric Program with the constraints and objectives specified by the user.

4. **GP solver**: The formulated Geometric program (GP) is solved using MOSEK [61]. The solution file contains the optimal values of the objective function and design variables.

5. **Post Analysis**: The user can also specify post analysis operations like SSTA, drawing PDFs, obtaining path delays and variances and so on. This section consists of various scripts to back annotate the MOSEK results in Spice file or schematics, for verification and/or visualization.

More elaborate description of each section is provided in the user manual for the tool [18].

B.1 Modeling issues in important circuit scenarios

The gate delay and energy can be accurately modeled as posynomials. The calculation of overall circuit delay also leads to convex timing constraints. This facilitates the formulation of the sizing problem as a GP. However, there are important constraints of a real design that are either not convex or are hard to model. Following are some of the important scenarios covered in the optimizer. The aim is to use a work-around in a way that leads to the correct sizing for the real design constraint.

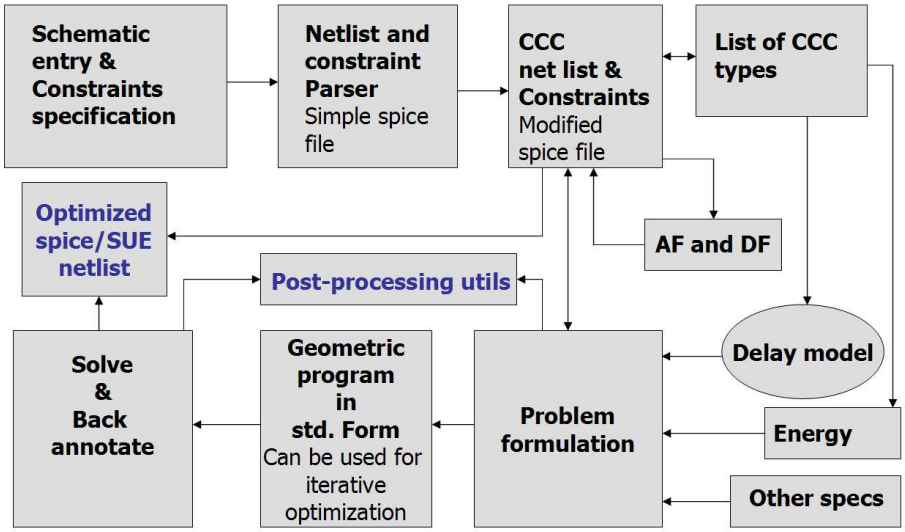


Figure B.1: Block diagram of Stanford Circuit Optimization Tool (SCOT)

B.1.1 Signal rise/fall time constraints

To avoid signal integrity issues like excess short circuit current and meta-stability it is important to keep the signal rise and fall times within a certain limit. The accuracy of the delay model also degrades with slower rise/fall signal transitions. The non-linear nature of device currents, Miller capacitance kick back and other effects like cross talk make it hard to model the 10-90% rise/fall time of any signal. However the 10-90% rise/fall time is directly related to the step delay of the driving gate. Thus we simulate the rise/fall time constraint by constraining the delay per stage to be within a specified limit.

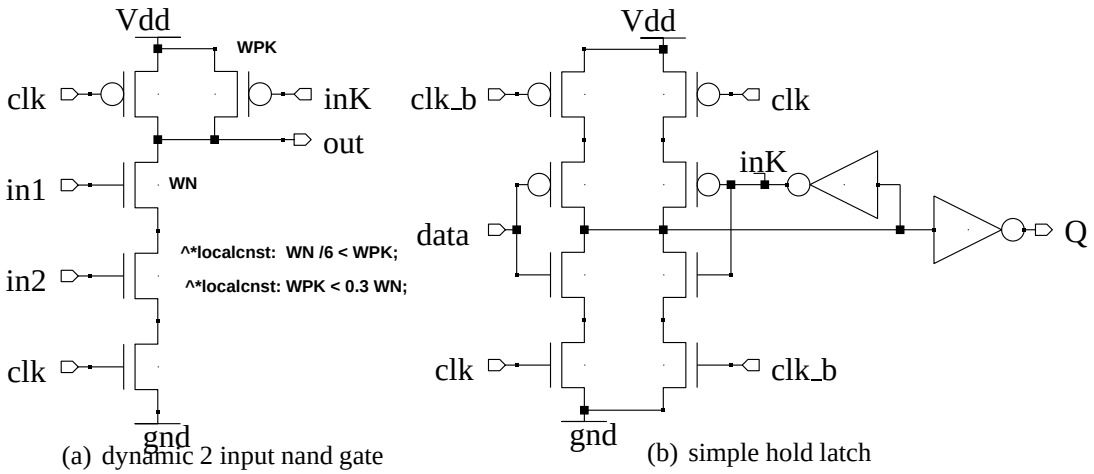
B.1.2 Transmission gate circuits

A transmission gate consists of an NMOS and PMOS transistor connected in parallel and turned on and off by a pair of complimentary signals. The sizing should be such that the arrival times of complimentary signals are sufficiently close for simultaneous switching of

both devices. The improvement in the drive strength due to having two parallel transistors of opposite kinds has to be modeled accurately for proper sizing of CCCs that contain a transmission gate. While constraining the complimentary signals to arrive within a certain time window is a non-convex constraint, what we care for is correct sizing that will enable it. We model the arrival time of both the inputs of the transmission gate as the maximum of their actual individual arrival times. This way the circuit generating the complimentary signal is forced to speed paths leading to both the inputs, making them as close to each other as possible. Examination of optimized circuits shows that indeed this technique is successful. With the assurance of almost simultaneous switching, we model the delay of the transmission gate by using the same drive current expression as that of a CMOS gate, with a higher effective width of the switching transistor. In our work, the width NMOS and PMOS devices in a transmission gate is kept the same.

B.1.3 Local feedback - keepers

Figure B.2 shows two examples where local feedback is used in the form of keepers in dynamic logic and feedback inverters in latches. It is necessary to recognize the transistors in the local feedback paths while sizing. First, their effect on forward signal propagation due to parasitic capacitive loading and current fighting (if any) should be considered. The feedback devices have to be large enough to maintain the value the node against leakage, and small enough to not prevent the node transition due to current fighting. SCOT allows these two sided constraints to be included easily within the CCC itself. Having ensured the transition, the increase in delay due to current fighting can be taken into account by assuming a higher effective load capacitance at that node. For sizing purposes, this works well and enables the sizing of keepers and feedback inverters. Secondly, the delay through the feedback path is not a part of the signal propagation delay through the circuit. In



(a) dynamic 2 input nand gate (b) simple hold latch

Figure B.2: Local feedback in logic circuits: keepers and feedback inverters

SCOT, users can specify the inputs to feedback devices by naming them in a special way. The optimizer then ignores the timing arcs that go through them. This also means ignoring the timing arcs of any logic that feeds the keeper device alone. These circuits are then sized based on other constraints like the signal rise/fall constraints and so on. To size self-reset logic gates, additional pulse width constraints have to included for every gate to ensure a good timing margin for reset.

B.1.4 Pulse width constraints

Certain circuits like pulse-mode flip-flops are designed to have a fixed pulse width between two particular nets. This is usually done with an inverter chain. Figure B.3 shows a cartoon where the problem is to optimize the entire combinational circuit CL for energy-efficiency while designing sub-circuit S to have a fixed timing pulse T_{pulse} between two nets n1 and n2. The delay of sub-circuit S, T_{ckt}, assuming n1 as the input would be a posynomial. Ideally we would like to have a constraint such as $T_{\text{ckt}} = T_{\text{pulse}}$. However, equality involving a posynomial is not allowed in GP. We can try to implement this constraint by specifying the inequalities $T_{\text{ckt}} \leq T_{\text{pulse}}$ and $T_{\text{ckt}} \geq T_{\text{pulse}}$. While the former is acceptable, the latter is

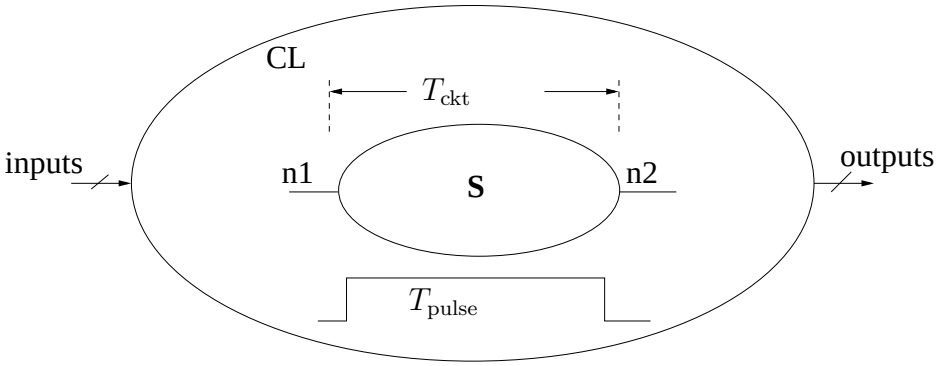


Figure B.3: Combinational logic block CL with pulse width constraints on a sub-block S

still not a valid GP constraint[1]. A possible work around is to add the following constraints instead.

$$T_{\text{ckt}} \leq T_{\text{pulse}}, T_{\text{n1}} + \max(T_{\text{pulse}}, T_{\text{ckt}}) \leq T_{\text{n2}}.$$

Here, T_{n1} and T_{n2} are the signal arrival times at nets n1 and n2. The first constraint forces the pulse to be no wider than T_{pulse}, while the second provides room for slowing S to the point where $T_{\text{ckt}} = T_{\text{pulse}}$. Unless the minimum size or slope constraint becomes active, this is usually achieved in practice because downsizing S to achieve T_{pulse} saves energy that can be distributed to other parts of the circuit.

B.1.5 Modeling dual-rail gate delay

Dual rail domino gates like the one shown in Figure B.4 have complimentary input-pairs that the gate delay modeling algorithm may mistake for independent inputs. This will result in modeling the delay for the path that contains both the input and its complement. This is clearly incorrect as such a path can never be sensitized. To avoid this in SCOT, the user can specify a pair of inputs as complimentary inputs in the schematics so that the delay

[1] as it does not represent a convex space in the log-log domain.

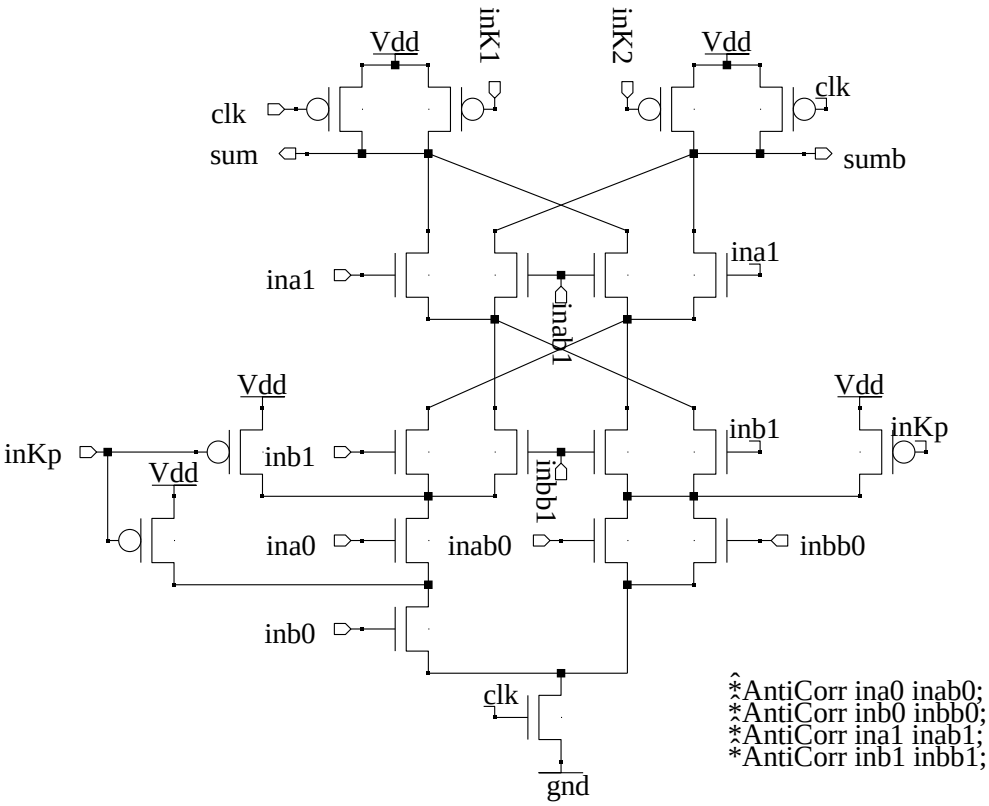


Figure B.4: Schematic of the CCC that generates the bit sum for carry=0 in a 2bit sum select ling adder. Special commands shown on the bottom right are necessary to avoid choosing false transistor paths for delay

modeling routine avoids such erroneous paths.

B.1.6 Handling discrete variables

While the GP algorithm admits only continuous variables, in real designs we have discrete V_{th}s to choose from. Cell libraries also have discrete cell sizes. Dealing with the discrete variables exactly leads to a combinatorial problem which is hard to solve. Instead, we relax the discrete requirements by solving with continuous variables and then progressively snap the variables to their discrete values starting with snapping the ones closest to the grid values first and iterating. In most circuits of any reasonable size (gate count > 20) the

sensitivity of delay to individual device sizes or V_{th}s is very small. We observed less than

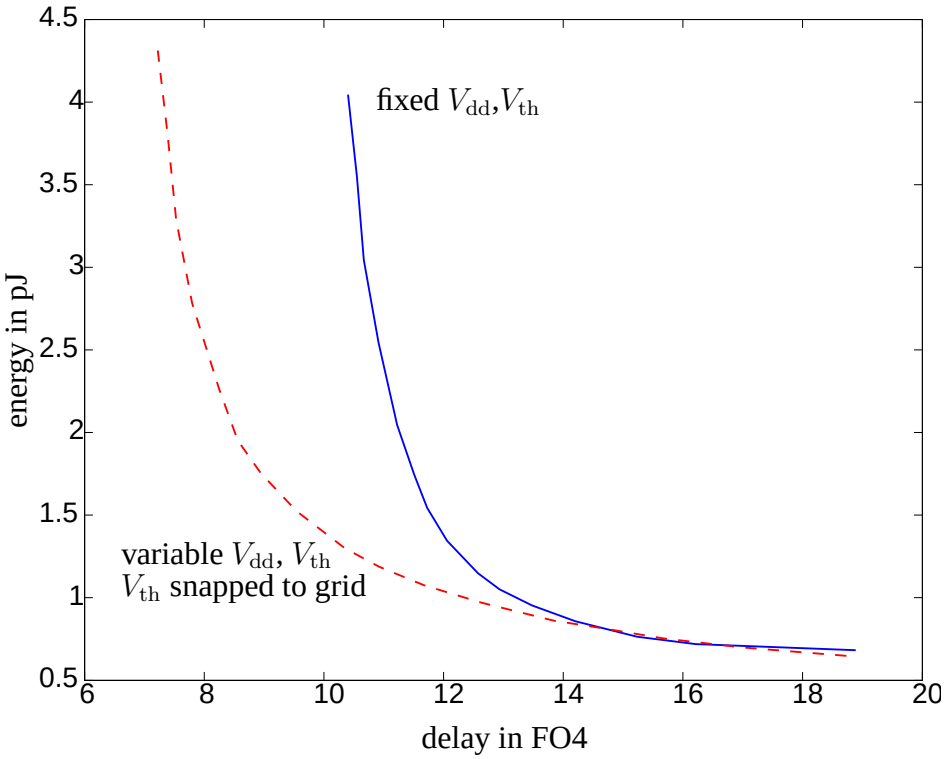


Figure B.5: Snapping of discrete variables causes sub-optimality

1% degradation in 32-bit adder designs from using discrete V_{th}s. However, snapping does lead to visible sub-optimality under some conditions. Figure B.5 shows the energy-delay tradeoff curves for two optimizations on a 32bit adder, one in which V_{dd} and V_{th} are fixed to nominal values and the other in which they are variable with three discrete V_{th}s. Each device has its own V_{th}. Naturally the latter results in a better tradeoff, as it has more variables to design with. However, near the energy-delay points where the nominal values of V_{dd} and V_{th} are already very close to the optimal, designing with variable V_{dd} and V_{th} and then snapping can lead to a worse design. However the effect is negligible.

Bibliography

[1] M. Hershenson, S. Boyd, and T. Lee. Design of Pipeline Analog-to-Digital Converters via Geometric Programming. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pages 317–324, November 2002.

[2] S. Augsburger and B. Nikolić. Combining dual-supply, dual-threshold and transistor sizing for power reduction. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 316–321, September 2002.

[3] O. Azizi, J. Collins, D. Patil, H. Wang, and M. Horowitz. Processor performance modeling using symbolic simulation. To be presented at *The IEEE International Conference on Computer Design: VLSI in Computers and Processors*, April 2008.

[4] D. Bertsimas, K. Natarajan, and C.-P. Teo. Probabilistic combinatorial optimization: Moments, semidefinite programming and asymptotic bounds. *SIAM Journal on Optimization*, 15(1):185–209, 2005.

[5] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the* 40^{th} *IEEE/ACM Design Automation Conference*, pages 338–342, June 2003.

[6] S. Boyd and S.-J. Kim. Geometric Programming for circuit optimization. In *Proceedings of International Symposium on Physical Design (ISPD)*, pages 44–46, April 2005.

[7] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz. Digital circuit sizing via geometric programming. *Operations Research*, 53(6):899–932, 2005.

[8] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz. A heuristic method for statistical digital circuit sizing. In *Proceedings of SPIE*, volume 6156, 2006.

[9] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on Geometric Programming. *Optimization and Engineering*, 8(1):67–127, 2007.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[11] S. Boyd, L. Vandenberghe, A. El Gamal, and S. Yun. Design of robust global power and ground networks. In *Proceedings ACM/SIGDA Symposium on Physical Design (ISPD)*, pages 60–65, April 2001.

[12] R. E. Bryant. A switch-level model and simulator for MOS digital systems. *Transaction on Computers*, C-33:160–177, Feb 1984.

[13] J. Burt and M. Garman. Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science*, 18(1):207–217, September 1971.

[14] N. Chabini, I. Chabini, E. Aboulhamid, and Y. Savaria. Methods for minimizing dynamic power consumption in synchronous designs with multiple supply voltages. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(3):346–351, 2003.

[15] K. Chen, H. Hu, P. Fang, M. Lin, and D. Wollesen. Predicting CMOS speed with gate oxide and voltage scaling and interconnect loading effects. *IEEE Transactions on Electron Devices*, 44(11):1951–1957, 1997.

[16] T. Chen and S. Naffziger. Comparison of Adaptive Body Bias (ABB) and Adaptive Supply Voltage (ASV) for improving delay and leakage under the presence of process variation. *IEEE Transactions on VLSI systems*, 11(5):888–899, Oct 2003.

[17] A. Conn, I. Elfadel, W. Molzen Jr., P. O'Brien, P. Strenski, C. Visweswariah, and C. Whan. Gradient-based optimization of custom circuits using a static-timing formulation. In *Proceedings of the* 36^{th} *IEEE/ACM Design Automation Conference*, pages 452–459, June 1999.

[18] D. Patil and S.-J. Kim. *The Stanford Circuit Optimization Tool (SCOT) User Guide*, 2006. Available from `http://www.stanford.edu/class/ee371/tools/SCOT_UserGuide.pdf`.

[19] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfets with very small physical dimensions. *IEEE journal of solid state circuits (JSSC)*, 9(5):256–268, Oct 1974.

[20] L. Devroye. Inequalities for the completion times of stochastic PERT networks. *Mathematics of Operations Research*, 4(4):441–447, 1979.

[21] A. Dharchoudhury and S. M. Kang. Analytical fast timing simulation of MOS circuits driving RC interconnects. In *Proceedings of* 10^{th} *International Conference on VLSI design*, pages 111–116, Jan 1997.

[22] B. Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33:862–881, 1985.

[23] W. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19(1):55–63, 1948.

[24] J. Fishburn and A. Dunlop. TILOS: A posynomial programming approach to transistor sizing. In *Digest of Technical Papers of IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 326–328, 1985.

[25] P. Glasserman. *Monte Carlo Methods in Financial Engineeting*. Springer-Verlag, 2003.

[26] J. Hagstrom and N. Jane. Computing the probability distribution of project duration in a PERT network. *Networks*, 20:231–244, 1990.

[27] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers*, 16(3):72–80, 1999.

[28] D. Harris. A taxonomy of parallel prefix networks. In *Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 2213–2217, November 2003.

[29] D. Harris. Logical effort of higher valency adders. In *Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, pages 1358–1362, November 2004.

[30] D. Harris and I. Sutherland. Logical effort of carry propagate adders. In *Proceeding of the the Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, pages 873–878, 2003.

[31] M. Hashimoto and H. Onodera. A performance optimization method by gate sizing using statistical static timing analysis. In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 111–116, may 2000.

[32] U. Heller. On the shortest overall duration in stochastic acyclic network. *Methods of Operations Research*, 42:85–104, 1981.

[33] M. Hershenson, S. Boyd, and T. Lee. Optimal design of a CMOS op-amp via geometric programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):1–21, January 2001.

[34] M. A. Horowitz. *Timing models for MOS circuits*. PhD thesis, Stanford University, 1984.

[35] M. A. Horowitz. *Lecture notes of ee313, Digital MOS integrated circuits,* 2000. Available from `http://eeclass.stanford.edu/ee313/`.

[36] W. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, M. Irwin, and Y. Tsai. Total power optimization through simultaneously multiple-V_{DD} multiple-V_{TH} assignment and device sizing with stack forcing. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, pages 144–149, August 2004.

[37] F. Ishihara, F. Sheikh, and B. Nikolić. Level conversion for dual-supply systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(2):185–195, 2004.

[38] E. T. A. F. Jacobs and M. R. C. M. Berkelaar. Gate sizing using a statistical delay model. In *Proceedings of Design and Test in Europe (DATE) Conference*.

[39] M. Johnson, D. Somasekhar, L.-Y. Chiou, and K. Roy. Leakage control with efficient use of transistor stacks in single threshold CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(1):1–5, February 2002.

[40] S. Joshi and S. Boyd. An efficient method for large-scale gate sizing. To appear in *IEEE Transactions on Circuits and Systems*, Submitted in December 2006.

[41] S. Joshi, S. Boyd, and R. W. Dutton. Optimal doping profiles via Geometric Programming. *IEEE Transactions on Electron devices*, 52(12):2660–2675, Dec 2005.

[42] K. Kasamsetty, M. Ketkar, and S. Sapatnekar. A new class of convex functions for delay modeling and its application to the transistor sizing problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):779–788, July 2000.

[43] J. F. Kenney and E. S. Keeping. *Mathematics of Statistics, Part 2*. Van Nostrand, second edition.

[44] M. Ketkar and S. Sapatnekar. Standby power optimization via transistor sizing and dual threshold voltage assignment. In *Proceedings of the IEEE/ACM international Conference on Computer-Aided Design*, pages 375–378, November 2002.

[45] S.-J. Kim, S. Boyd, S. Yun, D. Patil, and M. Horowitz. A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing. *Optimization and Engineering*, 8(4):397–430, 2007. Available from `www.stanford.edu/~boyd/heur_san_opt.html`.

[46] S. Knowles. A family of adders. In *Proceedings of* 14^{th} *IEEE Symposium on Computer Arithmetic*, pages 30–34, 1999.

[47] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: a 32-way multithreaded Sparc processor. *IEEE Micro*, 25:21–29, Mar-Apr 2005.

[48] S. Kulkarni and D. Sylvester. High performance level conversion for dual V_{DD} design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(9):926–936, 2004.

[49] R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of ACM*, 27(4):831–838, Oct 1980.

[50] A. S. Leon, K. W. Tam, J. L. Shin, D. Weisner, and F. Schumacher. A power-efficient high-throughput 32-thread SPARC processor. *IEEE Journal of Solid-State Circuits*, 42:7–16, 2007.

[51] H. Ling. High speed binary adder. *IBM Journal of Research and Development*, 25(3):126–130, May 1981.

[52] A. Ludwig, R. Möhring, and F. Stork. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research*, 102:49–64, 2001.

[53] M. Mani, A. K. Singh, and M. Orshansky. Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization. In *(ICCAD) IEEE/ACM International Conference on Computer-Aided design.*, pages 19–26, Nov 2006.

[54] D. Marković, V. Stojanović, B. Nikolić, M. Horowitz, and R. Brodersen. Methods for true energy-performance optimization. *IEEE Journal of Solid-State Circuits*, 39(8):1282–1293, 2004.

[55] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar. A 4GHz 130-nm address generation unit with 32-bit sparse-tree adder core. *IEEE Journal of Solid State Circuits*, 38(5):689–695, 2003.

[56] S. K. Mathew, M. A. Anders, B. Bloechel, T. Nguyen, R. K. Krishnamurthy, and S. Borkar. A 4-GHz 300-mW 64-bit integer execution ALU with dual supply voltages in 90-nm CMOS. *IEEE Journal of Solid-State Circuits*, 40(1):44–51, Jan 2005.

[57] R. McGowen, C. A. Poirier, C. Bostak, J. Ignowski, M. Millican, W. H. Parks, and S. Naffziger. Power and temperature control on a 90-nm Itanium family processor. *IEEE Journal of Solid-State Circuits*, 41:229–237, 2006.

[58] I. Mejilson and A. Nadas. Convex majorization with an application to the length of critical path. *Journal of Applied Probability*, 16:671–677, 1979.

[59] Micro Magic Inc. *SUE Design Manager datasheet*, 2004. Available from `http://www.micromagic.com/tools/sue_datasheet.html`.

[60] G. Moore. Cramming more Components into Integrated Circuits. *Electronics Magazine*, 38(8):256–268, April 1965.

[61] MOSEK ApS. *The MOSEK Optimization Tools Version 3 User's Manual and Reference*, 2002. Available from `http://www.mosek.com`.

[62] S. Naffziger. A sub-nanosecond 0.5μm 64b adder design. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 362–363, February 1996.

[63] S. Narendra, V. De, S. Borkar, D. A. Antoniadis, and A. P. Chandrakasan. Full-chip subthreshold leakage power prediction and reduction techniques for sub-0.18-μm CMOS. *IEEE Journal of Solid-State Circuits*, 39(3):501–510, 2004.

[64] S. Nassif. Within chip variability analysis. In *International Electron Devices Meeting (IEDM) Technical Digest*, pages 283–286, Dec 1998.

[65] T. N. Nguyen and J. D. Plummer. Physical mechanisms responsible for short-channel effects in MOS devices. In *International Electron Devices Meeting (IEDM) Technical digest*, pages 596–599, 1981.

[66] E. J. Nowak. Maintaining the benefits of CMOS scaling when scaling bogs down. *IBM Journal of Research and Development*, 46(2/3):169–180, May 2002.

[67] V. G. Oklobdzija, B. R. Zeydel, H. Q. Dao, S. Mathew, and R. Krishnamurthy. Comparison of high-performance VLSI adders in the energy-delay space. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(6):754–758, june 2005.

[68] D. Patil, R. Ho, O. Azizi, R. Ananthraman, and M. Horowitz. Robust energy-efficient adder topologies. In *Proceedings of* 18^{th} *IEEE Symposium on Computer Arithmetic*, pages 30–34, 2007.

[69] D. Patil, S. Yun, S.-J. Kim, S. Boyd, and M. Horowitz. A new method for design of robust digital circuits. In *Proceedings of* 6^{th} *International Symposium on Quality Electronic Design (ISQED)*, 2005.

[70] M. Pelgrom, C.-J. Duinmaijer, and A.-P.-G. Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid State Circuits*, 24(5):1433–1439, October 1989.

[71] D. C. Pham, T. Aipperspach, D. Boerstler, M. Bolliger, R. Chaudhry, D. Cox, P. Harvey, P. M. Harvey, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Pham, J. Pille, S. Posluszny, M. Riley, D. L. Stasiak, M. Suzuoki, O. Takahashi, J. Warnock, S. Weitzel, D. Wendel, and K. Yazawa. Overview of the architecture, circuit design, and physical implementation of a first-generation Cell processor. *IEEE Journal of Solid-state circuits*, 41:179–196, Jan 2006.

[72] R. F. Pierret. *Semiconductor device fundamentals*. Addison-Wesley, 1996.

[73] S. Raj, S. B. K. Vrudhula, and J. Wang. A methodology to improve timing yield in the presence of process variations. In *Proceedings of the* 41^{st} *IEEE/ACM Design Automation Conference (DAC)*, pages 448–453, San Diego, California, Jun 2004.

[74] P. Robillard and M. Trahan. The completion times of PERT networks. *Operations Research*, 25:15–29, 1976.

[75] S. Rusu, S. Tam, H. Muljono, D. Ayers, and J. Chang. A dual-core multi-threaded Xeon processor with 16MB L3 cache. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 315–324, Feb 2006.

[76] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–593, 1990.

[77] V. Sandararajan and K. K. Parhi. Synthesis of low power CMOS VLSI circuits using dual supply voltages. In *Proceedings of* 35^{th} *Design Automation Conference*, June 1999.

[78] R. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, New York, 1980.

[79] A. Shogan. Bounding distributions for a stochastic PERT network. *Networks*, 7:359–381, 1977.

[80] J. Sklansky. Conditional-sum addition logic. *IRE Transactions on Electronic Computers*, EC-9:226–231, June 1960.

[81] R. Van Slyke. Monte Carlo methods and the PERT problem. *Operations Research*, 11:839–860, 1963.

[82] C. G. Sodini, P. K. Ko, and J. L. Moll. The effect of high fields on MOS device and circuit performance. *IEEE Transactions of Electron Devices*, 31:1386–1393, Oct 1984.

[83] A. Srivastava and D. Sylvester. Minimizing total power by simultaneous $V_{\rm dd}/V_{\rm th}$ assignment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(5):665–677, 2004.

[84] A. Srivastava, D. Sylvester, and D. Blaauw. Power minimization using simulataneous gate sizing dual-Vdd and dual-Vth assignment. In *Proceedings of the* $41^{\rm st}$ *Design Automation Conference*, pages 783–787, jun 2004.

[85] L. Su, R. Schulz, J. Adkisson, K. Beyer, G. Biery, W. Cote, E. Crabbe, D. Edelstein, J. Ellis-Monaghan, E. Eld, D. Foster, R. Gehres, R. Goldblatt, N. Greco, C. Guenther, J. Heidenreich, J. Herman, D. Kiesling, L. Lin, S. H. Lo, J. McKenna, C. Megivern, H. Ng, J. Oberschmidt, A. Ray, N. Rohrer, K. Tallman, T. Wagner, and B. Davari. A high-performance sub-0.25μm CMOS technology with multiple thresholds and copper interconnects. In *Symposium on VLSI Technology Digest of Technical Papers*, pages 18–19, June 1998.

[86] R. Sullivan and J. Hayya. A comparison of the Method of Bounding Distributions (MBD) and Monte Carlo simulation for analyzing stochastic acyclic networks. *Operations Research*, 28(3):614–617, May-Jun. 1980.

[87] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort: Designing fast CMOS Circuits*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[88] Y. Taur and T. H. Ning. *Fundamentals of modern VLSI devices*. Cambridge University Press, ninth edition, 2006.

[89] K.-Y. Toh, P.-K. Ko, and R. Meyer. An engineering model for short channel MOS devices. *IEEE Journal of Solid-State Circuits*, 23(4):950–958, 1988.

[90] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits*, 37(11):1396–1402, Nov 2002.

[91] N. Tzartzanis, W. W. Walker, H. Nguyen, and A. Inoue. A 34word$\times$64b 10R/6W write-through self timed dual-supply-voltage register file. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 338–339, Feb 2002.

[92] K. Usami and M. Horowitz. Clustered voltage scaling technique for low-power design. In *Proceedings of the 1995 International Symposium on Low Power Design*, pages 3–8, San Diego, CA, June 1995.

[93] A. van der Vaart. *Asymptotics Statistics*. Cambridge University Press, 1998.

[94] Y. Wang and K. K. Parhi. A unified adder design. In *Record of the* 35^{th} *Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 177–282, 2001.

[95] G. Weiss. Stochastic bounds on distributions of optimal value functions with applications to PERT, network flows and reliability. *Operations Research*, 34(4):595–605, 1986.

[96] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, third edition, 2004.

[97] L. S. Y. Wong, S. Hossain, A. Ta, J. Edvinsson, D. H. Rivas, and H. Naas. A very low-power CMOS mixed-signal IC for implantable pacemaker applications. *IEEE Journal of Solid-State Circuits*, 39:2446–2456, 2004.

[98] P. N. Strenski D. J. Hathaway X. Bai, C. Visweswariah. Uncertainty aware circuit optimization. In *Proceedings of the* 39^{th} *IEEE/ACM Design Automation Conference (DAC)*.

[99] Y.-J. Yeh, S.-Y. Kuo, and J.-Y. Jou. Converter-free multiple-voltage scaling techniques for low-power CMOS digital design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):172–176, 2001.

[100] R. Zlatanovici and B. Nikolić. Power-performance optimal 64-bit carry-lookahead adders. In *Proceedings of the* 29^{th} *European Solid-State Circuits Conference (ESSCIRC)*, pages 321–324, 2003.