A REDUCED CONSTRAINT SET LINEAR PROGRAM FOR LOW-POWER DESIGN OF DIGITAL CIRCUITS

BY TEZASWI RAJA

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Vishwani D. Agrawal and Prof. Michael L. Bushnell

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

March, 2002

ABSTRACT OF THE THESIS

A Reduced Constraint Set Linear Program for Low-Power Design of Digital Circuits

by Tezaswi Raja

Thesis Director: Dr. Vishwani D. Agrawal and Prof. Michael L. Bushnell

A new linear programming formulation is derived for the minimum energy design of digital CMOS circuits. The variables of the *linear program* (LP) are the delays of gates and buffers assumed on fanouts. Using a constraint set, the LP determines the required gate and buffer delays, minimizing the number of non-zero delay buffers. It is known that the dynamic power consumption of the circuit is minimized when the delay of each gate equals or exceeds the differential path delay at its inputs. Under this condition all unneccessary power-consuming transitions are suppressed. In addition, the overall input to output circuit delay must stay within the specified performance limit. This method can also be used simultaneously to speed up the critical paths of the circuit. In the previously published work, these constraints have been expressed as path delays. Since the number of paths can be an exponential function of the number of gates in the circuit, such a formulation becomes impractical for large circuits.

The contribution of the present work is to derive an LP with a constraint set whose size is linear in the number of gates, without compromising the accuracy of the solution. We introduce two new delay variables at each gate or buffer. These are the earliest and the latest arrival times, respectively. Including its own delay, thus a gate or a buffer is characterized by three variables. The advantage of increasing the number of variables is that now the global path delay inequalities can be replaced by local inequalities. The constraints to be satisfied at a gate or buffer only involve its own variables and those of its neighbors. We show that the total number of such constraints for the entire circuit is a linear function of the number of gates. As a result, we find that the number of constraints for the benchmark circuit c880 is reduced to 3,611 from 6.96 million needed for the path constraints.

We analytically prove that the new reduced constraint-set LP produces the same exact solution as that of the path constraint LP. The constraint derivation for several benchmark circuits shows that the size of the constraint set indeed grows linearly with the circuit size. The LP is solved using the AMPL modeling language to obtain the minimum energy designs for example circuits of varying sizes.

Acknowledgements

I am thankful to Dr. Vishwani Agrawal and Dr. Michael Bushnell without whose cooperation and encouragement this document would not be what it is. I would like to thank my seniors at Rutgers, Vivek, Aditya and Abhishek, for their support and advice. I thank my friends Vijay, Dennis, Sameer, Hari, Zakir and Tambe for making my life easier in this foreign country. I am appreciative of my endless list of junior colleagues who have made my time in the lab an enjoyable experience.

Dedication

To my parents

Table of Contents

List of Figures

Chapter 1
Introduction

The contribution of this work is a new *linear programming* (LP) formulation for the gate-level optimization of digital circuits for low power with a given upper bound on the overall input to output delay of the circuit. The size of the constraint set of the new linear program remains linear in the size of the circuit and makes optimization of large circuits feasible. This is a significant improvement over the existing LP models whose constraint set grows exponentially with the size of the circuit.

1.1 Motivation

In this section, we address several basic issues that have motivated this work.

1.1.1 Why do we need to reduce power in CMOS circuits?

The continuing decrease in feature size and corresponding increase in chip density and operating frequency have made power consumption a major concern in VLSI design. Excessive power dissipation in integrated circuits discourages their use in a portable systems. It also causes overheating, which degrades the performance and reduces chip lifetime. To control their temperature levels the chips need specialized and costly packaging and cooling arrangements which would result in the further escalation of the system cost. The growing need for portable communication devices and computing systems has increased the need for optimization of power consumption in a chip. Overall, low power design is a critical technology

needed in the semiconductor industry today. Simultaneously, we also need to speed up the critical paths of the circuit, while reducing its power consumption.

1.1.2 Why do we need linear programming (LP) for optimization?

Power optimization techniques have been implemented in various levels of design. This work deals mainly with the logic level. The optimization of power is achieved by eliminating the spurious transitions known as *glitches* in the circuit. A total elimination of glitches requires differential path delays to satisfy certain conditions at every gate. This is done either by controlling the delays of gates or by inserting delay buffers [1, 2]. At the same time the overall delay of the circuit should remain within the specified limit. The system is modeled as a linear program with constraints set up to obtain the desired conditions at every gate in the circuit and to limit the overall delay. A solution finds the delays of all gates and buffers and at the same time minimizes the number of buffers. Linear programming is preferred as it guarantees a global optimum and hence the result would produce the maximum power savings.

1.1.3 Why are existing LP techniques not feasible?

The above-mentioned optimization techniques using LP are quite efficient for circuits of small size and complexity. The constraint set is derived for every gate by enumerating all paths from the inputs of the gate to the *primary inputs* (PIs) of the circuit. It is known that the number of paths in a circuit can increase exponentially with the size of the circuit and so can the number of constraints. This would make the constraint set size increase exponentially with the circuit size. The feasibility of solving an optimization problem is limited to the maximum number of constraints that can be handled by the optimizing algorithm. Since this maximum constraint limit is exceeded by circuits of large size, this technique could not become industrially useful. For example, the ISCAS'85 benchmark circuit c880 has 469 gates

and would require 6.95 million constraints for the path delay method. Thus a new LP model is needed that preserves all the relevant attributes of the original techniques but with a much reduced and linearly growing constraint set.

1.2 Problem statement

The problem solved in this thesis is: *Find a new LP formulation for the elimination of all glitches in a combinational digital circuit, while holding the overall input to output delay within the specified bound, such that the constraint set size is linear in the size of the circuit.*

1.3 Original contributions of the thesis

In this work we propose a new LP formulation for glitch elimination in a digital circuit. The earlier LP formulations have used a single variable for every gate and net in the circuit. The new constraint set is obtained by introducing two new arrival time variables at every gate. These constraints guarantee the same proven minimality of the dynamic power as the path delay constraints. The experimental results confirm the claims made here and we also have results for much larger circuits that could not be optimized by the existing LP techniques due to the size constraint. As an example, the circuit c880, which has 469 gates, required 3,611 constraints in comparison to the 6.95 million constraints needed for the path delay method.

1.4 Organization of the thesis

We survey the published literature on this subject in Chapter 2. Readers who are familiar with the prior work can skip Chapter 2. Chapter 2 also gives the power estimation techniques that have been used in this research for comparison and to test the effectiveness of the optimization. We give basic definitions and concepts related to two of the original LP

optimization techniques in Chapter 3. The new LP formulation is described in Chapter 4. We derive theorems on which the new formulation is based and also give examples to illustrate the constraint set formation. The validity of this model is proven in the latter sections of Chapter 4, showing it to be equivalent to the original models. In Chapter 5, we present the results of optimization on a simple one bit adder and a 4-bit ALU. We have also included data showing the drastic decrease in the number of constraints from the original model for all ISCAS'85 benchmark circuits. Finally in Chapter 6, we present the conclusion and propose future work.

Chapter 2
Prior Work

2.1 Introduction

We review the basic mechanisms of power dissipation in CMOS circuits. We have also chronicled several existing techniques of low power design with emphasis on the specific disadvantages of each method that inspired this work. Although all types of circuits, such as analog, digital logic, and memory, are important our discussion will focus on digital circuits.

2.2 Power dissipation in CMOS circuits

There are three main sources of power dissipation in digital CMOS circuits:

- dynamic power
- short circuit power
- leakage current power.

2.2.1 Dynamic power

The dynamic power dissipation in a CMOS gate is due to the charging and discharging of load capacitance driven by the gate. This capacitance consists of the internal capacitances of the gate, wire capacitances of the fanout net and the capacitances of the gate terminals of

the transistors being controlled by the fanout net. This power dissipation can be calculated by the following equation [21]:

$$P_{dyn} = \frac{1}{2} C_{load} V_{dd}^2 f D \tag{2.1}$$

where

- P_{dyn}: dynamic power dissipation of gate
- C_{load}: load capacitance of the gate
- f: clock frequency
- V_{dd}: supply voltage
- D: transition density of the output of the gate

The *transition density* is the average number of transitions during a clock cycle. The dynamic power dissipated is thus proportional to the number of transitions occurring at a gate. In prior technologies, dynamic power accounted for most of the power used by CMOS circuits [49]. But with the advent of deep sub-micron technology the other components of power consumpton are also becoming significant. However, several low power techniques have concentrated on minimizing dynamic power, as will be explained later. Dynamic power can be classified into necessary switching activity for the correct functioning of the circuit and unnecessary transitions due to unbalanced paths in the circuit. The latter component of dynamic power dissipation is the *glitching power* and is elaborated in the next section.

2.2.2 Hazards and glitch power

Before signals of a digital circuit reach steady state, gates can have multiple transitions. Since the power consumed is directly proportional to the number of transitions, these unnecessary transitions increase power consumption. These transitions are called *glitches* or *hazards*.

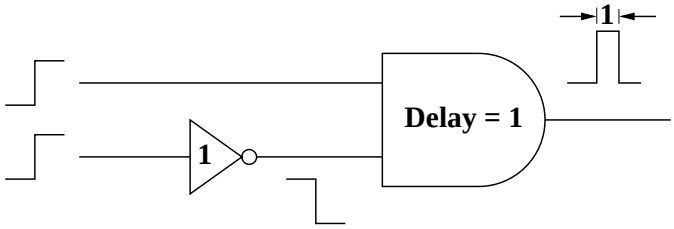

Figure 2.1: An example of static hazard.

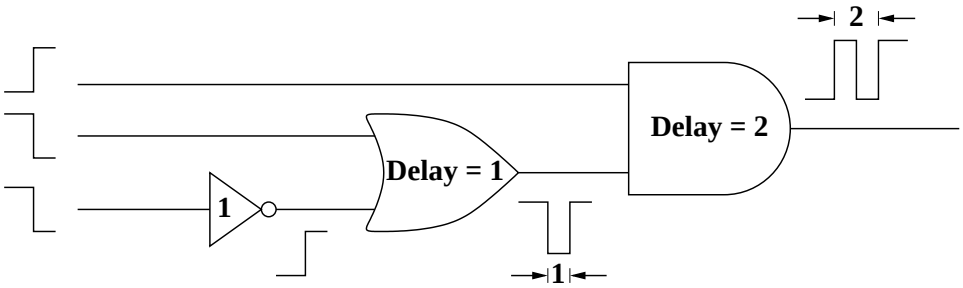

Figure 2.2: An example of a dynamic hazard.

Glitches are produced in a circuit due to the difference in signal arrival times at the inputs of a gate. Power consumed by glitches is called *glitch power* and it typically amounts to about 20% of the overall power consumption of the chip and even 70% in some typical cases as the combinational adder [52]. In this section we familiarize the reader with the terminology of hazards and also the issues involved before we discuss the techniques for glitch suppression in the next section.

Consider the example of Figure 2.1 with each gate having one unit of delay. Due to the difference in arrival times of signals at the inputs of the AND gate the logic value glitches or emits a pulse of 1 unit width, which equals the inverter delay. This is known as a *static hazard*. In Figure 2.2 the OR gate produces a static hazard of 1 unit. The transient consists of three edges, two rising and 1 falling. This is a *dynamic hazard* with a combined width of 2 units.

The number of edges in transients at the output of a gate may equal the number of arriving signals at the gate. The maximum difference in arrival time of signals at the inputs

of a gate is called *differential path delay* and is also the maximum width of the possible glitch at the circuit output. A *hazard producing gate* has more than one input and has a non-zero differential path delay. Every gate has an *inertial delay* due to the finite switching speed of the transistors. It is the time a device takes to switch the output after the cause for the change has occurred at the input [1]. Inertial delay plays a major role in distorting the glitches produced at the gates and in the next section we present ways of eliminating glitches by either making the differential path delay zero or by increasing the inertial gate delay.

2.2.3 Short circuit power dissipation

Short-circuit power dissipation occurs when a gate switches. During the transition there is a short time when both the *nMOS* and *pMOS* transistors conduct. This is equivalent to shorting the supply and ground rails for a brief amount of time. The current flowing through these transistors dissipates power.

The value of the short circuit current greatly depends on the capacitive load on the output of the gate. Consider the example in Figure 2.3. For high capacitive load, the output fall time is significantly larger than the input rise time. It follows that the short circuit current is very close to zero in this case. Conversely, for a low capacitive load, the output fall time is substantially smaller than the input rise time. It follows that the short circuit current is at its maximum. Thus, the short circuit power optimization techniques are aimed at making the output rise/fall times larger than the input rise/fall times. Thus, the amount of short-circuit power depends on switching speed of the gate. The amount of short-circuit power dissipated by an inverter can be calculated by the formula:

$$P_{short-circuit} = \frac{\beta}{12}(V_{dd} - 2V_T)^3 \frac{\tau}{T} \tag{2.2}$$

where

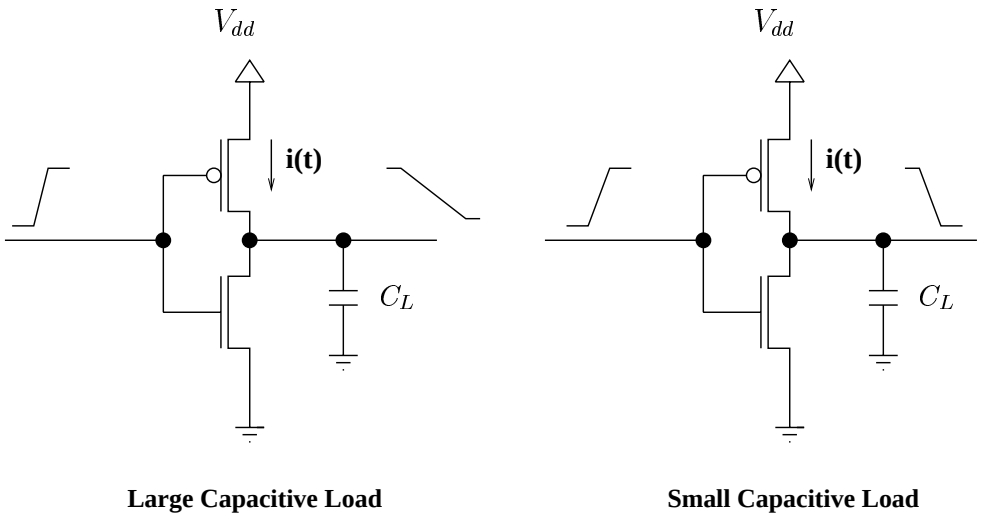

Figure 2.3: Effect of load capacitance on short circuit power dissipation.

- $P_{short-circuit}$: short circuit dissipation
- β: gain factor of the gate
- V_{dd}: supply voltage
- τ: rise or fall time of a signal
- T: period of the signal.

Figure 2.4 is a plot of energy dissipation versus the ratio of input rise/fall times to output rise/fall times [58]. As can be seen, the longer the input rise/fall time, the longer the short circuit current will flow, and the average short circuit power increases. For most ICs, the short circuit dissipation is approximately $5 - 10\%$ of the total power dissipated.

2.2.4 Leakage power dissipation

There are two types of leakage currents: reverse biased leakage on the transistor drains, and sub-threshold leakage through the channel of a device [11]. The magnitude of those currents

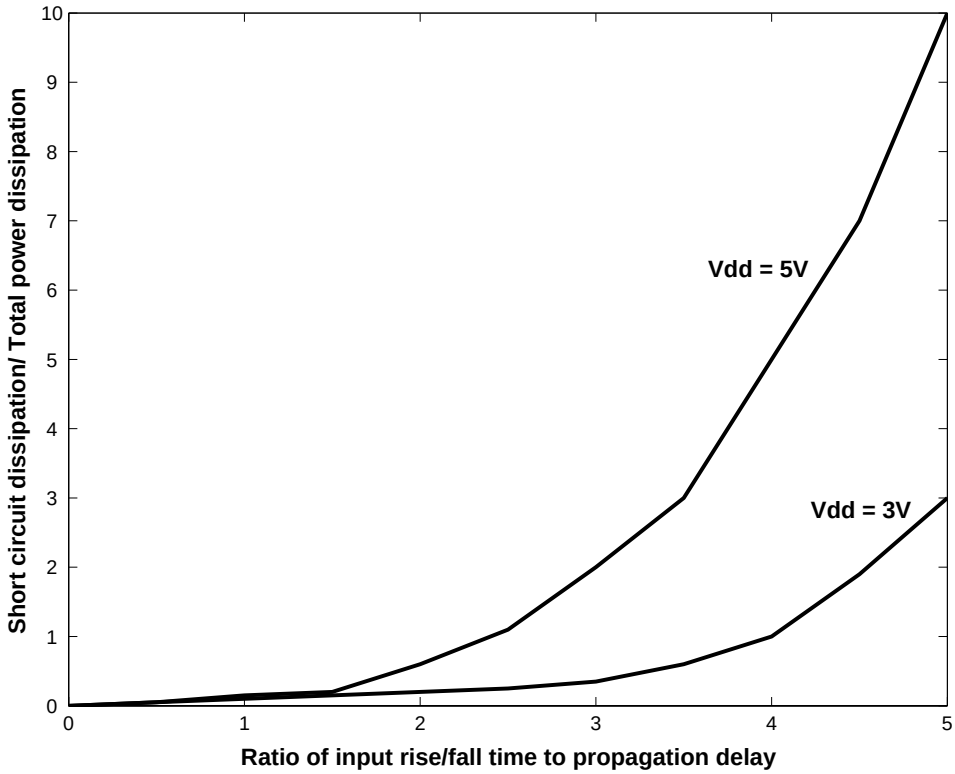

Figure 2.4: Short circuit energy dissipation versus input rise/fall time.

is set predominantly by the processing technology. However, there are some things that a designer can do to minimize their contribution.

Diode leakage current occurs when the transistor is turned off and, another active device charges up/down the drain with respect to the former's bulk potential. Consider the inverter in Figure 2.5 when it is given a low voltage as input. The pMOS transistor will be turned off but the nMOS will charge through the junction, making its drain-to-bulk voltage $-V_{dd}$. This diode current thus flows from a junction to the substrate and is given by the expression:

$$I = A_d \; J_s \tag{2.3}$$

where

- A_d: area of diffusion at the periphery of the transistor

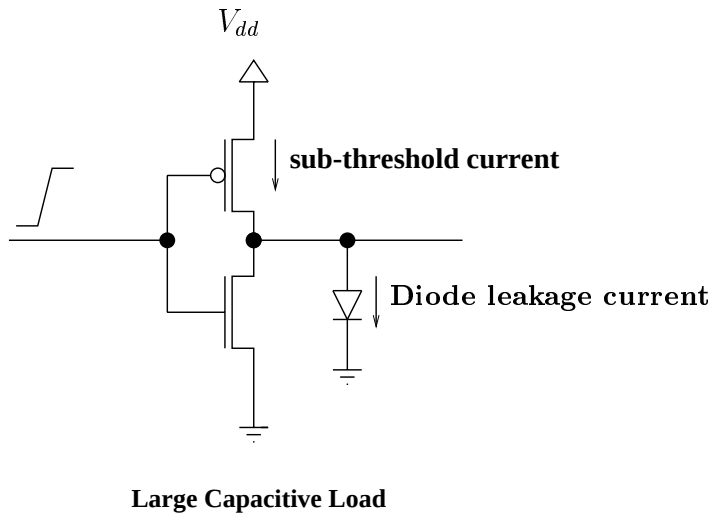

Figure 2.5: Leakage current in an inverter.

- J_s: leakage current density, set by the technology.

For a sub-micron process A_d is of the order of $10\mu m^2$ per transistor and J_s is of the order of $1-2pA/\mu m^2$ [64]. It is proportional to the diffusion area and perimeter, hence it is desired to minimize these two quantities. The leakage current density J_s is temperature sensitive, hence it can increase leakage power dissipation dramatically at high temperatures.

The other component of leakage current is the sub-threshold leakage. In the inverter shown in the Figure 2.5, even when the transistor is turned there is a current flowing through its channel due to the drain-to-source voltage V_{ds} and this is known as sub-threshold current I_d. The plot of I_d vs V_{ds} as shown in Figure 2.6 has an exponential relation in the sub-threshold region. The negative slope of the curve as seen in the figure in the sub-threshold region is known as the *drain induced barrier lowering* (DIBL).

The magnitude of the sub-threshold current is both a function of the process, device sizing and supply voltage. The process parameter that affects the current value is the threshold voltage V_t. Reducing V_t exponentially increases the sub-threshold current. For supply voltages of 1V, the V_t can be reduced upto 0.4V in deep-submiron enhancement-mode MOS devices [35]. The sub-threshold current is also proportional to the device size

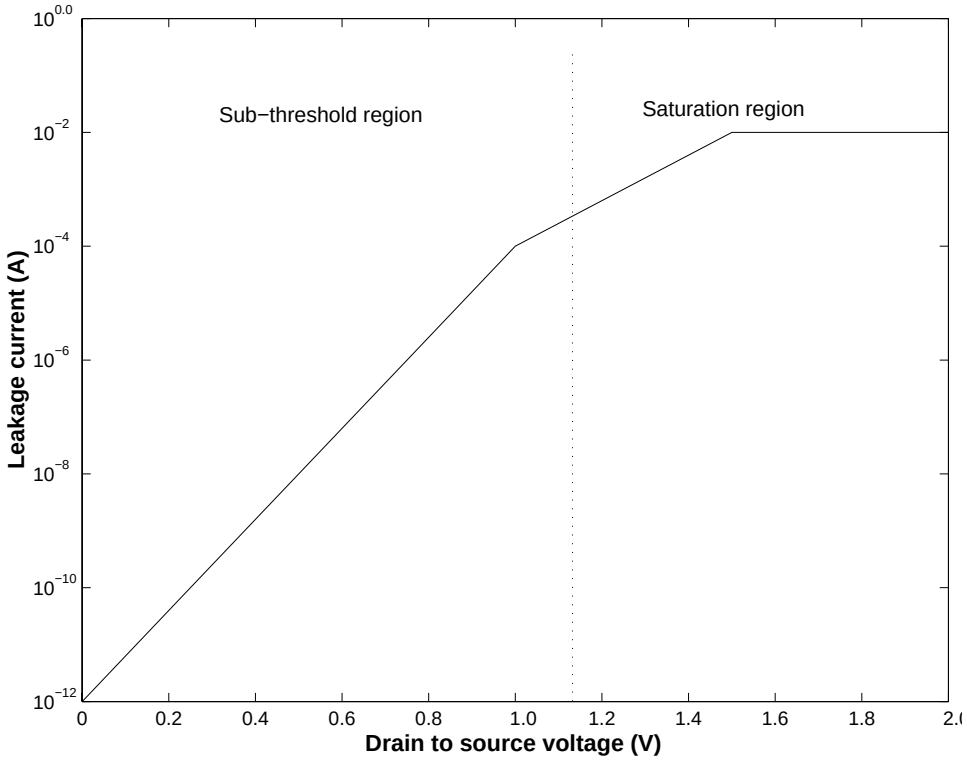

Figure 2.6: Leakage current vs. drain to source voltage characteristic of MOSFET.

(width/length). Thus the current can be minimized by reducing the transistor sizes, and by reducing the supply voltage. Voltage scaling is one of the techniques that reduces leakage power drastically and is described in the next section.

2.3 Techniques for low power design

There are several low-power techniques in vogue. We have listed a few of the major ones in this section. While we focus on digital circuits, a reader interested in the low-power analog circuits may refer to the books by Serdijn [51], and Sinencio and Andreou [54].

2.3.1 Voltage scaling

Dynamic power depends quadratically on the *supply voltage* of the circuit. Hence it is obvious that there is a quadratic reduction in power by reducing the supply voltage. The voltage scaling also increases the device reliability as it effectively reduces the ambient temperature of the chip and prevents devices from crossing their upper threshold temperatures [35]. But the price paid is the increase in the circuit delay. Since the decrease in supply voltage decreases the driving capacity of the gate, switching time is increased. This also increases the short-circuit and leakage power dissipation of the circuit [35, 51, 54].

Chandrakasan *et al.* [13, 14, 15, 16] propose a first order theory to estimate the dependencies of circuit styles and architectures on voltage scaling. They propose *architecture driven voltage scaling*, which decreases the power consumption but essentially keeps the overall delay of the system constant. They achieve this by introducing parallelism into the architecture. Consider the example of a datapath with a tolerable worst case frequency of f Hz and a supply voltage of V Volts. The dynamic power consumed would amount to:

$$P_{dynamic} = CV^2 f \tag{2.4}$$

Now if we duplicate the datapath into two separate datapaths with the same combined throughput, each of them could perform at frequency $f/2$. By their first order theory we can reduce the supply voltage to $V/2$ and since the capacitance C is doubled by duplication, we have the new expression for power as:

$$P_{dynamic} = 2C\frac{V^2}{4}\frac{f}{2} \tag{2.5}$$

$$P_{dynamic} = 0.25CV^2 f \tag{2.6}$$

Thus effectively the power has been reduced to 25% of its original value using architecture driven technique. Increase in chip area is the price of this technique.

The architecture level techniques have been taken to a new level by the new *Crusoe* chip of Transmeta Corporation [25]. The idea behind the chip was to design architecture with the

compiler's specification in mind. They offload most of the functioning of the microprocessor to the software, which allowed them to design streamlined hardware with about half the number of transistors as that of an *X86* microprocessor chip. Fewer transistors meant lower power. Another technique used to reduce hardware by Transmeta was the use of *virtual devices*. They work by using *code morphing* software to monitor the input and output instructions sent to the device, then to send those instructions to the virtual device instead. Detailed description of both code morphing and virtual devices is beyond the scope of this document.

Gonzalez *et al.* [26] suggest that for sub-micron technologies, lowering of supply and threshold voltages is advantageous when the transistors are velocity saturated and the nodes have a high activity factor. They suggest optimal power performance of chips at even 0.5V supply operation. But if there is any uncertainty in maintenance of the supply voltage at its prescribed levels the advantage of the low voltage operation vanishes. Hence, they showed examples of active feedback being used to eliminate uncertainty and greatly decrease power consumption.

Kawaguchi *et al.* [31] describe a *super cut-off CMOS* (SCCMOS) scheme to achieve high speed and low stand-by current in sub 1V supply CMOS technology. The main problem with reducing the supply voltage to such low levels is that the leakage power would increase in proportion to the dynamic power and would be comparable to it. But they propose to reduce this by decreasing the *threshold voltage* which would limit the leakage current to 1 pico-Ampere. Published results have shown operating chips at even 0.2V supply.

Gabara [24] described *pulsed power supply CMOS* (PPSCMOS) where supply voltages V_{dd} and V_{ss} are replaced by pulsed supply waveforms with same pulse width but a phase difference of ϕ. By making the period of the pulses equal to the largest gate delay in the circuit we ensure propagation of signals through the circuits. The advantage of this technique is that the existing CMOS circuits can be used in this mode by just a changing the power

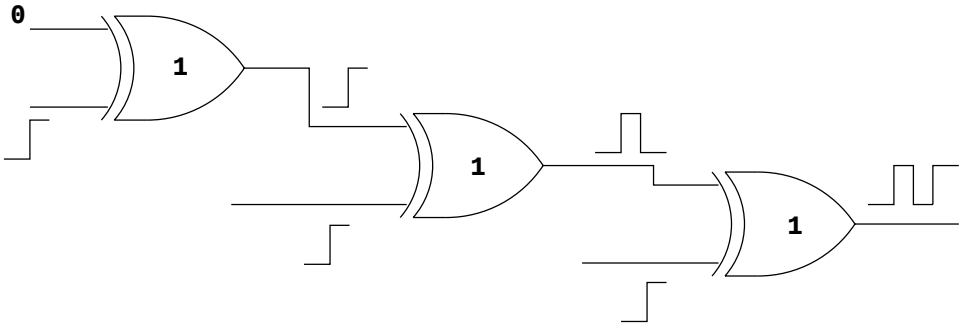

Figure 2.7: Glitching behavior of an ill-structured circuit.

supply.

Algorithms have been proposed for optimizing power at the *register-transfer level* through efficient scheduling of transformation selection [36] and power management during synthesis [37, 38].

2.3.2 Path balancing

As stated earlier glitch power is a major component of overall power consumption and there are many techniques to reduce it. One of the earliest is the *path balancing*. The glitches are produced due to difference in arrival times of signals at gate inputs. The idea behind this technique is to prevent glitches from occurring by balancing the delays of the gates such that at any given gate the signals arrive at the input terminal at the same time. Consider the example circuit shown in Figure 2.7. The signals at the inputs of gate arrive at different times which would probably result in glitches. This behavior can be suppressed by restructuring the circuit as shown in Figure 2.8 [40]. However, as the next example illustrates, a simple restructuring of gates may not be sufficient for some circuits.

Wong *et al.* in their papers [60, 61] on *wave pipelining* have described a method of balancing delays for circuits by the inserting buffers at selected inputs of the gates. The addition of buffers is done such that it will not increase the critical delay of the circuit, but will effectively eliminate spurious transitions. The buffers are inserted only in the fast paths

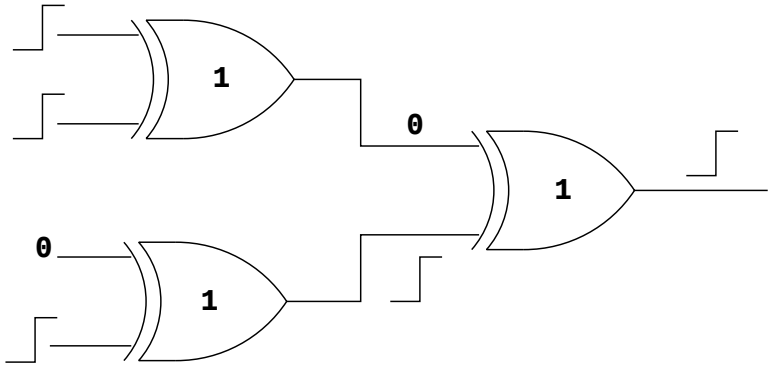

Figure 2.8: Removal of glitches after restructuring of the circuit in Figure 2.7.

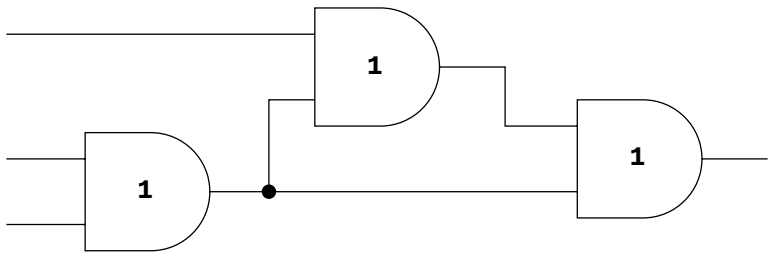

Figure 2.9: Glitch behavior of a circuit where simple restructuring will not balance paths.

of the circuit and since the slowest paths determine the speed of the chip they are untouched. For example the circuit in Figure 2.9 cannot be path balanced without the introduction of buffers as shown in Figure 2.10. However the addition of buffers increases the switching activity of the circuit which may offset the reduction in switching activity due to elimination of glitches.

In CMOS circuits the fanout load significantly affects the gate delay. To account for this Kim *et al.* [32, 33] developed a delay buffer insertion using a chain of buffers, thus eliminating the need for merging common buffers. The problem of delay buffer insertion on a multiple fanout net is simplified by assuming that only one size of delay buffer is used. This method permits a quick estimation of the number of buffers needed to balance the circuit and can therefore be used in heuristics for logic restructuring [12].

Path balancing has been used for some earlier multiplier architectures which increase

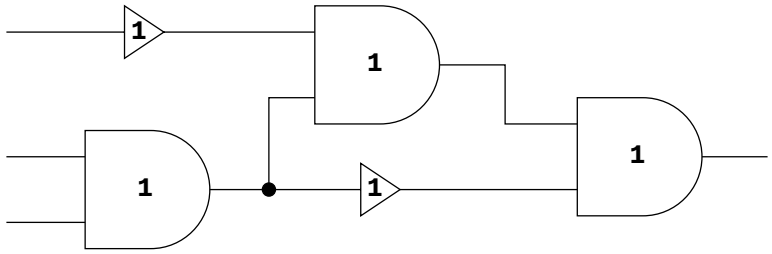

Figure 2.10: Path balancing of circuit in Figure 2.9 by buffers.

computational speed by reducing functional activity and also eliminate hazards [47]. Another method is to use self-timed design in which a combinational element would not compute until all signals have stabilized. This requires aditional circuitry to generate the timing or enable signals for various stages of logic [39, 48, 56]. The added circuitry reduces the power saving.

2.3.3 Hazard filtering

This method eliminates glitches in the circuit using inertial gate delays. Inertial delay is the interval that elapses after an input change before a gate can produce an output change. When the time between successive edges in the input signal is less than the inertial delay of the gate, the effects of these edges are suppressed. This is known as the *filtering effect* of the gates [1]. CMOS gates have a built-in delay called the *inertial delay* of the gate and they suppress signals that are of smaller width than the inertial delay from passing through the gate. This is known as the filtering effect of the gate. Instead of balancing the delays of the gate inputs to be exactly equal (as in delay balancing), this method evaluates the differential delay of inputs and increases the inertial delay of the gate to exceed the differential delay such that the glitch will be suppressed within the gate.

Consider the example inverter shown in Figure 2.11 with the total capacitance at the output node as C and the short circuit impedance R. When an input pulse as shown is incident at the input, a charging current *i(t)* flows through the upper transistor charging the capacitor C. This charging raises the output voltage until time $t = \tau$ when the rising edge

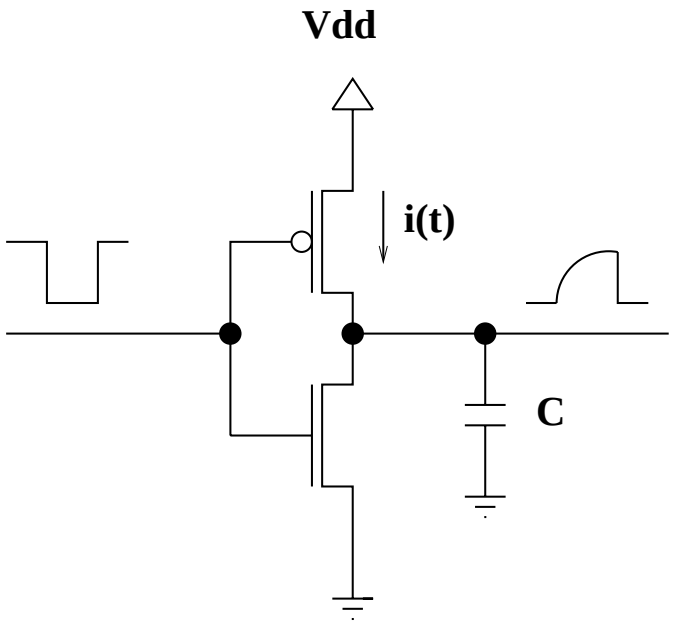

Figure 2.11: Energy dissipation in an inverter.

occurs at the input. The energy consumed by the gate is [59]:

$$E(\tau) = CV_{dd}{}^2(1 - e^{-\frac{\tau}{RC}}) \tag{2.7}$$

For hazard filtering, we increase the time constant RC of the gate and thus increase the charging time needed for the capacitor reach V_{dd}. When the time constant is large, there is practically no energy dissipation in the gate due to this short glitch of length τ. Energy dissipation for various values of the time constant with respect to the width of the pulse τ is shown in Figure 2.12 [1].

A simulated example of a full adder circuit showed that the method reduced power by 42% [1]. The glitch-free circuit had gates that have been slowed down to 20% of their original speed but with very little reduction in overall speed of the circuit.

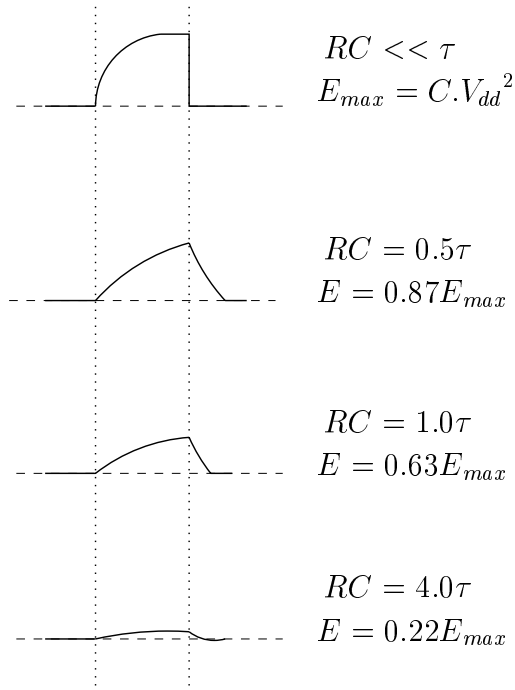

Figure 2.12: Effect of gate delay on energy dissipation.

2.3.4 Gate sizing

Gate sizing is defined as assigning load drive capabilities to the gates of a Boolean network, such that a given delay limit is obeyed, and the total cost in terms of power consumption is minimal [6]. Similar to the hazard filtering technique, this method also uses the filtering effect of the gates but achieves the increase in inertial delay by resizing the gates in the circuit. The delay model frequently used for solving this problem as given in [6]:

$$\tau_{gate} = \tau_{int} + c\ C_{load} \tag{2.8}$$

$$C_{load} = C_{wire} + C_{in} \tag{2.9}$$

where

- τ_{gate}: delay of the gate
- τ_{int}: internal delay of the gate

- C_{load}: capacitive loading experienced by the gate
- c: a constant
- C_{wire}: routing capacitance of the output net of the gate
- C_{in}: capacitive loading due to gates in the fanout cone

Berkelaar and Jacobs [7] use a parameter called the *speed constant* to reformulate the delay model but this converted the problem into the non-linear domain. Berkelaar *et al.* [6] have tried to solve the near-LP problem with the use of a piecewise linear simulator and have published results for all ISCAS benchmark circuits. This method was found to be faster than the LP method for circuits with less than 1000 gates but was not very useful for larger circuits. Berkelaar and Jacobs [7] have tried to formulate the gate sizing problem as an LP problem but the non-linear structure of their delay model posed problems in finding the global optimum solution. The details of this are dealt with in the next chapter.

Berkelaar and Jacobs [8] have tried gate sizing under a statistical delay model [4, 5] but the resulting problem was a non-linear problem that took the solver considerable time and resources to solve. This method too could not be used for circuits with more than 1000 gates.

2.3.5 Transistor sizing

Transistor sizing [9, 22] is similar to gate sizing but the essential difference is that in gate sizing all the transistors of a gate are sized equally but in transistor sizing particular transistors can be targeted and made to have different sizes. Traditionally, transistor sizing is done to reduce the area and delay of a VLSI chip [55].

Datta *et al.* [19] have considered transistor sizing for low-power and high-performance static CMOS circuits. The transistors on the critical paths of the circuit are sized to obtain

a better power and delay performance. To improve the switching speed and the output transition characteristics of a particular circuit block on the critical path, one may seek to increase the widths of the transistors in the block. This results in an increased current drive and better output transition time. Faster input/output transition times implies lower rush-through current, hence smaller short-circuit power dissipation. It is to be noted, however, that even though the delay of a particular block and its succeeding block are reduced, an increase in transistor widths increases the capacitive loading of the preceding block and may severely affect its power and delay. Thus, the issues regarding delay and power dissipation are fairly interlinked [50]. The algorithm described [19] minimizes the delay, the area and the power dissipation of a circuit by optimizing the sizes of the gates on the critical paths of the circuit. However, the constraint set for this model becomes non-linear and hence the solution of large circuits becomes tedious and complex.

2.3.6 Linear programming (LP) approach

A linear program determines a set of variables such that an objective is minimized under given constraints [23]. To eliminate the glitching power from a circuit the inertial delay of the gate has to be altered as dictated by a hazard filtering technique. But the altering has to be done without affecting the critical path of the circuit and also taking into account the change in delay of the gates in the fanout cone whose delay will effect the delay of the gate in question. There can be infinite number of solutions but the we are only interested in the global optimum for obvious reasons. Hence, the problem now becomes an optimization problem. The linear programming model guarantees the global optimum for every feasible solution of the problem.

A variety of LP models have been investigated to express the glitch removal problem (see chapter 3). Agrawal *et al.* [2] have described a gate level model to express the problem. The variables consist of inertial delays of gates in the circuit and also delays of buffers that may

need to be inserted in the circuit for complete glitch removal. The constraints are written by path enumeration from the gate inputs to the primary inputs (PIs). The details of this model are dealt with, in the next chapter.

2.4 Techniques for power estimation

Power estimation is the problem of estimating the average and peak power dissipation in a digital circuit [43, 47]. Accurate and efficient power estimation during the design phase is required in order to meet the power specifications of the chip. Hence the power estimation is done at various levels of design to satisfy this constraint.

We focus first on the calculation of dynamic power as the other two components depend on technology styles and are also of lesser orders of magnitude compared to the dynamic power dissipation. As seen earlier the dynamic power dissipation can be calculated from [59]:

$$P_{dyn} = \frac{1}{2}\, C_{load}\, V_{dd}^2\, f\, D \tag{2.10}$$

where

- P_{dyn}: dynamic power dissipation in gate
- C_{load}: load capacity of the gate
- f: clock frequency
- V_{dd}: supply voltage
- D: transition density or activity of the output of the gate

Transition density of a gate x is the average number of transitions per second at gate x and is denoted by $D(x)$. For a given chip all the other quantities except the transition density are fixed. It is the estimation of transition density that has resulted in various techniques

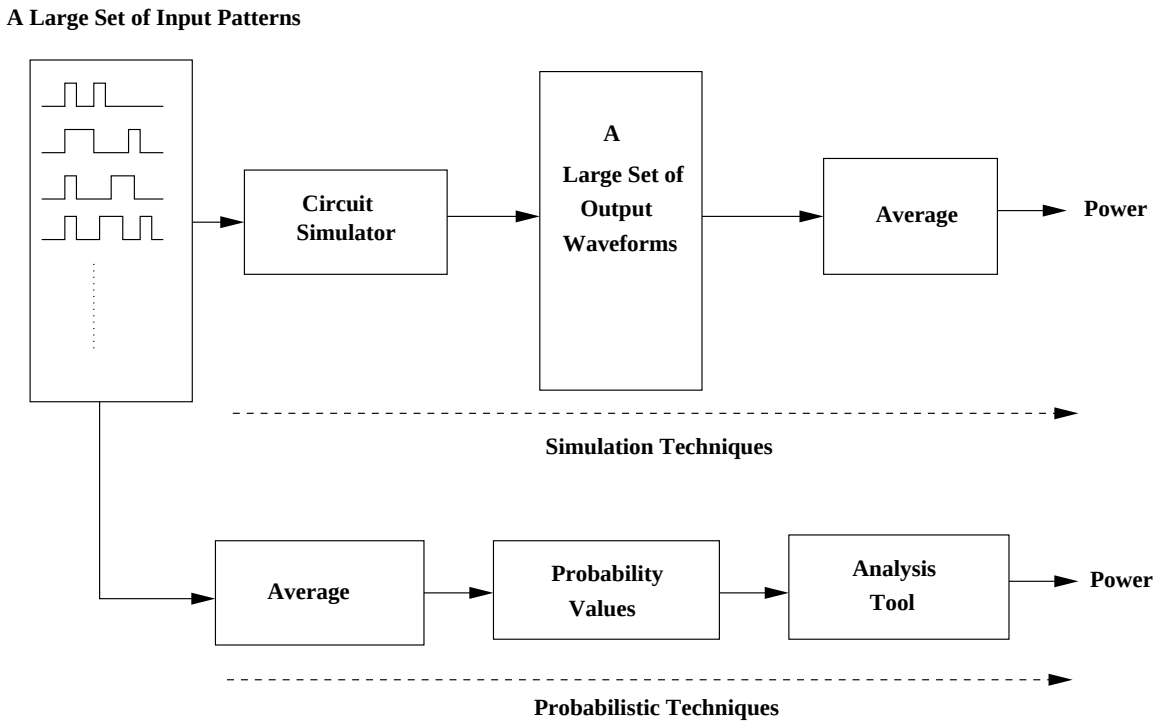

Figure 2.13: Simulation and probabilistic techniques of power estimation.

of power estimation. This section tries to familiarize the reader with a few of these major techniques.

A lot of work has been done recently on the estimation and reduction of leakage power in a circuit and the inquisitive reader is directed to the literature [18, 29, 34].

2.4.1 Simulation-based techniques

The simplest and most direct power estimation can be done by circuit simulation and monitoring the power supply current waveform as shown in Figure 2.13. The transition density can be directly gotten by keeping track of the overall transitions of each gate during simulation and can easily be averaged out over the set of input patterns. The advantages of this technique lie in its accuracy and generality. It can be used to estimate the power of any circuit, technology or design style. But the simulation results are directly related to the

input patters given to the PIs of the chip. Thus, this method is known as *strongly pattern dependent* [30, 63]. This disadvantage is very serious as most of the estimation is done in early design stages when the input patterns cannot be determined as yet. Hence, the power estimate would be erroneous as some of the patterns for which the estimate has been done may never occur in the circuit. Hence, the problem is finding a set of input patterns that are typical of the circuit during its normal mode of operation. If these typical patterns cannot be assessed in a circuit the designer would have to turn to more complex techniques of estimation which follow.

There are circuit-level simulator power estimators available in the market. The most important ones are SPICE and PowerMill. SPICE [41] operates by solving a large matrix of nodal currents using Kirchhoff's current law (KCL). It accurately takes into account non-linear capacitances, which cannot be captured by high-level tools [3, 21, 64]. Also, it can accurately measure short-circuit and leakage currents. Power measurement in SPICE is not direct. However, accurate methods for power measurement using SPICE have been reported [53]. It is a pattern dependent power analysis tool.

The main drawback of SPICE is that it cannot be used for large circuits or chips due to the time-consuming nature of the simulator. PowerMill [20] applies an event-driven simulation algorithm to increase computational speed by two or three orders of magnitude over SPICE. Also, it uses a table lookup to determine the terminal current of the device [3]. PowerMill can also identify the *hot spots* (areas that consume more dynamic power) and *trouble spots* (which consume unexpectedly large amounts of leakage power). This allows the designer to resize the circuit to reduce the rise/fall time.

2.4.2 Probabilistic techniques

These techniques focus on modeling the transitions occurring at a gate as a probability function. It is possible practically to predict the probability of the transitions occurring at

the inputs of the circuit. Given these, the transition density for each gate can be calculated by analyzing the topology of the circuit. The final power estimate can be got from summing up the individual power estimates of each gates with their transition probability used as the transition density. The main issues for probabilistic techniques are determining to what extent these signals are independent and also the spatial and temporal correlation of signals. For a very detailed analysis of the issues the reader is referred to the literature [40, 50].

In one of the earliest attempts at probabilistic estimation of power, Cirit [17] modeled the gate as a combination of a resistor and a capacitor and estimated the probability of a current flowing to charge or discharge the capacitor, i.e., of a transition occurring at the gate. A *zero delay model* assumes that all the gates in the circuit contain zero delay. This model has been used [17] to estimate the transition probability of the gates by a single run through the circuit. The simplest way to propagate transition probabilities is to work with the gate-level description of the circuit. If

$$y = AND(x_1, x2) \tag{2.11}$$

then

$$P_s(y) = P_s(x_1)P_s(x_2) \tag{2.12}$$

provided that x_1 and x_2 are independent. Similar expressions can be derived for other types of gates. The disadvantage of this method is that temporal as well as spatial independence was assumed.

Najm *et al.* [44] describe a technique called CREST whose input is in the form of a *probability waveform* at the PIs, which is a timing wave representing the estimated rise and fall times of signals at a gate. This probability waveform is then propagated through the circuit with the signals added at required gates. This method is more accurate than the previous one as the probabilities at the PIs are not lumped into a single variable but are allowed to vary and the actual averaging is done at the gates. The disadvantage, however, is it cannot predict the waveforms accurately.

Najm described another program DENSIM [42] that propagates the transition density values from the PIs to the circuit. He used the concept of *Boolean difference* to propagate the transition from level to level. As an example, consider $y = AND(x_1, x_2)$. In this case:

$$\frac{\partial y}{\partial x_1} = x_2 \tag{2.13}$$

$$\frac{\partial y}{\partial x_2} = x_1 \tag{2.14}$$

Hence, the transition density of output y can be given by:

$$D(y) = P(x_2)D(x_1) + P(x_1)D(x_2) \tag{2.15}$$

where

- $D(x)$: transition density of signal x
- $P(x)$: transition probability of signal x.

The calculation of Boolean differences gets complicated as we advance toward the *primary outputs* (POs) of the circuit but the primary advantage is that the transition densities can be computed more accurately by a single traversal of the circuit.

2.4.3 Statistical techniques

The main disadvantage of the probabilistic techniques is that we need to know the transition probability (or density as in DENSIM) of the PIs with reasonable accuracy. Since this is not very easy to predict quite often we revert to statistical techniques for the power estimation. The idea behind this technique is very simple and appealing: simulate the circuit repeatedly, using some logic simulator while monitoring the power. Eventually the power will converge to the average power. The issues are how to select the input patterns and when can we be assured that the power has converged. For this various statistical mean estimation techniques are employed, thus making this a *Monte Carlo* method.

Burch *et al.* [10] describe a program called McPOWER, which does the first known Monte Carlo approach for power estimation. McPOWER is a variable delay simulator which measures the power for various random sets of input patterns to the circuit and from that data estimates the average. The stopping criterion is when the mean is stable for a very large set of input patterns. This assumption is valid as the independence of the inputs is assured as they are selected randomly. McPOWER is slower than DENSIM by about 50% but is 500% more accurate than it. The major advantage of this approach was its ease of handling feedback circuits which could turn out to be a nightmare for probabilistic techniques.

Xakellis and Najm [62] have improved upon the McPOWER to come up with a *mean estimator of density* (MED), which estimates the transition densities of individual nodes statistically. The error tolerance levels can be specified by the user up-front and the nodes with least tolerance can also be identified. This speeds up convergence and estimates of power are done with desired confidence levels. The reason why one might want individual transition densities is to diagnose a high-power problem and, if possible, find a solution for it. The results show 99% accurate estimates for all benchmark circuits but the time taken is in hours. The accuracy can be traded off for simulation time in this method.

However, the major disadvantage of the statistical simulation technique is the neglect of the sequential and the higher order temporal correlations. Several suggestions have been made to improve their consideration by mixing application pattern simulation with statistical stopping criteria [45]. In addition they provide features which speed up simulation by separating the problem of current analysis from the problem of activity analysis. A detailed description of these is beyond the scope of this report.

2.5 Summary

This chapter has dealt with the various advances in low power design. First we looked at the various types of power dissipation in CMOS circuits and then gave an introduction

to the main problem of hazard or glitch power. Techniques have been described to lower the power consumption of a circuit by scaling the supply voltage, introducing buffers in the circuits to prevent glitching, increasing the inertial delay of a gate or by sizing the transistors accordingly and also have introduced programs to solve the models (LP). Then we surveyed various power estimation techniques.

When VLSI circuits are tested by non-functional vectors, generated either randomly or to cover specific target faults, the circuit activity and power dissipation can become so high that it can damage the device. Technique to control power dissipation during test, not discussed in this chapter, may be found in a recent book [46].

Chapter 3
Background on Linear Programming Techniques

In this chapter we present a detailed overview of two prior techniques that are similar to our approach. Both techniques, as well as that presented in the present work, seek an exact solution of the problem. We have included the previous techniques to highlight their disadvantages in designing circuits of large size and to further emphasize the superiority of the new method, which will be described in the next chapter.

3.1 Definitions

Since the concepts of delay and hazard have been explained in previous chapter we refresh only some of the major definitions for the benefit of the reader. The following definitions are helpful for understanding what follows:

- *Inertial delay* is the time taken by the gate output signal transition to occur after the input change causing the transition has occurred.
- *Differential delay* is the difference in signal arrival times at the inputs of a gate with more than one fanin.
- *Path balancing* is the technique of equalizing the delays of paths so that the differential delay at gate inputs is 0. This prevents the hazard transitions from being generated at the gate outputs.

- *Hazard filtering* [1] is the technique of increasing the inertial delay to be greater than the maximum differential delay at the gate inputs. This suppresses the generation of hazards at the gate outputs.

- *Gate sizing* is defined as assigning load drive capabilities to the gates of a Boolean network, such that a given delay limit is obeyed, and the total power consumption is minimal [6]. Similar to the hazard filtering technique, this method also uses the filtering effect of the gates but achieves the increase in inertial delay by resizing the gates in the circuit.

3.2 Overview of linear programming (LP) approaches

In this section we present two LP approaches for power optimization, which inspired this work. Both techniques are essentially the same, even though they are both applied at different levels of design. The method of Agrawal *et al.* is applied at the gate level whereas that of Berkelaar *et al.* is applied at the transistor level.

3.2.1 Berkelaar *et al.*'s method

To understand the LP in detail we need to look at the mathematical model of the delay of a gate expressed as [7]:

$$t_p = t_{int} + c\ \frac{C_{wire} + \sum S_i Cin, i}{S_{cell}} \tag{3.1}$$

where

- t_p: gate propagation delay

- t_{int}: gate internal delay

- C_{wire}: wire capacitance driven by the cell

- $C_{in,i}$: input capacitance of the gates in the fanout cone
- S_i: speed constant of cell i in the fanout cone
- S_{cell}: speed constant of the cell
- c: a constant

When a gate is sized, both the internal capacitance and the resistance of the cell increase. The gate delay due to internal capacitances remains constant during sizing. The input capacitance of the gate is affected by the resizing and hence sizing of a gate in fanout cone of some gate i affects the delay of gate i. This is modeled by including the speed factors of all the gates in the fanout in the delay equation. The wiring capacitance, however, is unaffected during sizing. The total load capacitance, consisting of wiring and fanout, is divided by the cell speed factor. This speed factor is the factor by which the drive capability of the gate is scaled.

A glitch (also called a hazard pulse) occurs when the differential delay at the PIs is greater than the propagation delay t_p of the gate. A linear program has been formulated as follows.

3.2.1.1 Objective function

We minimize the overall power but power is directly proportional to the gate load capacitance. Hence, we minimize the overall load capacitance from the expression:

$$\min : \Sigma_{i \in all\ gates} C_i S_i \tag{3.2}$$

3.2.1.2 Constraints

The constraints for solving the LP are:

$$T_i - T_o \leq t_p \tag{3.3}$$

$$t_p = t_{int} + c\ \frac{C_{wire} + \sum S_i Cin, i}{S_{cell}} \tag{3.4}$$

$$1 \leq S_i \leq limit \tag{3.5}$$

where

- T_i: total delay at the input i of gate from PIs
- T_o: total delay at the output i of gate from PIs
- t_p: gate propagation delay
- t_{int}: gate internal delay
- *limit*: maximum speed constant allowed for a gate

We see that the constraints for the formulation are almost linear except the expression for t_p. The non-linearity of this constraint poses problems for solving the model. For specific instances the formulation could be infeasible when the differential delays are too large. The number of constraints also increases exponentially with the number of gates in the input circuit as T_i has as many values as the paths in the circuit. Hence, this model is impractical for the optimization of larger circuits and also since it demands more complex non-linear solvers, it increases the optimization time, too.

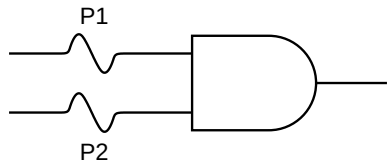

Figure 3.1: A gate illustrating paths from PIs.

3.2.2 Agrawal *et al.*'s method

Agrawal *et al.* [1, 2] described a *minimum transient energy* (MTE) design of digital circuits. A circuit is said to have MTE design if for any set of arbitrary simultaneous changes at PIs, steady state is reached with no more than one signal transition at each gate output. Consider the gate in Figure 3.1. Let $P1$ be the set of different paths leading from input 1 of the gate to the PIs and let $P2$ be the set of paths from input 2 to the PIs. Now the MTE condition can be ensured only if the delay difference in any pair of $P1$-$P2$ paths is not greater than the inertial delay (d) of the gate. Since the LP constraints have gate delays as the parameters, we need to write a constraint for every pair of paths from sets $P1$ and $P2$ for every gate as:

$$|\sum delay\ of\ gates\ on\ a\ path\ in\ P1 - \sum delay\ of\ gates\ on\ a\ path\ in\ P2| \leq d$$

The paper [2] describes a combination of delay balancing and hazard filtering for ensuring the circuit to be a MTE design. MTE design can always be guaranteed, even when the circuit is constrained with a maximum overall delay, provided that delay buffers are inserted. We will illustrate the LP formulation with a 1-bit adder shown in Figure 3.2. The circuit has 9 gates, which potentially consume energy.

3.2.2.1 Variables

Every gate i has a delay variable x_i. In addition, delay buffers with delays x_j are assumed at all PIs and fanouts. Hence, for the adder shown in Figure 3.2 we have:

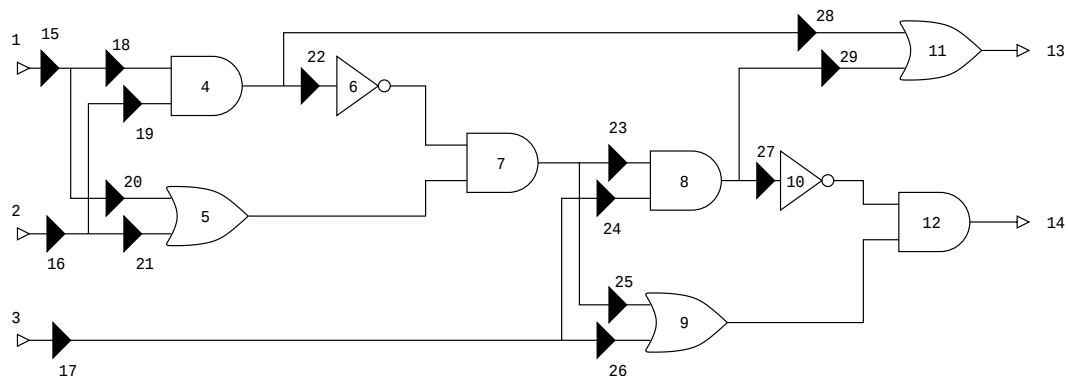

Figure 3.2: A 1-bit adder.

- Gate delays: x_i for $4 \leq i \leq 12$

- Buffer and PI delays: x_j for $15 \leq j \leq 29$

Buffers are shown as black triangular elements in Figure 3.2. PI delays are represented by buffers 15, 16 and 17. The PO signals 13 and 14 do not require any buffers since they do not fan out. Their delays are represented by the gates producing them, namely, gates 11 and 12.

3.2.2.2 Objective function

The main idea is to minimize the power dissipation of the circuit but this can be ensured by the conditions that follow. The insertion of buffers does ensure the elimination of glitches in the circuit but will increase the power consumption a bit, due to the extra gate in the circuit. Hence, it is our aim to eliminate glitches with the introduction of least number of buffers.

$$Objective : \min \sum_{15 \leq j \leq 29} x_j \tag{3.6}$$

3.2.2.3 Constraints

There are three types of constraints described.

- Initial constraints

  The lower bound is set for every parameter of the gate using this set of constraints. We write constraints for every gate as $x_i \geq 1$ and for every buffer as $x_j \geq 0$.

- Path constraints

  For every path leading to a gate we write constraints for the differential delay at the inputs and make sure that it is no greater than the inertial delay of the gate. For example for gate 5:

$$x_{15} + x_{20} - (x_{16} + x_{21}) \leq x_5$$
$$x_{16} + x_{21} - (x_{15} + x_{20}) \leq x_5$$

  The difference in the sum of delays along the path from gate 5 to the PI 1 and the sum of delays in the path from gate 5 to PI 2 must be less than the inertial delay of gate 5. Each pair of paths provides two constraints and thus every gate has as many constraints as there are path pairs leading from its inputs to the PIs.

- Maximum delay constraints

  This is to ensure that the solver does not insert buffers in the critical path of the circuit. The speed of a circuit is determined by the delay along the critical path and increasing it would increase the overall delay of the circuit. We set the maximum delay *(maxdelay)* of a path as a constraint:

$$x_{15} + x_{18} + x_4 + x_{22} + x_6 + x_7 + x_{23} + x_8 + x_{29} + x_{11} \leq maxdelay \tag{3.7}$$

  Similarly, every path between PIs and POs provides an inequality of the above type to the constraint set.

The LP model is then solved using AMPL [23] and the results have also been presented by the authors for small circuits. But we see that the constraints for every gate are proportional to the number of paths that traverse it and, hence, are exponential with respect to the size of the circuit. The number of constraints solvable by the solver is limited and hence large circuits cannot be solved using this model. This inspired us to build a model that makes solving large circuits feasible but still has the same efficiency and accuracy of this model.

3.3 Summary

In this chapter we have described the two LP optimizing approaches for MTE design, *viz.*, gate sizing and hazard filtering. In gate sizing [7] we discussed the mathematical modeling of the delay and the condition for elimination of glitches in the circuit. The shortcoming of the model was the non-linearity of a few constraints which increased the complexity and decreased the accuracy of the solution. In the method of Agrawal *et al.* [1, 2] we vividly described the set of variables and the generation of constraints. This method, too, was shown to be effective only for small circuits since it involves path enumeration, and it becomes infeasible for larger circuits. These have been the inspiration for us to build a new LP model that would not need path enumeration but would be of linear complexity and yet preserve the efficiency and accuracy of these two models.

Chapter 4
New Linear Programming Model

This chapter deals with the essence of the contribution of this work. We describe the problem we have tackled and also the new model that we propose for applying this method to larger circuits. A few examples are used to vividly illustrate the new technique.

4.1 Problem statement

The unwanted transitions in a digital circuit are called *hazards*, which result in wasted power. These hazards are a result of the unbalanced arrivals at the inputs of a digital gate. The hazards can be eliminated through the hazard filtering technique [1], which adjusts the inertial delays of all gates in the circuit to ensure that the arriving signals differ in time from each other, by no more than the inertial delay of that gate. A *linear programming* (LP) model has been used to find the optimal solution [2]. The models described in the prior papers generate constraints of the LP model through path enumeration. Since the number of paths in a circuit grows exponentially with the size of the circuit, so does the number of constraints. This limits the use of the technique to small circuits. In this paper we propose a new LP model with the number of constraints that grow linearly with the size of the circuit and hence the technique can be applied to larger circuits.

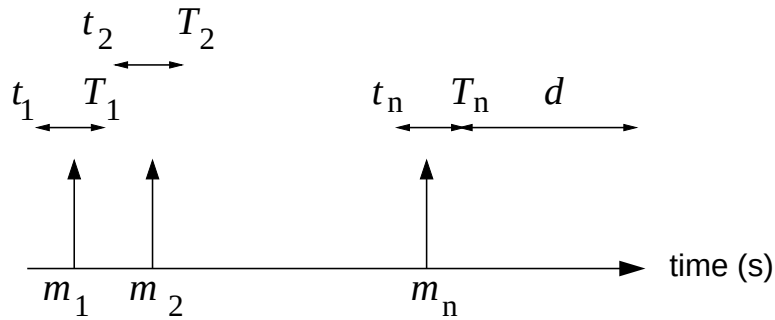

Figure 4.1: An arbitrary distribution of events at the inputs of a gate.

4.2 The concept of a timing window per gate

Instead of having a single parameter for a gate as was done earlier [1, 2], we introduce new variables for every gate, one for earliest time and the other for the most delayed time of arrival of signal at the output of a gate. The difference of these variables is a *timing window* within which the various signals arrive at the gate. Consider a gate i with n inputs. We define a variable T_i as the maximum time instant at which an event can occur at the output of the gate after the occurrence of an event at the PIs of the circuit. Similarly, t_i is the minimum time instant at which an event can occur at the output of the gate. This means that events always occur in the interval $[t_i,T_i]$ at the output of the gate i.

Theorem 1: Consider a gate i with n inputs, receiving events arriving from fanin gates $1,\ 2, \ldots n$ at times $m_1,\ m_2,\ \ldots m_n$. Assuming $T_1 \leq T_2 \leq ... \leq T_n$ and $t_1 \leq t_2 \leq ... \leq t_n$ are the arrival time parameters for the fanin gates, the number of events at the output of gate i cannot exceed

$$\min\{n, 1 + \lfloor \frac{T_n - t_1}{d_i} \rfloor\} \tag{4.1}$$

where d_i is the inertial delay of gate i.

Proof: We consider two cases for the gate i with n inputs.

- *First upper bound.* The maximum number of events cannot be greater than the maximum number of possible events at the input, which is n in this case (the number of fanins).

- *Second upper bound.* At the inputs of gate i, n events can be arbitrarily placed in time. Without loss of generality, we order them as $1, 2, \ldots n$ in Figure 4.1. From the definition of *ideal delay* [2] an output event can occur only if the separation between successive events is greater than d_i. The largest window in which the events can occur is $[t_1, T_n + d_i]$, and the number of events is given by

$$\lfloor \frac{T_n - t_1 + d_i}{d_i} \rfloor = 1 + \lfloor \frac{T_n - t_1}{d_i} \rfloor \tag{4.2}$$

Combining both upper bounds we get Equation 4.1. ■

From Theorem 1, the number of events takes the *least possible value* (the condition for minimum dynamic power) when

$$T_n - t_1 \leq d_i$$

According to Equation 4.1, the number of events at the output of the gate will not exceed 1. Also, since

$$\begin{aligned} T_i &= T_n + d_i \\ t_i &= t_1 + d_i \end{aligned}$$

Hence, the condition for MTE can be written as:

$$d_i \geq T_i - t_i \tag{4.3}$$

4.3 Linear programming

A linear program determines a set of variables such that an objective is minimized under the given constraints. We illustrate the linear programming model with the example of the adder circuit shown in Figure 3.2. Buffers are inserted at the PIs and at each fanout branch of a signal (PI or gate) with more fanouts than 1. The linear program is developed as follows.

4.3.1 Variables

The variables can be split into two categories:

- Gate variables
- Buffer variables

The gate variables are a set of three variables for each gate in the circuit. They are:

- T_i for every gate i: This is the maximum time at which the output of this gate can produce an event after the occurence of an event at the PIs.
- t_i for every gate i: This is the minimum time at which the output of this gate can produce an event after the occurence of an event at the PIs.
- d_i for every gate i: This is the inertial delay of the gate which has to be given as an output of the optimizer.

The buffer variables also have the same set of parameters as gate variables but are treated differently later in the program.

4.3.2 Objective function

The injection of buffers into the circuit increases the area of the circuit and hence the obvious objective would be to reduce the number of them. Hence, our objective function would be

to reduce the delay of the buffers. The real objective would be to achieve a balanced delay circuit with the minimum number of buffers possible and the sum of the buffer delays would be equally effective in achieving this objective.

4.3.3 Constraints

4.3.3.1 Initial constraints

The lower bound is set for every parameter of the gate using these set of constraints. We write the constraints
$d_i \geq 1$ for every gate i, $d_i \geq 0$ for every buffer i, $T_i \geq 0$ for every gate and buffer i, and $t_i \geq 0$ for every gate and buffer i.

4.3.3.2 Gate constraints

First let us deal with the constraints for a gate with a single fanout. This set of constraints include the buffers, too. Consider the buffer 19 in Figure 3.2. Its fanin is buffer 16. Hence its set of constraints would be:

$$\begin{aligned} T_{16} + d_{19} &= T_{19}; \\ t_{16} + d_{19} &= t_{19}; \end{aligned}$$

These constraints are self-explanatory as the maximum and minimum delay at the input of the gate would just be added to the delay of the gate (or buffer) as it proceeds to the fanout. Now consider the case where there are more than 1 fanins, as in gate 7. Then we have:

$$\begin{aligned} T_7 &\geq T_5 + d_7; \\ T_7 &\geq T_6 + d_7; \\ t_7 &\leq t_5 + d_7; \\ t_7 &\leq t_6 + d_7; \end{aligned}$$

$$d_7 \quad \geq \quad T_7 - t_7;$$

The first four constraints ensure that the parameter T_7 settles at the value that is the maximum (T_5, T_6) and t_7 would settle at the minimum (t_5, t_6). The condition for MTE is ensured by the last constraint.

4.3.3.3 Overall circuit delay constraints

To ensure that the delay balancing does not slowdown the circuit beyond the specified limit we use a given upper bound on the maximum delay at the output. This can be ensured by placing that upper bound on parameter T of all gates feeding the primary outputs of the circuit. Thus we have additional constraints as:

$$T_{11} \leq maxdelay \quad \text{and} \quad T_{12} \leq maxdelay$$

where *maxdelay* is specified according to the application of the device and the amount of speed the user is willing to sacrifice for power savings. It is a user-defined parameter.

4.4 Validation of the model

In this section, we validate the new LP model by showing its equivalence to the exact path constraint model [2]. Consider the block of circuitry shown in the Figure 4.2. This gate consists of three gates and delay buffers (shown as black triangles) on the three primary inputs and two fanout lines.

First let us generate the constraints using the old model described by Agrawal *et al.* [2]. The variables here are the gate and buffer delays $d_1 \cdots d_8$. Path constraints are required only for multi-input gates.

- For gate 6:

$$d_1 + d_3 - d_2 \quad \leq \quad d_6 \tag{4.4}$$

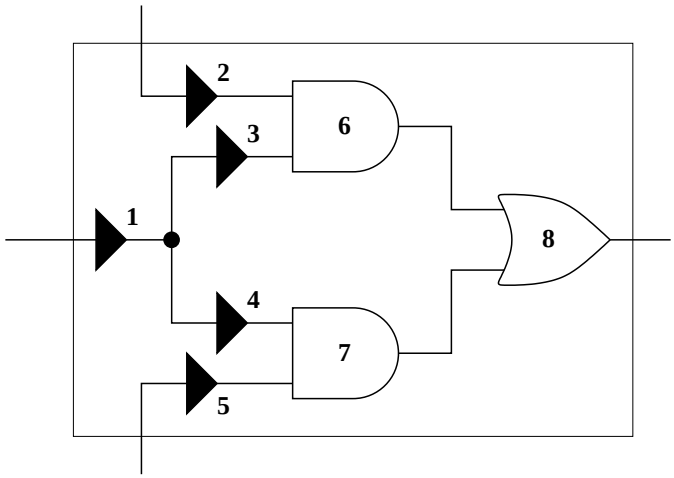

Figure 4.2: A combinational circuit block.

$$d_2 - d_1 - d_3 \leq d_6 \tag{4.5}$$

The two inequalities are needed for each pair of paths to ensure that the absolute value of the differential delay does not exceed the gate delay.

- For gate 7:

$$d_1 + d_4 - d_5 \leq d_7 \tag{4.6}$$

$$d_5 - d_1 - d_4 \leq d_7 \tag{4.7}$$

- For gate 8: Since there are two possible paths to every input of gate 8 we have four pairs of paths, giving rise to eight path inequalities. These can be easily constructed and, therefore, we have avoided listing them.

All of these constraints are based on paths between the gate and PIs. Similarly, the overall circuit delay constraints are based on all paths through the circuit. The constraints defined in Section 4.3, on the other hand, do not depend on paths. They are derived using three gate variables, T_i, t_i and d_i. Using the example of gate 6 in Figure 4.2 we will show

that the above path constraints are identical to and can be derived from the constraints of Section 4.3 and, hence, the new LP will produce the same solution.

For gate 6 in Figure 4.2, we write:

$$t_6 \leq t_2 + d_6 \tag{4.8}$$

$$t_6 \leq t_3 + d_6 \tag{4.9}$$

$$T_6 \geq T_2 + d_6 \tag{4.10}$$

$$T_6 \geq T_3 + d_6 \tag{4.11}$$

Also, the minimum transient energy (MTE) condition of Inequality 4.3 requires

$$T_6 - t_6 \leq d_6 \tag{4.12}$$

For buffers 1, 2 and 3, we have:

$$t_1 = T_1 = d_1,\ t_2 = T_2 = d_2,\ \text{and } t_3 = T_3 = d_1 + d_3 \tag{4.13}$$

Substituting in Inequalities 4.8 through 4.11 from 4.13, we obtain:

$$t_6 - d_2 \leq d_6 \tag{4.14}$$

$$t_6 - d_1 - d_3 \leq d_6 \tag{4.15}$$

$$d_6 \leq T_6 - d_2 \tag{4.16}$$

$$d_6 \leq T_6 - d_1 - d_3 \tag{4.17}$$

Adding Inequalities 4.14 and 4.17, we obtain

$$d_1 + d_3 - d_2 \leq T_6 - t_6 \tag{4.18}$$

Adding Inequalities 4.15 and 4.16, we obtain

$$d_2 - d_1 - d_3 \leq T_6 - t_6 \tag{4.19}$$

Finally, if we substitute Inequality 4.12 in 4.18 and 4.19 we obtain the path constraints of Inequalities 4.4 and 4.5. A similar procedure can be applied recursively to gates farther away from PIs to derive the path constraints from the local constraints of the new LP constraint set of Section 4.3. This applies to the output gate constraints specifying the overall circuit delay. Thus the equivalence of the present LP formulation and the previous method [2] is established.

4.5 Why is this model superior?

The strength of this model is in reducing the unnecessary constraints at each gate. The inertial delay constraint for hazard filtering is not written for all paths to the inputs as done previously but it isolates the maximum and minimum possible time instants (T and t, respectively) at which a signal can arrive at the input. Since we are interested only in the maximum differential delay we have just one constraint specifying that condition after the maximum T and minimum t have been isolated.

The superiority of the model can be easily visualized by looking at the example in the Figure 4.3. Let us consider the blocks such that each block is the circuit shown in Figure 4.2. As we saw in the example the number of constraints for the first block using the prior method would be a constant multiple of 4 (since it has four paths through it. Note that we ignore the initialization constraints as they will be similar for both methods.) If two blocks are cascaded then the circuit would require at least 4×4 constraints as there are as many paths

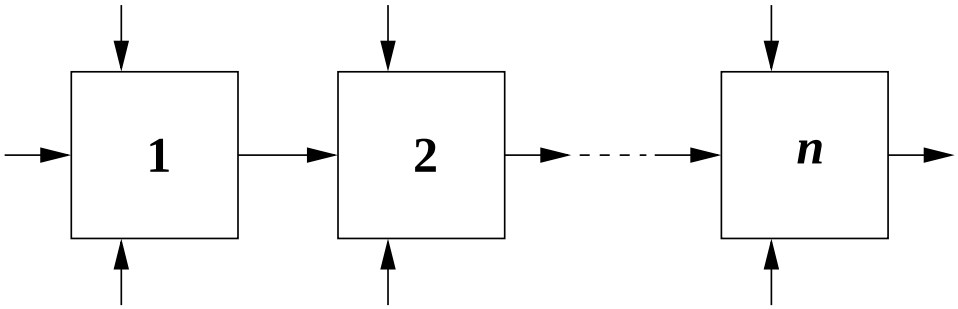

Figure 4.3: A combinational array circuit using the block of Figure 4.2.

leading to the last gate. Hence, we conclude that a circuit with n such blocks cascaded would require at least 4^n constraints.

Using the new method the constraints for a single block would be a multiple of 15 (five constraints per gate and we have three gates here). Now if two blocks are cascaded together the number of constraints would become 15×2 as it would just increase the number of gates as far as the model is concerned. Hence, a circuit with n cascaded blocks would have $15n$ constraints which is linear and clearly much less than the previous model.

To further illustrate the point we give the graph in Figure 4.4 with the number of constraints for the ISCAS'85 benchmark circuits. As seen in the figure the benchmark circuit c880 needed 6.9 million constraints with the old model but only 3,611 constraints with new model. The graph shows a linear increase with the number of gates for the new model. Though not obvious in the figure, the graph for the old model has an exponential rise and the constraint set could not be completed for some of the larger circuits (e.g., c6288 and c7552) due to the memory limits of the computer.

4.6 Summary

In this chapter we have described the new LP model for MTE design of circuits. We have explained the concept of the timing window and illustrated the formulation with vivid examples. The chapter also shows the validity of the model by unfolding the constraints to

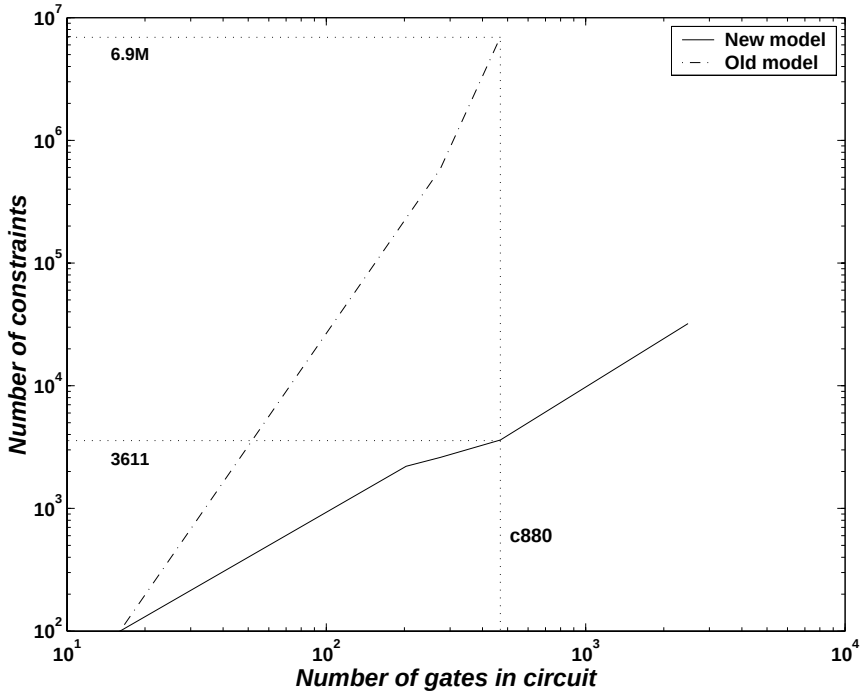

Figure 4.4: Number of constraints for the new LP model and the old (path enumeration) model.

reveal the constraints described for the previous path enumeration model of Chapter 3. The new LP model is shown to be linear in complexity with the size of the circuit and hence is much simpler to solve and also feasible for large circuits as well. Thus we have eliminated the shortcomings of the previous model and have achieved a significant improvement in the power optimization procedure.

Chapter 5
Results

This chapter discusses the various experiments that we performed with this model, in order to highlight its uses and advantages. First we describe the conditions that we have set for ourselves and the assumptions in computing the results. Then we move on to the actual results obtained.

5.1 Experimental conditions and procedure

5.1.1 Procedure

For better understanding purposes, we have chronicled the experimental procedure from the time a file is given to us in a hardware description language to the time we generate the MTE design of the circuit and compute the power savings.

5.1.1.1 Step 1 – Constraint generation

The optimization has been done at the logic level of design. We have used a C++ program to:

- parse the circuit in the *rutmod* (a hardware description language) format given as input. This gate level description consists of for each gate, a number, type and a fanin list. It can be substituted by any other similar type of description. If the description is hierarchical, it is flattened.

- Insert buffers at all primary inputs and in all fanout signals. These buffers are used as place holders for any possible delays to be inserted, if necessary. Initially, they all have zero delays.

- generate the constraint file.

5.1.1.2 Step 2 – Optimization

This output file of Step 1 is in the AMPL [23] format and hence is given as input to the solver. We also specify a *maxdelay* for that particular circuit. Then the solver gives an optimized set of parameters for all gates and also the buffers, if needed.

5.1.1.3 Step 3 – Analysis of results

Using the set of delays given by the solver a new netlist is generated. Now the power estimation is done to determine the effectiveness of the optimization. We used a power estimator [30] as the basis for comparison. A series of analyses have been done for the variance of maximum delay and the size of the circuit. The results will be explained in the next section.

5.2 Experimental results

Consider the 1-bit adder circuit in Figure 3.2. The LP model for this circuit in the AMPL format contained 94 gate constraints and 42 initial constraints. Since the minimum input-output delay is 6 units (assuming all gates of unit delay), there cannot be a solution for $maxdelay < 6$. For other values of maxdelay, we present the results in Table 5.1.

As shown in the Table when $maxdelay = 6$ is specified, we require two buffers. For $maxdelay = 7$ only one buffer is needed and for $maxdelay \geq 11$ no buffer is needed. In all cases, the condition of inequality in Equation 4.3 was satisfied at all multi-input gates

Table 5.1: Results from AMPL [23] for the 1-bit adder circuit.

	Gate and buffer delays											No. of
maxdelay	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{17}	d_{28}	Buffers
6.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	2.0	1.0	2.0	1.0	2
6.5	1.0	1.0	1.0	1.0	1.0	2.0	1.0	2.5	1.0	2.0	0.5	2
7.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	2.0	0.0	1
8.0	1.0	1.0	1.0	1.0	1.5	2.5	1.0	3.5	1.5	1.5	0.0	1
9.0	1.0	1.0	1.0	1.0	2.0	3.0	1.0	4.0	2.0	1.0	0.0	1
10.0	1.0	1.0	1.0	1.0	2.5	3.5	1.0	4.5	2.5	0.5	0.0	1
≥ 11.0	1.0	1.0	1.0	1.0	3.0	4.0	1.0	5.0	3.0	0.0	0.0	0

by optimally combining delay balancing and hazard filtering to meet the overall *maxdelay* requirement.

Notice that the linear objective function minimizes the sum of buffer delays instead of the number of buffers. The linear program does not always find the minimum buffer solution. However, as the *maxdelay* is increased, the solutions at which the number of buffers just drops to a lower value are optimal. This is because some larger delay values may also satisfy the constraints.

5.3 Analysis of results

To estimate the power consumption of the circuit we used a variable delay simulator that counts the number of transitions every gate undergoes for every vector transition occurring at the PIs of the circuit. We have conducted the simulations for the unit gate delay model and the model with gate delays from the optimization results of the LP. The results for the number of transitions for all possible transitions at the PIs are shown in Figures 5.1 and 5.2. The graphs in these figures are to be read as follows. The x-axis of the graph shows the input vector pair in the octal format. Thus, 00 means a pair of binary vectors (000,000) applied to the 1-bit adder circuit of Figure 3.2. Similarly, the last pair 77 means vectors (111,111). All pairs with identical vectors produce no transitions in either circuit.

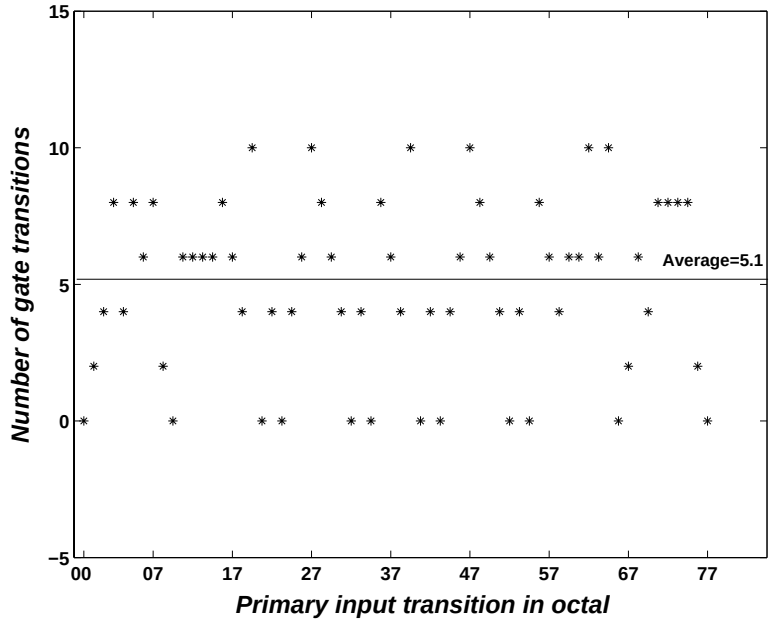

Figure 5.1: Number of transitions for all vector-pairs in 1-bit adder with unit-delays.

An event-driven logic simulator was used to determine the number of transitions produced by each of the 64 possible vector pairs.

The results for the unit-delay (unoptimized) circuit are given in Figure 5.1. An example of maximum power dissipation is the vector pair (010,001) or 21 (octal), which produced 10 transitions. The average number of transitions is 5.1. Since there are only 9 gates (excluding PIs and POs) at most 9 transitions can occur for a vector pair. This means that there are glitches in the unit delay circuit.

The optimized circuit delays correspond to the last row in Table 5.1. This design contains no buffers. We see that the maximum transitions in the circuit for any input transition is 6. One example is vector pair (000,011) or 03 (octal). The average number of transitions for the optimized circuit is thus reduced to 4.01.

If we assume that all vector pairs are equally likely to occur and that the dynamic

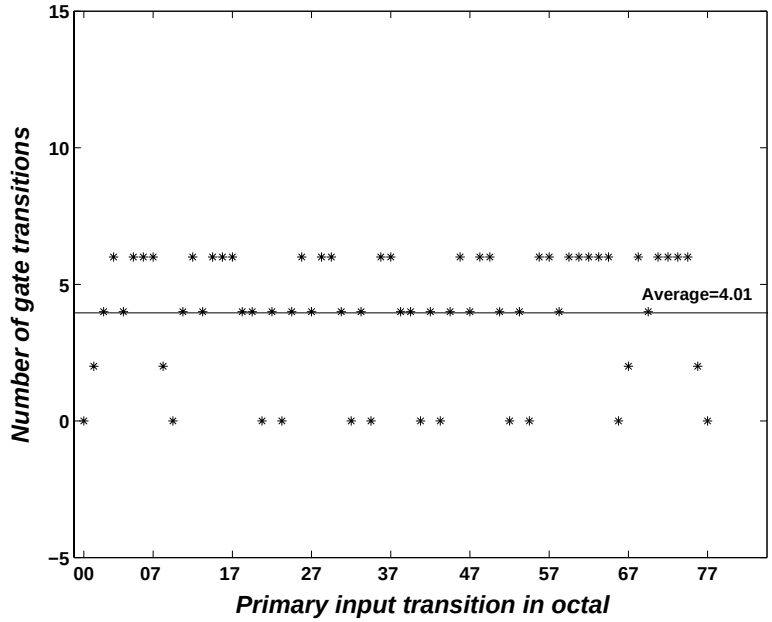

Figure 5.2: Number of transitions for all vector-pairs in 1-bit adder with optimized delays.

power consumption is proportional to the number of gate transitions, then the optimization produces a peak power savings of 40% and average power savings of 20%.

We applied the similar procedure to a larger circuit, a 4-bit ALU with 80 gates which required 25,000 constraints for the older method. The new model needed only 989 constraints and the results, which are exactly the same as those obtained by path enumeration [2], are shown in Table 5.2.

Table 5.2: Results from AMPL [23] for the 4-bit ALU circuit.

maxdelay	Buffers
7.0	5
10.0	2
12.0	1
≥ 15.0	0

The minimum input-output delay (with all unit gate delays) is 7 and hence there is

no solution for $maxdelay < 7$. We need 5 buffers when $maxdelay \geq 7$ and the no buffer solution needs the $maxdelay$ to be increased to 15. The power savings were also computed with an event-driven simulator for 10,000 random vector transitions and achieved a peak power savings of 33% and average power savings of 20%. The method is thus very effective even for large circuits.

5.4 Summary

In this chapter we have presented the results obtained on the 1-bit adder and 4-bit ALU circuits. We have also analysed the power optimization after the delays have been incorporated into the circuit. We have proved that the model works and it does reduce the number of transitions occurring in the circuit.

Chapter 6
Conclusion and Future Work

6.1 Conclusion

The new linear prgramming model has been shown to be an effective tool for the removal of glitches in any digital circuit. This is an effective tool for designing the minimum transient energy digital CMOS circuits. The main contribution is the reduction in complexity of the linear programs to be solved along with the reduction in the constraint set. The constraint set of the new linear program is a linear function of the circuit size. This is a significant simplification over the previous methods that required an exponentially growing constraint set. Results have shown circuit design examples with up to 20% reduction in average power by using this method. The reduction in the peak power can be much higher.

We obtain a reduction in the size of the LP constraint set by introducing two arrival time variables at gates. In the present application it is a novel idea. However, arrival time variables have been used to develop linear-time static timing analyzers [27, 28], which would otherwise have the problem of the exponential path complexity.

6.2 Future work

This work can be advanced in four possible directions.

1. *Transistor sizing.* The present work has enhanced the gate-level optimization method of Agrawal *et al.* [1, 2]. Similar extensions are possible for the transistor-level optimization

method of Berkelaar *et al.* [7].

2. *Cell-based design.* Application-specific integrated circuits (ASICs) are often designed using a pre-designed library of standard cells. Such cells can be easily optimized for low-power using the method presented. However, a circuit designed with those cells will have interconnect delays that must be accounted for. This will require a hierarchical application of the optimization procedure. First, the cells can be optimized so all internal glitches are suppressed for specified maximum input signal delay skew. Then the interconnect delays must be controlled to conform to that delay skew at the input of each cell.

3. *Low-power asynchronous circuits.* Proper operation of an asynchronous circuit requires that the combinational logic does not produce glitches in the feedback signals [57]. The present method has almost a direct application to the design of such circuits. If glitches are eliminated from all gates (including the outputs of the combinational logic) then power dissipation will be minimized and a proper asynchronous function can be guaranteed as well.

4. *Extension to sequential circuits.* Glitch removal in sequential circuits is much more complicated than doing the same in combinational circuits. The method would require careful timing analysis of the arrival times of the signals at gates. Since our method keeps track of the arrival times of signals, it might be useful for hazard filtering of sequential circuits.

References

[1] V. D. Agrawal, "Low Power Design by Hazard Filtering," in *Proc. of the International Conference on VLSI Design*, Jan. 1997, pp. 193–197.

[2] V. D. Agrawal, M. L. Bushnell, G. Parthasarathy, and R. Ramadoss, "Digital Circuit Design for Mimimum Transient Energy and Linear Programming Method," in *Proc. of the International Conference on VLSI Design*, Jan. 1999, pp. 434–439.

[3] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design.* Boston: Kluwer Academic Publishers, 1995.

[4] M. Berkelaar, "Statistical Delay Calculation," in *Workshop notes of the International Workshop on Logic Synthesis*, (Lake Tahoe, CA), May 1997.

[5] M. Berkelaar, "Statistical Delay Calculation: A Linear Time Method," in *Proc. of the IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, (Mierlo, The Netherlands), Dec. 1997, pp. 15–24.

[6] M. Berkelaar, P. Buurman, and J. Jess, "Computing Entire Area/Power Consumption versus Delay Trade-off Curve for Gate Sizing Using a Piecewise Linear Simulator," *IEEE Transactions on Circuits and Systems*, vol. 15, no. 11, pp. 1424–1434, Nov. 1996.

[7] M. Berkelaar and E. Jacobs, "Using Gate Sizing to Reduce Glitch Power," in *Proc. of the-ProRISC Workshop on Circuits, Systems and Signal Processing*, (Mierlo, The Netherlands), Nov. 1996, pp. 183–188.

[8] M. Berkelaar and E. T. A. F. Jacobs, "Gate Sizing Using a Statistical Delay Model," in *Proc. of the Design Automation and Test in Europe Conference*, (Paris, France), Mar. 2000, pp. 283–290.

[9] M. Berkelaar and J. A. G. Jess, "Transistor Sizing in MOS Digital Circuits with Linear Programming," in *Proc. of the European Design Automation Conference*, (Mierlo, The Netherlands), Mar. 1990, pp. 217–221.

[10] R. Burch, F. A. Najm, P. Yang, and T. Trick, "McPOWER: A Monte Carlo Approach to Power Estimation," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1992, pp. 90–97.

[11] T. D. Burd, "Low-Power CMOS Library Design Methodology," Technical report, U. of California, Electronics Research Laboratory, Berkeley, California, 1994. MS Thesis, Dept. of EECS.

[12] W. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave Pipelining: A Tutorial and Survey of Recent Research," *IEEE Transactions on VLSI Systems*, vol. 6, no. 3, pp. 464–474, Sept. 1998.

[13] A. Chandrakasan and R. Brodersen, editors, *Low-Power CMOS Design.* New York: IEEE Press, 1998.

[14] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design.* Boston: Kluwer Academic Publishers, 1995.

[15] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "HYPER-LP: A System for Power Optimization Using Architectural Transformations," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1992, pp. 300–303.

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

[17] M. A. Cirit, "Estimating Dynamic Power Consumption of CMOS Circuits," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1987, pp. 534–537.

[18] D. Duarte and Y. F. Tsai and N. Vijaykrishnan and M. J. Irwin, "Evaluating Run-Time Techniques for Leakage Power Reduction," in *Proc. of the International Conference on VLSI Design*, Jan. 2002, pp. 31–38.

[19] S. Datta, S. Nag, and K. Roy, "ASAP: A Transistor Sizing Tool for Area, Delay and Power Optimization of CMOS Circuits," in *Proc. of the International Symposium on Circuits and Systems*, May 1994, pp. 61–64.

[20] A. Deng, "Power Analysis for CMOS/BiCMOS Circuits," in *Proc. of the International Workshop on Low-Power Design*, Apr. 1994, pp. 3–8.

[21] M. S. Elrabaa, I. S. Abu-Khater, and M. I. Elmasry, *Advanced Low-Power Digital Circuit Techniques.* Boston: Kluwer Academic Publishers, 1997.

[22] J. P. Fishburn and A. E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," in *Proc. IEEE International Conf. Computer-Aided Design*, Nov. 1985, pp. 326–328.

[23] R. Fourer, D. M. Gay, and B. M. Kernighan, *AMPL: A Modeling Language for Mathematical Programming.* South San Francisco, California: The Scientific Press, 1993.

[24] T. Gabara, "Pulsed Low Power CMOS," *International Journal of High Speed Electronics Systems*, vol. 5, no. 2, pp. 159–177, June 1994.

[25] L. Geppert and T. S. Perry, "Transmeta's Magic Show," *IEEE Spectrum*, vol. 37, no. 5, pp. 26–33, May 2000.

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

[27] R. B. Hitchcock Sr., "Timing Verification and the Timing Analysis Program," in *Proc. of the 19th Design Automation Conf.*, June 1982, pp. 594–604.

[28] R. B. Hitchcock Sr., G. L. Smith, and D. C. Cheng, "Timing Analysis of Computer Hardware," *IBM Journal of Research & Development*, vol. 26, no. 1, pp. 100–105, Jan. 1982.

[29] W. Jiang, V. Tiwari, E. D. la Iglesia, and A. Sinha, "Topological Analysis for Leakage Prediction of Digital Circuits," in *Proc. of the International Conference on VLSI Design*, Jan. 2002, pp. 39–44.

[30] S. M. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits," *IEEE Journal of Solid-State Circuits*, vol. 21, no. 5, pp. 889–891, Oct. 1986.

[31] H. Kawaguchi, K. Nose, and T. Sakurai, "A Super Cut-off CMOS Scheme for 0.5 V Supply Voltage with Pico-Ampere Stand-By Current," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 10, pp. 1498–1501, Oct. 2000.

[32] T. Kim, W. Burleson, and M. Ciesielski, "Logic Restructuring for Wave Pipelined Circuits," in *Workshop notes of the International Workshop on Logic Synthesis*, Jan. 1993.

[33] T. S. Kim, W. Burleson, and M. Ciesielski, "Delay Buffer Insertion for Wave Pipelined Circuits," in *Workshop notes of the International Workshop on Logic and Architecture Synthesis*, Dec. 1993.

[34] R. Kumar and C. P. Ravikumar, "Leakage Power Estimation for Deep-Submicron Circuits in an ASIC Design Environment," in *Proc. of the International Conference on VLSI Design*, Jan. 2002, pp. 45–50.

[35] J. B. Kuo and J. Lou, *Low-Voltage CMOS VLSI Circuits.* New York: Wiley Interscience Publications, 1999.

[36] G. Lakshminarayana and N. K. Jha, "FACT: A Framework for the Application of Throughput and Power-Optimizing Transformations to Control-Flow Intensive Behavioral Descriptions," in *Proc. of the Design Automation Conference*, June 1998, pp. 102–107.

[37] G. Lakshminarayana and N. K. Jha, "Synthesis of power-optimized and area-optimized circuits from hierarchical behavioral descriptions," in *Proc. of the Design Automation Conference*, June 1998, pp. 439–444.

[38] G. Lakshminarayana, A. Raghunathan, N. K. Jha, and S. Dey, "Power Management in High-Level Synthesis," in *Proc. of the International Conference on VLSI Design*, Jan. 1998, pp. 24–29.

[39] C. Lemonds and S. S. Shetti, "A Low Power 16 by 16 Multiplier Using Transitions Reduction Circuitry," in *Proc. of the International Workshop on Low Power Design*, Apr. 1994, pp. 139–142.

[40] J. Monteiro and S. Devadas, *Computer-Aided Design Techniques for Low Power Sequential Logic Circuits.* Boston: Kluwer Academic Publishers, 1997.

[41] L. W. Nagel, "SPICE2, A Computer Program to Simulate Semiconductor Circuits," Technical Report ERL Memorandum ERL-M520, U. of California, Electronics Research Laboratory, Berkeley, California, May 1975. PhD Dissertation, Dept. of EECS.

[42] F. A. Najm, "Transition Density: a New Measure of Activity in Digital Circuits," in *Proc. of the International Conference on Computer-Aided Design*, Feb. 1993, pp. 310–323.

[43] F. A. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[44] F. A. Najm, R. Burch, P. Yang, and I. Hajj, "CREST - A Current Estimator for CMOS Circuits," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1988, pp. 204–207.

[45] W. Nebel and J. Mermet, *Low Power Design in Deep Submicron Electronics.* Boston: Kluwer Academic Publishers, 1997.

[46] N. Nicolici and B. M. Al-Hashimi, *Power-Constrained Testing of VLSI Circuits.* Boston: Kluwer Academic Publishers, 2002.

[47] M. Pedram, "Power Estimation and Optimization at Logic Level," *International Journal of High Speed Electronics and Systems*, vol. 5, no. 2, pp. 179–202, June 1994.

[48] D. L. Raatz and G. E. Sobelman, "Digital Signal Processor with Delay-Evaluation Array Multipliers and Low-Power Memory Addressing." United States Patent No. 5,333,119, July 26, 1994.

[49] J. M. Rabaey and M. Pedram, editors, *Low Power Design Methodologies.* Boston: Kluwer Academic Publishers, 1996.

[50] K. Roy and S. C. Prasad, *Low-Power CMOS VLSI Circuit Design.* New York: Wiley Interscience Publication, 2000.

[51] W. A. Serdijn, editor, *Low-Voltage Low-Power Analog Integrated Circuits.* Boston: Kluwer Academic Publishers, 1995.

[52] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1992, pp. 402–407.

[53] M. Shoji, *CMOS Digital Circuit Technology.* Upper Saddle River, New Jersey: Prentice Hall, 1988.

[54] E. S. Sinencio and A. G. Andreou, editors, *Low-Voltage/Low-Power Integrated Circuits and Systems.* New York: IEEE Press, 1998.

[55] M. J. S. Smith, *Application-Specific Integrated Circuits.* Reading, MA: Addison-Wesley, 1997.

[56] G. E. Sobelman and D. L. Raatz, "Low Power Multiplier Design Using Delayed Evaluation," in *Proc. of the International Symposium on Circuits and Systems*, May 1995, pp. 1564–1567.

[57] S. H. Unger, *Asynchronous Sequential Switching Circuits.* New York: Wiley-Interscience, 1969.

[58] H. Veendrick, "Short-Circuit Dissipation of CMOS Circuitry and its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468–473, Aug. 1984.

[59] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Approach.* Reading, MA: Addison Wesley Publication, 1985.

[60] D. Wong, G. D. Micheli, and M. Flynn, "Inserting Active Delay Elements to achieve Wave Pipelining," in *Proc. of the International Conference on Computer-Aided Design*, Nov. 1989, pp. 270–273.

[61] D. Wong, G. D. Micheli, and M. Flynn, "Designing High Performance Digital Circuits Using Wave Pipelining: Algorithms and Practical Experiences," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 1, pp. 25–46, Jan. 1993.

[62] M. G. Xakellis and F. A. Najm, "Statistical Estimation of Switching Activity in Digital Circuits," in *Proc. of the Design Automation Conference*, June 1994, pp. 728–733.

[63] G. Y. Yacoub and W. H. Ku, "An Accurate Simulation Technique for Short Circuit Power Dissipation Based on Current Component Isolation," in *Proc. of the International Symposium on Circuits and Systems*, 1989, pp. 1157–1161.

[64] G. Yeap, *Practical Low Power Digital VLSI Design.* Boston: Kluwer Academic Publishers, 1998.