Hierarchical Optimization of Digital CMOS Circuits for Power, Performance and Reliability

A Dissertation
Presented to
The Academic Faculty

By

Yuvraj Singh Dhillon

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering

Georgia Institute of Technology
May, 2005

Hierarchical Optimization of Digital CMOS Circuits for Power, Performance and Reliability

Approved by:

Dr. Abhijit Chatterjee, Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jeffrey A. Davis
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. D. Scott Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Thomas G. Habetler
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Adit D. Singh
School of Electrical Engineering
Auburn University

Date Approved: April 15, 2005

This dissertation is dedicated to my parents, Jaspal and Rajwant Dhillon, my sister, Monica and my wife, Puneet

Acknowledgements

I am grateful to my advisor, Dr. Abhijit Chatterjee, for his support and guidance throughout my graduate career. He continuously motivated me to strive for my best and this helped me to improve the quality of my dissertation. I am thankful to Dr. Adit Singh for his keen interest in my research and his insightful comments/suggestions that helped me out of various dead-ends.

I thank Dr. Jeff Davis and Dr. Scott Wills for their suggestions during my proposal examination and afterwards that helped me to improve the work. I would like to thank Dr. Sean Lee for his informative and lively discussions about computer architecture, the “world” outside academia and the politics of conferences.

Thanks to Dr. Smaragdakis, Dr. Linda Wills, Dr. Doug Blough and Dr. Shomu Banerjee for their excellent teaching, which made taking courses so enjoyable.

My colleague, Utku Diril, has been an invaluable friend throughout my studies at Georgia Tech. He cheered me out of innumerable research lows and his cheerful spirits motivated me to keep striving for my goal.

I want to thank all my lab-mates for their friendship. They made my research life so much less tedious by their jokes, laughter and moral support.

I want to thank my parents, my sister and my wife for all the love and support they have provided me. I have reached up to this step only because of their blessings and their belief in me.

Finally, I want to thank God Almighty for always watching over me and guiding me in the right direction in my darkest hours.

Table of Contents

List of Tables

List of Figures

Glossary

BJT	Bi-polar junction transistor
CAD	Computer-aided design
CED	Concurrent error detection
CMOS	Complementary metal oxide semi-conductor
DAG	Directed acyclic graph
DSM	Deep sub-micron
ECC	Error correcting codes
EDP	Energy-delay product
IC	Integrated circuit
MOSFET	Metal oxide semi-conductor field effect transistor
PI	Primary input
PO	Primary output
SER	Soft-error rate
SPICE	General-purpose circuit simulation program
SRAM	Static random access memory
$\mathbf{V_{DD}}$	Supply voltage
$\mathbf{V_{th}}$	Threshold voltage
ZSA	Zero-slack algorithm

Summary

Power consumption and soft-error tolerance have become major constraints in the design of DSM CMOS circuits. With continued technology scaling, the impact of these parameters is expected to gain in significance. Furthermore, the design complexity continues to increase rapidly due to the tremendous increase in number of components (gates/transistors) on an IC every technology generation. This research describes an efficient and general CAD framework for the optimization of critical circuit characteristics such as power consumption and soft-error tolerance under delay constraints with supply/threshold voltages and/or gate sizes as variables.

A general technique called Delay-Assignment-Variation (DAV) based optimization was formulated for the delay-constrained optimization of directed acyclic graphs. Exact mathematical conditions on the supply and threshold voltages of circuit modules were developed that lead to minimum overall dynamic and static power consumption of the circuit under delay constraints. A DAV search based method was used to obtain the optimal supply and threshold voltages that minimized power consumption.

To handle the complexity of design of reliable, low-power circuits at the gate level, a hierarchical application of DAV based optimization was explored. The effectiveness of the hierarchical approach in reducing circuit power and unreliability, while being highly efficient is demonstrated. The usage of the technique for improving upon already optimized designs is described. An accurate and efficient model for analyzing the soft-error tolerance of CMOS circuits is also developed.

Chapter I

Introduction

1.1. Motivation

Energy consumption is recognized as one of the most important parameters in the design of modern digital systems [1]. With the scaling of transistor feature sizes, more and more transistors can be made to fit in a given area of die. Furthermore, the clock frequencies also have been scaling up with every technology generation. The combination of increased transistor densities and increased clock frequencies has been causing the dynamic power consumption/dissipation density (W/cm^2) to increase very rapidly even though it has been slightly offset by the scaling down of supply voltages and node capacitances [2]. The high power density causes problems with chip packaging, increase in cooling costs, and reduced chip reliability. Even though the transistor counts per unit area have been going up, designers are putting more and more functionality into ICs. This causes die sizes to go up, for example, by 25% every micro-processor generation. Combined with the increasing power density, the increase in area leads to a roughly 75% increase in total chip dynamic power every technology generation, leading to correspondingly increased energy expenses for consumers. The scaling of technology features has also led to an increase in the leakage energy per transistor. This is because threshold voltages have been reducing to compensate for the decrease in transistor switching speeds due to reduced supply voltages. The reduction in threshold voltage causes an exponential increase in the transistor leakage current [3] which means that the

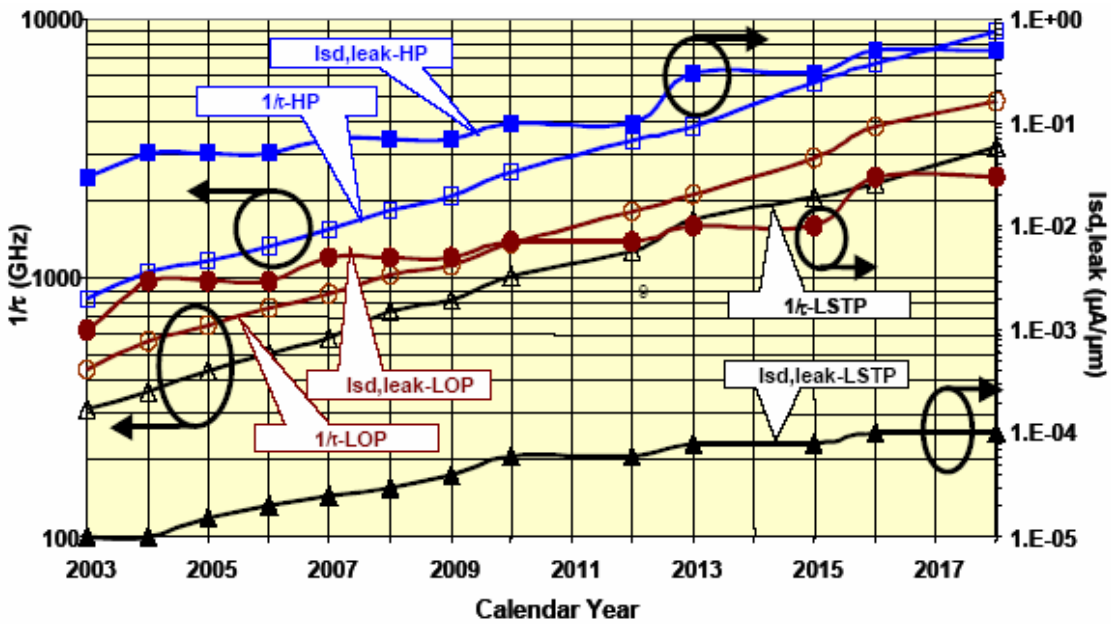


Figure 1.1. Scaling trends of transistor performance (1/τ) and sub-threshold leakage current ($I_{sd,leak}$) for high-performance (HP), low-operating-power (LOP), and low-standby-power (LSTP) logic [1].

chip wastes a lot of energy even when in standby mode. This has crucial implications in mobile applications which are battery-powered because the battery gets used up even when the device is not being operated.

Figure 1.1 shows the trends for performance and leakage current of transistors in three different kinds of applications viz. high-performance (HP), low-operating-power (LOP) and low-standby-power (LSTP). The transistor performance is characterized by the metric CV/I, where C is the gate input capacitance, V is the supply voltage and I is the drive current. HP chips use low V_{DD}s to reduce dynamic power and very low V_{th}s to maintain high-performance. They, therefore, have the highest leakage power. LOP chips are typically for relatively high-performance mobile applications, such as notebook computers, where the battery is likely to be high capacity and the focus is on reduced operating power. LSTP chips are typically for lower performance consumer type

applications, such as consumer cell-phones, with lower battery capacity and an emphasis on the lowest possible static power dissipation, i.e., the lowest possible leakage current.

Another side-effect of technology scaling that is rapidly coming to the forefront is the increased susceptibility of logic and memory circuits to soft-errors caused due to alpha particles in the chip material and cosmic rays from space [1][4]. Because of the reduced node capacitances, a smaller injected charge is needed to induce a glitch at a circuit node. Thus, low-energy particle strikes that earlier had no effect on a circuit can now cause soft-errors. Because of the reduced supply voltages, noise margins are reduced, which also increases the susceptibility to particle strikes. Increasing clock frequencies increase the probability of a soft-error getting latched. Furthermore, due to super-pipelining, the number of gates in pipeline stages have been reducing, which in

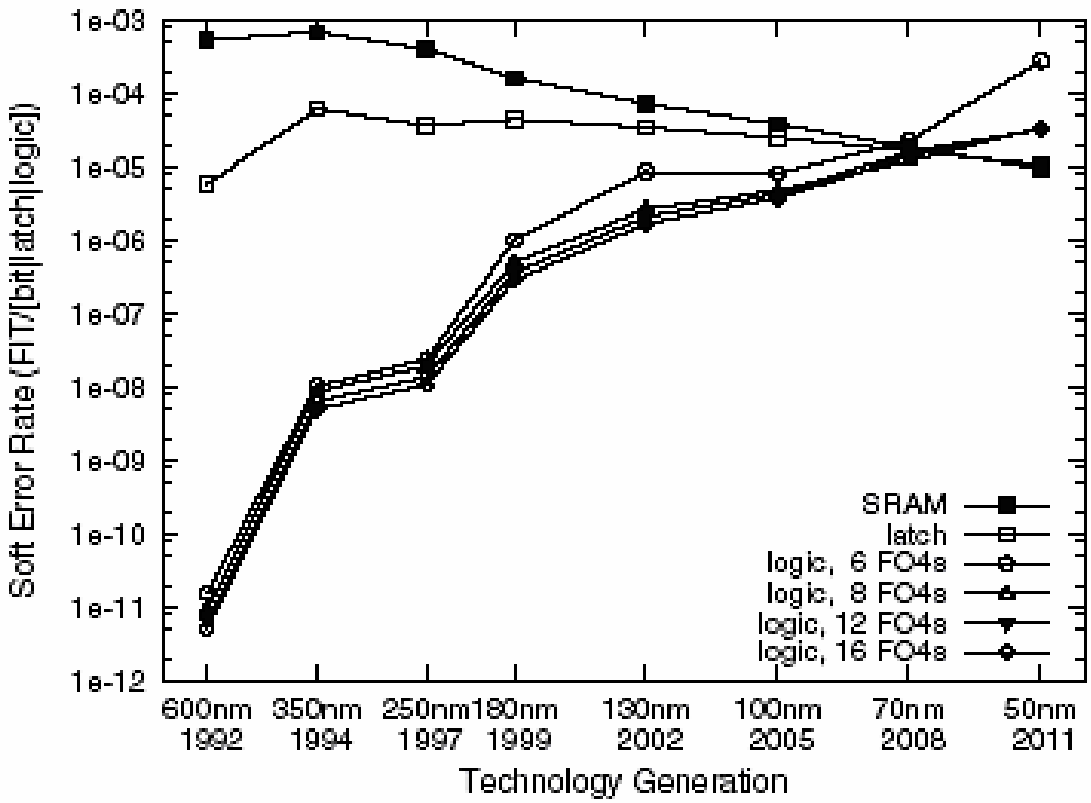


Figure 1.2. Future soft-error trends for memories, latches, and logic with different pipeline depths [4].

turn reduces the electrical attenuation of glitches as they propagate to the latches. Although the factors described above affect the power consumption and soft-error tolerance of both memory and combinational elements, this work focuses on combinational elements only. Figure 1.2 shows the predicted SER by technology and pipeline depths. The x-axis plots the CMOS technology generation and the y-axis plots the SER (in FIT or failures in time) for each element on a log scale. 1 FIT is equal to 1 error in 10^9 hours. The SER of a single SRAM cell declines gradually with decreasing device size, while the SER of a latch stays relatively constant. The SER for a single logic chain shows the most significant change – increasing over five orders of magnitude from 600nm to 50nm. The effect of super-pipelining is illustrated by the increasing SER for logic circuits at higher pipeline depths (smaller clock period in FO4 delays) within each technology generation.

The power consumption of a circuit is the product of the supply voltage and the current drawn by the circuit from the supply voltage. Thus, the power consumption can be reduced by reducing the supply voltage and/or the current drawn. The supply voltage can only be reduced to a certain extent beyond which the circuit delay becomes unacceptable. One way to reduce the current drawn from the supply voltage is by developing technologies in which transistors need less current for operation. This is seen in the shift of usage from BJT transistors to MOSFETs. Another way is by implementing the circuit using a logic style that draws less current. An example is the usage of CMOS logic style instead of the NMOS logic style to reduce the current drawn. The current drawn can also be reduced by appropriate transistor level design of the circuit. Given the technology, the logic style and the circuit design, there are various other optimization

techniques proposed in literature to reduce the power consumption at the circuit level. Since the major parameters that affect circuit power consumption are the supply voltage(s) (V_{DD}), threshold voltage(s) (V_{th}), circuit transistor sizes and switching activity; most techniques try to optimize one or more of these parameters. Optimization is necessary since changing the values of these parameters can adversely impact other circuit characteristics such as delay and area. This work focuses on reducing the power consumption of CMOS circuits by optimizing the supply voltages, threshold voltages and gate sizes simultaneously.

The soft-error tolerance of a circuit also depends on the technology, the logic style and the circuit design; and there have been various studies on the effect of these on the soft-error tolerance. However, there has been little investigation into usage of circuit parameter optimization to reduce soft-errors. It is stressed in [1] that such techniques will be urgently needed for mitigation of soft-errors in logic in future technology generations. This work shows that optimization can indeed significantly improve the attenuation of particle-hit induced glitches through CMOS combinational circuits, thereby increasing the soft-error tolerance.

Although circuit reliability refers to a host of issues besides soft-error tolerance such as electro-migration, thermal reliability and noise reliability (cross-talk, ground bounce, substrate coupling, etc); in this thesis, reliability of a circuit will be used to refer solely to its soft-error tolerance.

1.2. Organization of Thesis

Chapter 2 surveys commonly used techniques for low power and reliable digital design. Since this thesis focuses mainly on optimization at the logic level, technique at

this level will be given more attention. Structural/behavioral level techniques will also be described briefly for the sake of completeness. Chapter 3 first gives a new matrix representation of the topology (**T**) of a directed acyclic graph (DAG) and then describes the use of the new representation in the development of Delay-Assignment-Variation (DAV) based optimization, a generic technique for delay-constrained optimization of DAGs. In Chapter 4, the topology matrix representation, **T**, is used to derive exact conditions on the supply and threshold voltages of circuit modules that yield minimum energy consumption under delay constraints. Search based methods are then used to obtain the values of these voltages and cluster them into small groups that can be used in a practical implementation of the circuit. A metric is developed that can be used by designers to determine the energy efficiency of their designs at the module level. Experimental results of optimizing ISCAS'85 benchmarks are provided. In Chapter 5, the usage of the topology matrix representation for optimization of gate level netlists is explored. A hierarchical method is developed that is able to handle large netlists and provides energy reductions beyond those possible by other methods given in literature. Chapter 6 describes ASERTA (accurate soft-error tolerance analyzer), a tool for fast and accurate estimation of the soft-error tolerance of combinational circuits. The tool yields soft-error estimates close to those generated by SPICE in orders of magnitude less computation time. The energy optimization method developed in Chapter 5 is then extended to also optimize the soft-error tolerance of circuits. Finally, Chapter 7 concludes by summarizing the main contributions of the thesis and providing suggestions for future work. The applicability of the topology matrix representation for optimization at the behavioral and software levels is also discussed.

Chapter II

Survey of Low-Power, Reliable CMOS Circuit Optimization Techniques

2.1. Introduction

This chapter surveys techniques described in literature for the design of low-power and reliable CMOS circuits. Low power digital design requires optimization at all levels of the design hierarchy viz. system, structural/behavioral, logic, circuit, and device technology level. However, techniques at all levels basically boil down to a fundamental set of concepts: energy dissipation is reduced by lowering the supply voltage, the voltage swing, the physical capacitance, the switching activity or a combination of the above [5]. The application of these techniques may result in an increase in the delay and/or area of the system. Since the focus in this work is on low power CAD techniques at the logic level, design techniques at this level are surveyed in detail in Section 2.3. Design techniques at the behavioral level are also described briefly in Section 2.2, as these can impact the techniques at the logic level.

Soft-error tolerant design has mostly focused on memories and flip-flops since these are highly vulnerable. However, the focus has recently started shifting to combinational logic. In Section 2.4, traditional approaches to soft-error reduction at the combinational level are described as well as emerging approaches. Finally, Section 2.5 categorizes the work in this thesis according to the techniques discussed and compares it to previous approaches.

2.2. Low Power Structural/Behavioral Design Techniques

At the structural/behavioral level, a high-level specification of a problem is mapped into the register-transfer level. The high-level specification is typically in the form of a control-flow graph (CFG) and a data-flow graph (DFG) or a combination of the two (CDFG). The register-transfer level specifies the number of different types of hardware modules needed (allocation), what operations of the CDFG are mapped onto what modules (assignment) and at what time-step what operation is carried out (scheduling).

Power reduction can be achieved with the use of behavioral transformations that reduce the number of time-steps by increasing concurrency and through proper scheduling. The reduced time-steps allow usage of a slower clock, enabling the use of lower supply voltages. The quadratic reduction in power due to reduced supply voltage can often offset the increase in capacitance due to more concurrency. A number of other high-level transformations of the CDFG that reduce the number of operations, reduce the switching activity, increase utilization of modules, etc have been described in [6].

If a number of modules, with a range of power/delay costs, are available for allocation, then an appropriate mapping of modules to operations can lead to lower power consumption for the same performance. In [7], an Integer Linear Programming (ILP) approach is used to generate low-power schedules using an optimal set of supply voltages while accounting for level conversion costs. Approaches that takes up available slack in schedules in a power-aware way are presented in [8][9].

Dynamic Voltage Scaling (DVS) [10][11] is another approach that can yield large power savings while meeting performance constraints. The basic idea is to run modules at

a lower supply voltage (and hence lower speed) when the system is idle or has lower computing requirements and to increase the supply voltage (increase speed) when there is an increase in computing demand. The major issues in this technique are (i) whether to have software or hardware control of voltage scaling, (ii) accurate prediction of when the system is going to be idle and when it is going to be busy, (iii) the granularity of voltage changes allowed, and (iv) the frequency of voltage changes allowed. DVS can also be combined with dual/multiple threshold voltage usage to reduce leakage power as well. A low-power RISC processor built using hardware-based DVS and variable-threshold CMOS (VTCMOS) technology has been described in [12].

2.3. Low Power Logic Design Techniques

The power consumption of any CMOS logic circuit is composed of dynamic power, short-circuit power and leakage power. Although dynamic power has been the predominant component of total power in older technologies, the leakage power component is becoming significant due to the reduction of supply and threshold voltages in current deep sub-micron (DSM) technologies. Short-circuit power is a small fraction of the total power and its importance is not expected to grow with device scaling [13].

The dynamic power consumption of any CMOS logic circuit is given by:

$$P_{dyn} = 0.5 \cdot f_{CLK} \cdot V_{DD}^2 \cdot \sum_i \alpha_i \cdot C_i \tag{2.1}$$

where f_{CLK} is the clock frequency, V_{DD} is the supply voltage, α_i is the switching activity of the ith node in the circuit, and C_i is the load capacitance of the ith node in the circuit. Hence, dynamic power of the circuit can be reduced by reducing the supply voltage or by reducing the product of the switching activity and load capacitance (the switched

capacitance) at the circuit nodes. The clock frequency generally cannot be lowered as it is determined by the timing constraints.

Lowering the supply voltage is an effective mechanism for reducing dynamic power because of the quadratic dependence in Equation 2.1. However, lowering the supply voltage affects the delay of the logic gates. The delay of a CMOS logic gate is given by:

$$d = \frac{k \cdot C \cdot V_{DD}}{(V_{DD} - V_{th})^{\alpha}} \tag{2.2}$$

where k is a constant depending on the logic function computed by the gate, C is the output capacitance of the gate (includes the load capacitance as well as the parasitic junction capacitance), V_{DD} is the supply voltage, V_{th} is the threshold voltage, and α is the velocity saturation coefficient [14] which is between 1.2 and 1.5 for short-channel devices and 2 for long-channel devices. Thus, lowering V_{DD} leads to an increase in the delay.

The most common approach uses two or more supply voltages for reducing power. The gates on the critical path use the highest supply voltage to meet the timing constraint. Gates on the non-critical paths that have available slack can use lower supply voltages. There are two major problems facing this approach. The first is the need for generating and routing multiple supply voltages to different parts of the chip and the second is the need for level converters whenever a low supply voltage gate drives a high supply voltage gate. Level converters are needed to prevent the flow of large static current from supply to ground because of the PMOS transistor in a high supply voltage gate not being completely turned off by the lower voltage output by a preceding low

supply voltage gate. The first problem is limited by restricting the number of supply voltages used to two [15][16]. The area, delay, and power impact of the level converters is reduced by two ways. In Clustered Voltage Scaling (CVS) [17][18], it is tried to cluster the gates that operate at the reduced supply voltage so that the number of interfacing level-converters is reduced. The other approach [19] is to partition the gate level netlist into modules and use multiple supply voltages at the module level. The area, delay and energy penalties of the level converters are shown to be small compared to the modules. The idea of taking up slack can be straight-forwardly extended to the usage of gate sizing and dual/multiple V_{DD}s simultaneously [20].

The delay impact of reduced supply voltage can be decreased by reducing the threshold voltage as can be seen from Equation 2.2. This trend is being increasingly followed in current technologies where the supply and threshold voltages are lowered to reduce dynamic power as well as not impact delay significantly. The reduction in threshold voltage, however, has the adverse effect of increasing the leakage power per transistor exponentially as can be seen from the leakage power equation below:

$$P_{leak} = V_{DD} \cdot I_S \cdot e^{\frac{V_{GS} - V_{th}}{n \cdot v_T}} \cdot \left(1 - e^{\frac{V_{DS}}{v_T}} \right) \tag{2.3}$$

I_S is a circuit and process dependent constant, n is the sub-threshold swing coefficient, V_{DD} is the supply voltage, V_{th} is the threshold voltage and v_T is the thermal voltage.

Leakage energy is expected to equal, if not overtake, dynamic energy as the major component of total energy in future technology generations. To tackle this problem, researchers have proposed using two [21][22] or more [23] threshold voltages. Similar to the dual voltage approach, the idea here is to use the lower threshold voltage for gates on

the critical paths so that they can operate at high speed albeit with high leakage energy dissipation and to use the higher threshold voltage for gates on the non-critical paths so that they operate with less leakage.

The combined usage of dual/multiple V_{DD}s and dual V_{th}s has also been suggested [24][25]. [25] gives an exact method that achieves the upper bound on energy savings possible using multiple supply and threshold voltages. As a practical application of these techniques, an ARM processor has been designed using Clustered Voltage Scaling (CVS) and dual threshold voltages in 0.18μ CMOS technology [26]. The problem of circuit sizing and dual threshold voltage assignment has been studied in [27]. Using multiple gate sizes to achieve low-power while meeting timing constraints is a standard optimization carried out by commercially available CAD tools and also studied in literature [28][29][30].

Methods have been proposed to simultaneously use multiple supply voltages, multiple threshold voltages and multiple gate sizes to reduce power [31][32][33]. These methods use heuristics to determine the supply voltage, threshold voltage and gate size to be used for every gate in a circuit such that the total power is minimized while the delay constraint is met. Rules of thumb for choosing the values of V_{DD}s, V_{th}s and sizes are given in [34].

The switched capacitance of a logic circuit is primarily dependent on the logic function implemented by the circuit. However, different circuits implementing the same function may have different internal switching activities and loads. For example, among various adder implementations, a ripple carry adder has the least average number of logic transitions while a conditional sum adder has the largest number, for the same set of

computations [35]. For arbitrary logic functions, a power-aware cost function can be used while doing logic minimization [36], resulting in lower switching activity.

The switched capacitance may be reduced by pre-computation [37] in which a subset of inputs is used to guess the final answer. If the guess is correct, the remaining inputs are disabled so as to reduce switching activity. Another approach is to “gate” the inputs to those logic blocks whose output will not be needed for the present computation, thus reducing switching activity in the blocks. This approach can be very useful when applied to clock signals [38], since clock signals are generally heavily loaded.

Switching activity can also be optimized at the logic level by appropriate technology decomposition and mapping. Technology decomposition is the problem of converting a Boolean network into a network of primitive gates (e.g. NOT and 2-input NAND) before mapping the network to a cell library. Even though the switched capacitance of the final circuit is unknown at this stage, it is good to have a decomposition that minimizes the sum of the switching activities. Technology mapping is the step in which sets of primitive gates are mapped to gates in some target cell library. The general approach is to hide nodes with high switching activity inside the gates where they drive smaller load capacitances [39]. As a final step of technology mapping, power aware signal to pin assignment can also reduce power by assigning high switching signals to input pins with low input capacitance. The pin permutation for low power should take place on non-critical gates as it generally interferes with pin permutation for minimum delay.

Path balancing is an approach which tries to minimize the level difference between the inputs of nodes which drive highly capacitive nodes. This reduces the

chances of the occurrence of glitches at the circuit inputs, which reduces the switching activity at the nodes output. Balanced paths can be achieved by inserting buffers in the shorter paths. The amount of power saved due to reduced glitching has to be weighed against the increase in capacitance due to the buffers. Another approach is to use smaller sized (and hence slower) gates on the shorter paths, which can reduce glitches as well as power consumption in the smaller gates [29].

In finite state machine synthesis, states with highest transition frequency to one another can be given uni-distance codes so as to minimize switching activity [40]. However, such a state assignment might increase the activity in the combinational logic of the state machine. This problem has been handled in [41].

This section listed the major techniques for low-power logic design, but this list is brief and far from complete. The interested reader is referred to [5][42][43][44][45] for more details and more techniques.

2.4. Soft-Error Tolerant Design Techniques

Soft-errors, also called transient errors or single-event upsets (SEUs), are errors in digital circuits that arise due to electrical noise or external radiation rather than design or manufacturing defects. The existence of this phenomenon has been known for a very long period and various studies have focused on protecting memory elements, particularly caches and latches/flip-flops, from soft-errors. The focus on caches has been due to the fact that caches occupy a large chunk of chip area and that the soft-error rate (SER) primarily depends on the area exposed to the environment. Latches have been also studied because they are very vulnerable to soft-errors; a bit flipped can have catastrophic effects. Logic circuitry has been much less susceptible to soft-errors because of

occupying less chip area and also because of three effects that mask soft-errors: (i) Logical masking, which refers to the case when a soft-error is stopped form propagating to a latch because of the path not being sensitized, (ii) Electrical Masking, which refers to the case when a soft-error just attenuates away before reaching a latch and (iii) Latching-window masking, which refers to the case when a soft-error arrives at a latch input when the latch is not accepting data.

However, recent studies [4][46] have shown that the SER of combinational logic is expected to rise with decreasing feature sizes, supply voltages and increasing clock rates. Decreasing feature sizes and supply voltages reduce the critical charge needed to cause a soft-error, thereby increasing the number of soft-errors generated and hence possibly latched. Increasing clock rates reduce the time when latches/flip-flops are not accepting data, thereby increasing the probability of errors getting latched. Furthermore, reducing logic depths (due to super-pipelining) reduce the attenuation of glitches before they reach the flip-flops. Therefore, it is becoming necessary to study techniques for hardening combinational logic against soft-errors.

In view of the above, research interest has been increasing in the development of techniques for reduction of soft-errors in combinational logic. The most common way has been to use concurrent error detection (CED) logic [47][48] that monitors the output of a logic block for the occurrence of an error. If an error is detected, the system can correct it and keep operating. However, these methods add a lot of area and power overhead to circuits. Furthermore, the speed of the circuit is also reduced because of the presence of the extra checking circuit. Low-cost methods for increasing soft-error tolerance of commodity applications using time-redundancy [49] and partial duplication [50] have

been proposed. However, these methods still add additional delay overhead to the original circuit due to the presence of the checker circuit. Also, these methods have system level overheads (such as pipeline flushes) when an error is detected, either to correct the error or to do the computation again.

Recently, there has been work in using gate sizing as a way to reduce soft-error rates. The key insight is that the amplitude and width of the voltage pulse (glitch) generated at a node due to a particle hit is a function of the capacitance at that node [51][52]. Hence, it is appropriate to increase the sizes of the gates loading nodes more susceptible to soft-errors. These techniques mostly use analytical models [53][54] to simulate glitch propagation from the point where the particle hit to the primary output, and then try to minimize some cost function that depends on gate sizes.

2.5. Research Categorization and Comparison

This research focuses on the problem of power and soft-error tolerance optimization at the logic level, specifically by using gate sizing, dual/multiple V_{DD}s and dual/multiple V_{th}s. The problem of determining the optimal V_{DD}s and V_{th}s that minimize total energy consumption (dynamic + static) at the module level is first tackled. An arbitrary interconnection of modules is assumed as compared to [24], which assumes node and edge-disjoint paths. Furthermore, an exact condition on the supply and threshold voltage values is developed as compared to [24], which uses rules of thumb.

The problem of determining optimal gate sizes, V_{DD}s and V_{th}s is tackled next. While [28][32] stop after taking up the circuit slack using some heuristics, the approach described in this thesis searches for the best way to take up slack. It is shown that taking up slack using heuristics is atleast 12% off from the optimal. In [33], a mixed-integer-

linear-programming (MILP) problem is solved repeatedly to determine the optimal circuit parameters. This approach is prohibitively expensive in terms of CPU time. On the other hand, this research develops a novel hierarchical optimization approach that can yield near optimal circuit parameters with approximately $O(N^3)$ worst case optimization complexity, where N is the problem size (number of gates).

Finally, this thesis presents a novel way to reduce soft-errors in combinational logic. A unique approach that uses circuit sizing, multiple V_{DD}s/V_{th}s and output loading to increase the attenuation of particle-hit induced glitches is developed. It is shown that this approach performs better than an approach that just uses gate sizing and it also performs better than an approach that uses higher supply voltage and output loading for reducing soft-errors.

Chapter III

Delay-Assignment-Variation (DAV) Based Optimization

3.1 Introduction

This chapter lays the mathematical groundwork for the rest of the thesis. A novel representation of the circuit topology is first developed. The representation is then used to develop a generic delay-constrained optimization framework. Although the discussion focuses on circuit netlists, it is applicable to any directed acyclic graph (DAG) based problem.

3.2 Topology Matrix Representation of Circuit Netlist

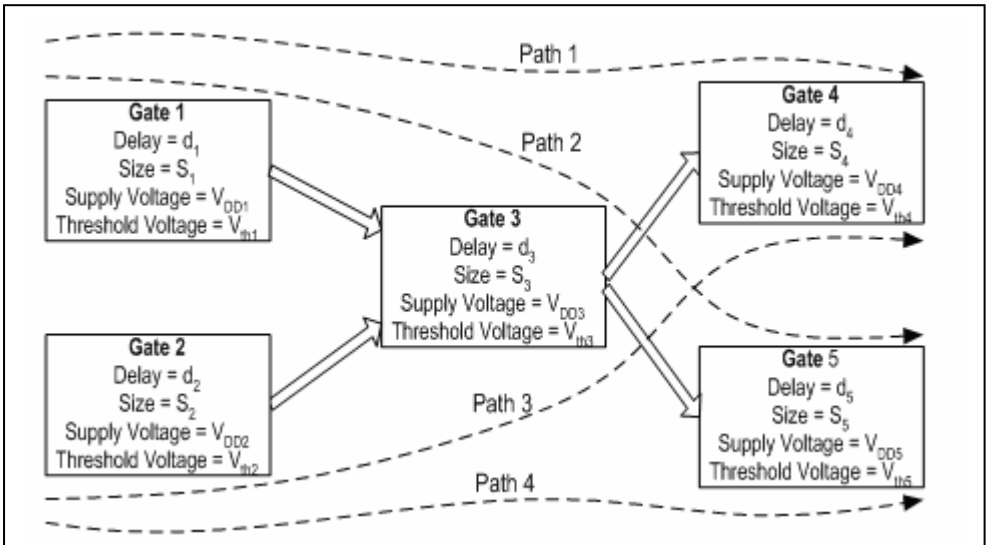


Figure 3.1. An example circuit with 5 gates and 4 paths (N=5, P=4).

Consider a digital circuit of N gates and P paths from primary inputs to primary outputs. A binary matrix, **T**, of P rows and N columns, representing the topology of the circuit is constructed as follows:

$$T_{ji} = 1 \ if \ gate \ i \ lies \ on \ Path \ j$$
$$= 0 \ otherwise \tag{3.1}$$

For example, the **T** matrix corresponding to the circuit in Figure 3.1 (N=5, P=4) is:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{3.2}$$

If a path P_u is a subset of P_v (i.e. all gates on P_u also lie on P_v), then the row of **T** corresponding to P_u ($Row_u(\mathbf{T})$) is removed from **T** to reduce unnecessary computation. The delay of each path in the system is given by:

$$T_j = \sum_{i \in P_j} d_i \quad for \ all \ j \tag{3.3}$$

We can represent the above equation in vector form as follows:

$$\overline{\mathbf{T}} = \mathbf{T} \cdot \overline{\mathbf{d}} \tag{3.4}$$

where $\overline{\mathbf{T}} = [T_1 \ T_2 \ . \ . \ . \ T_P]^T$ is the vector of path delays and $\overline{\mathbf{d}} = [d_1 \ d_2 \ . \ . \ . \ d_N]^T$ is the vector of gate delays.

Equation 3.4 represents P simultaneous equations in N variables. The P equations are known to be consistent since they were formed from the circuit topology. If P is greater than N, then the equations must be dependent. Even if P is less than or equal to N, the equations might be dependent. Hence, some equations can be dropped without losing any information. Only R (= rank(**T**)) number of rows can actually be used to represent the topology of the circuit. This is a significant benefit since P can be an exponential function of N and the corresponding **T** matrix would be very large.

The reduced matrix $\mathbf{T}^r$ that has only R independent rows (R<=N) can be computed using Gaussian-Elimination from the **T** matrix. However, it would still require

computation of the P circuit paths and thus could have exponential complexity. Gaussian-Elimination would also be computationally wasteful since information about the topology of the circuit would not be used at all. A procedure for computing the reduced topology matrix, $\mathbf{T}^r$, in an efficient manner is outlined below. The circuit in Figure 3.1 is used as an example.

The **T** matrix for the circuit in Figure 3.1 represents the following delay equations:

$$d1 + d3 + d4 = T1 \tag{3.5a}$$

$$d1 + d3 + d5 = T2 \tag{3.5b}$$

$$d2 + d3 + d4 = T4 \tag{3.5c}$$

$$d2 + d3 + d5 = T5 \tag{3.5d}$$

These can be rewritten as:

$$d1 + d3 + d4 = T1 \tag{3.6a}$$

$$d1 + d3 + d5 = T2 \tag{3.6b}$$

$$d2 - d1 = T4 - T1 \tag{3.6c}$$

$$d2 - d1 = T5 - T2 \tag{3.6d}$$

Equations 3.6c and 3.6d are equivalent and either of them can be dropped without losing any information about the circuit topology. Hence, the 4 equations, Eqs. 3.5a-d, are actually equivalent to the 3 equations, Eqs. 3.6a-c. The corresponding reduced topology matrix, $\mathbf{T}^r$, is:

$$\mathbf{T}^r = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix}_{3x5} \tag{3.7}$$

Note that row 3 of $\mathbf{T}^r$ is the difference of rows 1 and 3 of **T**.

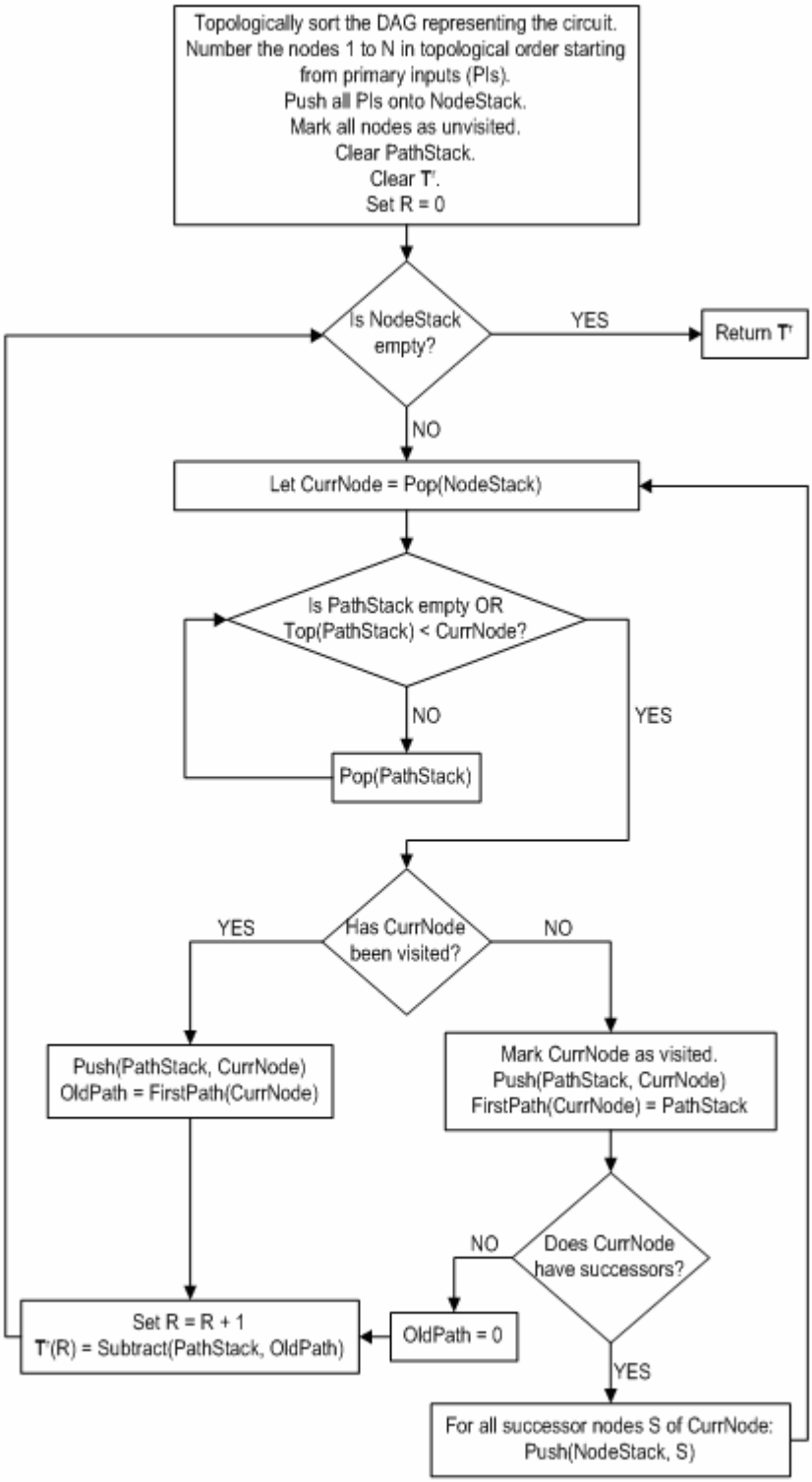


Figure 3.2. Algorithm for computing the reduced topology matrix, T^r.

The reduced set of equations can be obtained from the circuit by visiting all the gates in Depth First Order, starting from primary inputs. The complete algorithm for doing this is shown in Figure 3.2. The input to the algorithm is the Adjacency Matrix representation of the circuit. The output of the algorithm is the reduced topology matrix, $\mathbf{T}^r$. The first step is to topologically sort the nodes. The procedure for this is described in Appendix A. The procedure "Subtract" in the flowchart returns a "Difference" vector, PathDiff, of length N constructed from its 2 arguments Path1 and Path2 as follows:

(i) Initialize PathDiff to all zeros.

(ii) For all nodes p1 in Path1, set PathDiff(p1) = 1.

(iii) For all nodes p2 in Path2, set PathDiff(p2) = PathDiff(p2) -1.

Although $\mathbf{T}^r$ has less number of rows than $\mathbf{T}$ in general, it still might have more rows than R (=rank($\mathbf{T}$)) number of rows. In the worst case, it might have the same number of rows as $\mathbf{T}$. For example, consider the circuit in Figure 3.3. The $\mathbf{T}$ and $\mathbf{T}^r$ matrices corresponding to it are:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}^r = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix} \tag{3.8}$$

rank($\mathbf{T}$) is 3 but the reduced matrix $\mathbf{T}^r$ computed by the algorithm in Figure 3.2 has more rows than rank($\mathbf{T}$). However, the number of rows of $\mathbf{T}^r$ can still be bounded as shown in the following lemma.

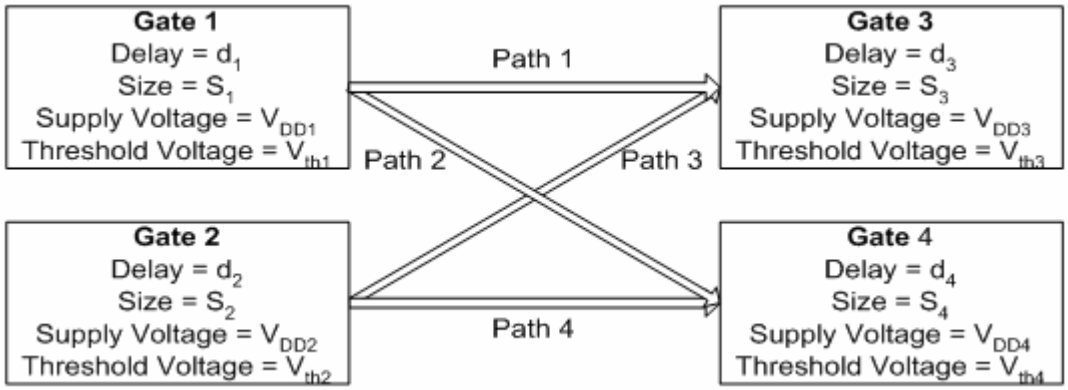


Figure 3.3. An example circuit with 4 gates and 4 paths (N=4, P=4).

Lemma 1: The number of rows of $\mathbf{T}^r$ is less than or equal to E, the number of edges in the circuit.

Proof: As seen from Figure 3.2, a row is added to $\mathbf{T}^r$ only when (i) a node that has already been visited is visited again or (ii) when the last node on a path is visited.

(i) A node i (with P_i number of predecessors) that was first visited through predecessor j can only be visited again through its remaining P_i -1 predecessors. This is because predecessor j would have been marked visited previous to its visit to i and hence all other (attempted) visits to i through j would stop at j. Similarly, i can only be visited by its remaining predecessors at most once because they also would have been marked visited previous to their visit to i. Hence, any node i can be visited again only P_i-1 times, leading to addition of atmost P_i-1 rows to $\mathbf{T}^r$. The total number of rows added is therefore $\sum_V (P_i - 1) = E - N$ where E is the number of edges and N is the number of gates.

(ii) A row is added to $\mathbf{T}^r$ also when a primary output is first visited. The total number of rows added is O, the number of primary outputs.

Hence, the total number of rows in $\mathbf{T}^r$ in the worst case is E-N+O which is less than or equal to E. □

The $\mathbf{T^r}$ matrix thus has significantly reduced number of rows compared to the **T** matrix. In the worst case of a completely connected graph, $\mathbf{T^r}$ has $O(N^2)$ rows whereas **T** has $O(2^N)$ rows.

3.3 Delay Assignment Variation based Optimization

Given the initial delay vector, $\overline{\mathbf{d}}_{init}$, the initial path delay vector, $\overline{\mathbf{T}}_{\mathbf{init}}$, can be obtained as in Equation 3.4. Since it is advantageous to work with the reduced topology matrix $\mathbf{T^r}$ instead of **T**, a path constraint vector, $\overline{\mathbf{T}}^{\mathbf{c}}_{\mathbf{init}}$, is constructed using $\mathbf{T^r}$ and used to specify the circuit delay constraints.

$$\overline{\mathbf{T}}^{\mathbf{c}}_{\mathbf{init}} = \mathbf{T}^{\mathbf{r}} \cdot \overline{\mathbf{d}}_{\mathbf{init}} \tag{3.9}$$

The delays of the gates in the circuit can be modified by adding a perturbation vector $\overline{\mathbf{\Delta}}$ to $\overline{\mathbf{d}}_{\mathbf{init}}$ such that $\mathbf{T^r} \cdot \overline{\mathbf{\Delta}} = \mathbf{0}$. This choice of $\overline{\mathbf{\Delta}}$ allows the new delay vector $\overline{\mathbf{d}}$ to still satisfy the initial path constraints as shown below:

$$\mathbf{T^r} \cdot \overline{\mathbf{d}} = \mathbf{T^r} \cdot \left(\overline{\mathbf{d}}_{\mathbf{init}} + \overline{\mathbf{\Delta}}\right) = \mathbf{T^r} \cdot \overline{\mathbf{d}}_{\mathbf{init}} + \mathbf{T^r} \cdot \overline{\mathbf{\Delta}} = \overline{\mathbf{T}}^{\mathbf{c}}_{\mathbf{init}} + \mathbf{0} = \overline{\mathbf{T}}^{\mathbf{c}}_{\mathbf{init}} \tag{3.10}$$

In other words, $\overline{\mathbf{\Delta}}$ has to lie in the nullspace of $\mathbf{T^r}$. Let the nullspace of $\mathbf{T^r}$ be **U**. **U** will have N rows and Q columns (Q<N), each column representing a basis vector for **U**. Then, any perturbation vector $\overline{\mathbf{\Delta}}$ can be constructed from any vector $\overline{\mathbf{r}}$ of length Q as follows:

$$\overline{\mathbf{\Delta}} = \mathbf{U} \cdot \overline{\mathbf{r}} \tag{3.11}$$

The vector $\overline{\mathbf{r}}$ represents the co-ordinates of $\overline{\mathbf{\Delta}}$ in the null-space of $\mathbf{T^r}$.

A delay assignment variation based optimization scheme can now be formulated as follows. Suppose the objective function to be optimized under delay constraints is a

function of the gate delays, represented by $F(\bar{\mathbf{d}})$. Then, $F(\bar{\mathbf{d}})$ can be translated into an equivalent function $G(\bar{\mathbf{r}})$ that has the delay constraints inbuilt into it by the following transformation:

$$G(\bar{\mathbf{r}}) = F(\bar{\mathbf{d}}_{\mathbf{init}} + \mathbf{U} \cdot \bar{\mathbf{r}}) \tag{3.12}$$

The benefit of this translation is that the number of variables is reduced from N (length of $\bar{\mathbf{d}}$) to Q (length of $\bar{\mathbf{r}}$). Any global optimization method can now be used to find the optimum value of $\bar{\mathbf{r}}$ that minimizes G and hence F.

Although theoretically $\bar{\mathbf{r}}$ can be any vector in $\mathbf{R}^Q$, there might be other limits on the allowed delay vector $\bar{\mathbf{d}}$ that will constrain $\bar{\mathbf{r}}$. For example, all entries of $\bar{\mathbf{d}}$ should be greater than 0. This constraint can be handled inside function G by scaling $\bar{\mathbf{r}}$ by a scalar α such that $\min(\bar{\mathbf{d}}_{\mathbf{init}} + \mathbf{U} \cdot \alpha\bar{\mathbf{r}}) > 0$. Lower and upper bound limits on $\bar{\mathbf{d}}$ can also be handled similarly.

Note that the above formulation only requires the nullspace of $\mathbf{T^r}$, **U**, which is the same as the nullspace of **T,** since both $\mathbf{T^r}$ and **T** represent the same set of equations. Hence, for notational convenience, **T** will be used with the understanding that the nullspace **U** is actually computed using $\mathbf{T^r}$.

In Chapter IV, the topology matrix representation is used to find exact conditions on module V_{DD}s and V_{th}s for minimum energy consumption. DAV based optimization is then used to find the values of the optimal V_{DD}s and V_{th}s. In Chapter V, a method is proposed for formulating the total energy consumption of a gate level netlist as a function of $\bar{\mathbf{d}}$ (the delays of the individual gates) and finding the optimum $\bar{\mathbf{d}}$ that minimizes the

energy consumption. In Chapter VI, a similar method is used to maximize the soft-error tolerance of the circuit.

The DAV optimization method presented can be used to further optimize designs that have already been optimized to some extent. For example, given the delays $\overline{\mathbf{d}}_{\mathbf{heuristic}}$ for the gates in a circuit that has been sized for speed using some heuristic, DAV based optimization can start from $\overline{\mathbf{d}}_{\mathbf{heuristic}}$ and using a circuit size cost function that can be formulated as a function of gate delays, C($\overline{\mathbf{d}}$), search for a perturbation $\overline{\mathbf{\Delta}}_{\mathbf{opt}}$ that lowers the cost function even more. In the worst case, $\overline{\mathbf{\Delta}}_{\mathbf{opt}}$ will be the zero vector, which will give us back the initial circuit. However, it is highly likely that the initial circuit hasn't been optimally sized, in which case $\overline{\mathbf{\Delta}}_{\mathbf{opt}}$ will be non-zero and C($\overline{\mathbf{d}}_{\mathbf{heuristic}} + \overline{\mathbf{\Delta}}_{\mathbf{opt}}$) will be lower than C($\overline{\mathbf{d}}_{\mathbf{heuristic}}$).

Chapter IV

Module Level Power Optimization

4.1. Introduction

Dynamic energy has been the main component of total energy since it is proportional to the square of $V_{DD.}$ However, with the shrinking of device sizes and reduction of supply voltages, static energy has become as important as dynamic energy. To obtain high gate overdrive (V_{DD} - V_{th}) for high speeds of operation, V_{th} is also decreased as V_{DD} is decreased. The decrease in threshold voltage increases the leakage current exponentially, which makes static energy consumption more significant in every new technology generation. Therefore, it has become essential to consider both supply and threshold voltage in any circuit optimization for low-energy consumption.

This chapter presents a method that considers both supply and threshold voltages *simultaneously* when optimizing circuits for low energy consumption at the module level. A metric is also provided that can be used by circuit designers to test how close the energy consumption of their module-level design is to the minimum possible. In this work, the metric is used to determine the stopping conditions of the optimization algorithm. If an unlimited number of supply and threshold voltages are available, the proposed algorithm *is optimum* in the sense that *no other voltage assignment for the given modules will give lower energy consumption for the given delay constraint.*

The complete procedure has two steps. The first step finds optimum supply and threshold voltage values for CMOS modules in a digital circuit that minimizes the total

energy consumption, using the techniques discussed in Chapter III. Considering a circuit as composed of modules allows energy optimization of much larger circuits than is possible with gate-level optimization algorithms. This is due to the significant reduction of problem complexity. The *exact* conditions for minimum energy are found using the Lagrange Multiplier Method. Then, the supply and threshold voltage values for each module that satisfy the minimum energy condition are found iteratively. If it is technologically feasible to assign the optimum (and perhaps all different) supply and threshold voltages to all the modules, then the algorithm stops here. Otherwise it continues to the next step.

The second step clusters the multiple voltages obtained from the first step into a fixed number of supply and threshold voltages (for example, 2 different supply voltages and 2 different threshold voltages). This step results in a feasible implementation of the system in current technologies. Section 4.2 gives the derivation of the minimum energy condition. Section 4.3 describes the search method used to obtain the optimal values. Section 4.4 describes the heuristic used to cluster the voltages. Section 4.5 gives experimental results.

4.2. Mathematical Condition for Minimum Energy Consumption at the Module Level

The topology matrix representation described in Chapter III can be used to derive an exact mathematical condition for minimum energy consumption at the module level. The energy-minimization problem for a digital system (assumed given to us) consisting of N modules and P paths from primary inputs to primary outputs can be stated as follows:

Minimize $\sum_{i=1}^{N} E_i$ under the constraints $\sum_{i \in P_j} d_i \leq T_d$ for all paths P_j where E_i is the energy consumption of the ith module, d_i is the delay of the ith module, T_d is the time constraint and the variables are V_{DDi} and V_{thi} for each module.

Figure 4.1 shows an example combinational circuit in which a module itself represents a block of logic with multiple gates. A module has to be chosen big enough so that the delay and energy overhead of the level-shifters that would be needed between modules is negligible compared to module delays and energies. In order to derive the exact mathematical condition for minimum energy consumption, we need equations for module delay, dynamic energy, and static energy in terms of V_{DD} and $V_{th.}$

For a CMOS circuit module, delay can be approximated as being proportional to $V_{DD} /(V_{DD} - V_{th})^{\alpha}$ [14].

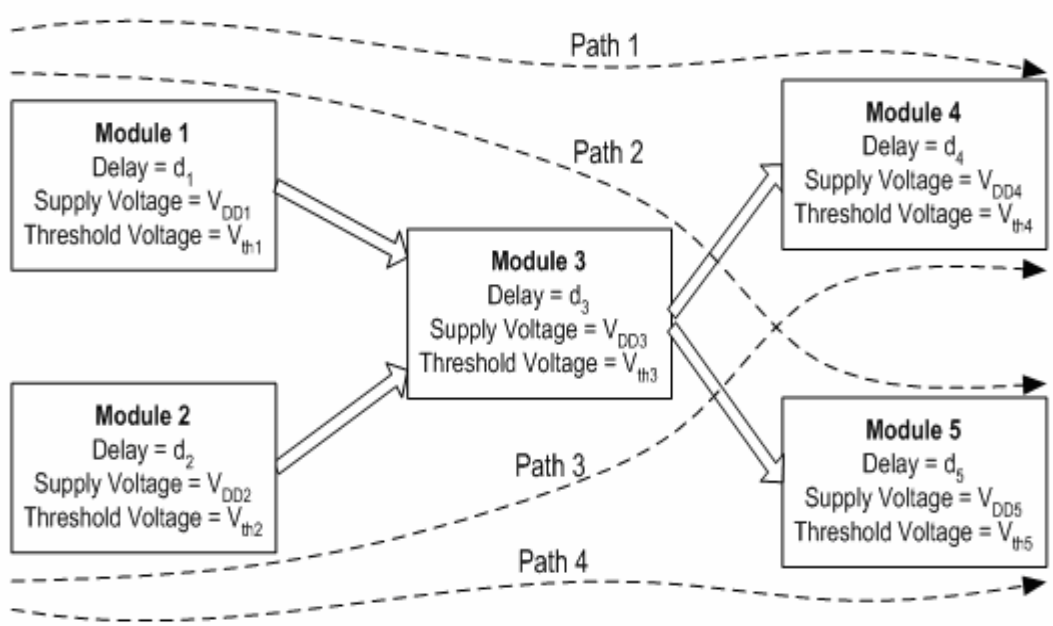


Figure 4.1. An example combinational circuit with 5 modules and 4 paths (N=5, P=4).

$$d_i = \frac{k_{0i} \cdot V_{DDi}}{(V_{DDi} - V_{thi})^{\alpha}} \tag{4.1}$$

Here d_i is the delay of the ith module, k_{0i} is a constant for that module, V_{DDi} and V_{thi} are the supply voltage and threshold voltage, respectively applied to that module, and α is the velocity saturation coefficient. For current technologies, α is between 1.2 and 1.5. The delay constant (k_{0i}) includes the effects of process, device sizes, load capacitance, and gate depth in that module. Gate depth can be included in the constant because of the additive characteristics of delays.

The energy consumption of a module is the sum of its dynamic and static energy consumptions. The dynamic energy consumed in a module can be written as:

$$E_{di} = 0.5 \cdot C_i \cdot V_{DDi}^2 \tag{4.2}$$

Here C_i is the term for all the capacitances that are switched during operation of the ith module including possible multiple switching of some nodes. To simplify the derivation, dynamic energy is rewritten as:

$$E_{di} = k_{1i} \cdot V_{DDi}^2 \tag{4.3}$$

where k_{1i} stands for the circuit, process, and application dependent terms including switching activity. Short circuit power dissipation can also be included in k_{1i} because of the quadratic dependence of short-circuit power dissipation to V_{DD} [13].

For static energy consumption, a generalized model is used.

$$E_{si} = E_{subi} + E_{gatei} = k_{2i} \cdot V_{DDi} \cdot e^{k_{3i} \cdot V_{DDi} - k_{4i} \cdot V_{thi}} \cdot T_d + k_{5i} \cdot V_{DDi} \cdot e^{k_{6i} \cdot V_{DDi} - k_{7i} \cdot V_{thi}} \cdot T_d \tag{4.4}$$

E_{subi} stands for the sub-threshold leakage component of the static energy consumption. This component is strongly influenced by the threshold voltage. E_{gatei} stands for the gate leakage component of the static energy. This component is much smaller than E_{subi} when V_{th} is small. When V_{th} is large, the main contribution to the static energy comes from this component. Also, gate leakage increases as the gate oxide thickness becomes smaller. k_{2i} and k_{5i} are circuit-dependent parameters. k_{3i}, k_{4i}, k_{6i}, and k_{7i} are process-dependent parameters. The values of the process-dependent parameters can be found by fitting SPICE simulation results of a simple gate to Equation 4.4. The values for these parameters can be used for any circuit designed in the same technology. After the process dependent parameters have been found, the circuit dependent parameters can be found by fitting the static energy consumption of the modules for different V_{DD}s and V_{th}s to Equation 4.4.

The delay constrained energy minimization problem can be formulated using the method of Lagrange Multipliers by constructing an auxiliary function as follows:

$$G(V_{DD1}, V_{th1}, \cdots, V_{DDN}, V_{thN}, \lambda_1, \cdots, \lambda_P) = \sum_{i=1}^{N} E_i - \sum_{j=1}^{P} \lambda_j \cdot [\sum_{i \in P_j} d_i - T_d] \tag{4.5}$$

where $E_i = E_{di} + E_{si}$ is the energy consumption of the ith module and λ_j is the Lagrange Multiplier for the constraint that the delay of the jth path be less than T_d.

Then, for minimum energy consumption, the following has to hold:

$$\frac{\partial G(V_{DD1}, V_{th1}, \cdots, V_{DDN}, V_{thN}, \lambda_1, \cdots, \lambda_P)}{\partial V_{DDi}} = 0 \quad for\ all\ i \tag{4.6}$$

$$\frac{\partial G(V_{DD1}, V_{th1}, \cdots, V_{DDN}, V_{thN}, \lambda_1, \cdots, \lambda_P)}{\partial V_{thi}} = 0 \quad for\ all\ i \tag{4.7}$$

Equations 4.6 and 4.7 become:

$$\frac{\partial E_i}{\partial V_{DDi}} = Row_i(\mathbf{T}^T) \cdot \overline{\boldsymbol{\lambda}} \cdot \frac{\partial d_i}{\partial V_{DDi}} \quad for\ all\ i \tag{4.8}$$

$$\frac{\partial E_i}{\partial V_{thi}} = Row_i(\mathbf{T}^T) \cdot \overline{\boldsymbol{\lambda}} \cdot \frac{\partial d_i}{\partial V_{thi}} \quad for\ all\ i \tag{4.9}$$

where $\overline{\boldsymbol{\lambda}} = [\lambda_1\ \lambda_2 \cdots \lambda_P]^T$ and $Row_i(\mathbf{T}^T)$ refers to the ith row of $\mathbf{T}^T$. Two vectors, the Constant Threshold Energy Gradient Vector ($\overline{CTEG}$) and the Constant Supply Energy Gradient Vector ($\overline{CSEG}$), can be defined as follows:

$$\overline{CTEG} = \left[\frac{\partial E_1}{\partial V_{DD1}} \Big/ \frac{\partial d_1}{\partial V_{DD1}} \quad \frac{\partial E_2}{\partial V_{DD2}} \Big/ \frac{\partial d_2}{\partial V_{DD2}} \quad \cdots \quad \frac{\partial E_N}{\partial V_{DDN}} \Big/ \frac{\partial d_N}{\partial V_{DDN}} \right]^T \tag{4.10}$$

$$\text{and } \overline{CSEG} = \left[\frac{\partial E_1}{\partial V_{th1}} \Big/ \frac{\partial d_1}{\partial V_{th1}} \quad \frac{\partial E_2}{\partial V_{th2}} \Big/ \frac{\partial d_2}{\partial V_{th2}} \quad \cdots \quad \frac{\partial E_N}{\partial V_{thN}} \Big/ \frac{\partial d_N}{\partial V_{thN}} \right]^T \tag{4.11}$$

Following are the equations for the partial derivatives of the energy function, E_i. These equations are obtained using Equations 4.1, 4.3 and 4.4.

$$\frac{\partial \mathrm{E_i}}{\partial V_{DDi}} = 2 \cdot k_{1\mathrm{i}} \cdot V_{DDi} + T_d \cdot V_{DDi} \cdot \left(k_{2i} \cdot k_{3i} \cdot e^{(k_{3i} \cdot V_{DDi} - k_{4i} \cdot V_{thi})} + k_{5i} \cdot k_{6i} \cdot e^{(k_{6i} \cdot V_{DDi} - k_{7i} \cdot V_{thi})} \right)$$

$$+T_d \cdot \left(k_{2i} \cdot e^{(k_{3i} \cdot V_{DDi} - k_{4i} \cdot V_{thi})} + k_{5i} \cdot e^{(k_{6i} \cdot V_{DDi} - k_{7i} \cdot V_{thi})} \right) \tag{4.12}$$

$$= \frac{2 \cdot E_{di}}{V_{DDi}} + k_{3i} \cdot E_{subi} + k_{6i} \cdot E_{gatei} + \frac{E_{subi} + E_{gatei}}{V_{DDi}}$$

$$\frac{\partial d_i}{\partial V_{DDi}} = k_{0i} \cdot \frac{\left(V_{DDi} - V_{thi}\right)^{\alpha} - \alpha \cdot \left(V_{DDi} - V_{thi}\right)^{\alpha - 1} \cdot V_{DDi}}{\left(V_{DDi} - V_{thi}\right)^{2 \cdot \alpha}}$$

$$= -\frac{d_i \cdot \left((\alpha - 1) \cdot V_{DDi} + V_{thi}\right)}{V_{DDi} \cdot \left(V_{DDi} - V_{thi}\right)} \tag{4.13}$$

$$\frac{\partial \mathrm{E_i}}{\partial V_{thi}} = -T_d \cdot V_{DDi} \cdot \left(k_{2i} \cdot k_{4i} \cdot e^{(k_{3i} \cdot V_{DDi} - k_{4i} \cdot V_{thi})} - k_{5i} \cdot k_{7i} \cdot e^{(k_{6i} \cdot V_{DDi} - k_{7i} \cdot V_{thi})} \right)$$

$$= -k_{4i} \cdot E_{subi} - k_{7i} \cdot E_{gatei} \tag{4.14}$$

$$\frac{\partial d_{\mathrm{i}}}{\partial V_{thi}} = \frac{\alpha \cdot k_{0i} \cdot V_{DDi}}{\left(V_{DDi} - V_{thi}\right)^{\alpha + 1}} = \frac{\alpha \cdot d_i}{\left(V_{DDi} - V_{thi}\right)} \tag{4.15}$$

Finally, from Equations 4.12-15:

$$CTEG_i = \frac{2 \cdot E_{di} + V_{DDi} \cdot \left(k_{3i} \cdot E_{subi} + k_{6i} \cdot E_{gatei}\right) + E_{subi} + E_{gatei}}{d_i \cdot \left((\alpha - 1) V_{DDi} + V_{thi}\right) / \left(V_{DDi} - V_{thi}\right)} \tag{4.16}$$

$$CSEG_i = \frac{k_{4i} \cdot E_{subi} + k_{7i} \cdot E_{gatei}}{\alpha \cdot d_i} \cdot \left(V_{DDi} - V_{thi}\right) \tag{4.17}$$

Using Equations 4.10, 4.11, 4.16 and 4.17, Equations 4.8 and 4.9 can now be written concisely as follows:

$$\begin{aligned} &\textit{Minimum Energy Condition}: \\ &\overline{\mathbf{CTEG}} = \overline{\mathbf{CSEG}} = \mathbf{T}^{\mathbf{T}} \cdot \overline{\boldsymbol{\lambda}} \end{aligned} \tag{4.18}$$

The minimum energy condition states that if the ith module has delay d_i, then the module consumes minimum energy when the supply voltage V_{DDi} and threshold voltage V_{thi} are uniquely chosen (out of all possible combinations that yield delay d_i) such that $CTEG_i$=$CSEG_i$. These optimal values of V_{DDi} and V_{thi} can be found by solving $CTEG_i$=$CSEG_i$ along with Equation 4.1 numerically. However, Equation 4.18 states that the entire system will consume minimum energy (under the delay constraint) if and only if each $CTEG_i$ (and $CSEG_i$) is also equal to the ith term of $\mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}$.

But $\bar{\boldsymbol{\lambda}}$ is an artificial quantity used to represent delay constraints and its value is unknown. The problem is thus to find the optimal delay vector $\bar{\mathbf{d}}^{\mathbf{opt}} = [d_1^{opt}\ d_2^{opt} \cdots d_N^{opt}]^T$ that meets the delay constraints and satisfies Equation 4.18 as well.

4.3. Procedure for obtaining optimal values of supply and threshold voltages

The optimal delay vector is found by starting from an initial delay vector $\bar{\mathbf{d}}^{\mathbf{init}}$ that meets the delay constraints and generating new delay vectors $\bar{\mathbf{d}}$ by adding perturbation vectors $\bar{\boldsymbol{\Delta}}$ (chosen from the nullspace of $\mathbf{T}$) to $\bar{\mathbf{d}}^{\mathbf{init}}$ (as described in Section 3.3). To check if this new $\bar{\mathbf{d}}$ is optimal, the "Normalized Energy Gradient" (NEG) metric, defined as follows, is used:

$$\mathrm{NEG}(\bar{\mathbf{d}}) = norm\left(\mathbf{T}^{\mathbf{T}} \cdot \mathbf{T}^{\mathbf{T}\dagger} \cdot \overline{\mathbf{CTEG}} - \overline{\mathbf{CTEG}}\right) \Big/ norm\left(\overline{\mathbf{CTEG}}\right) \tag{4.19}$$

where $\mathbf{T}^{\mathbf{T}\dagger}$ is the pseudo-inverse of $\mathbf{T}^{\mathbf{T}}$.

At the minimum energy point, when $\bar{\mathbf{d}} = \bar{\mathbf{d}}^{opt}$ (and correspondingly $\overline{\mathbf{CTEG}}^{opt} = \overline{\mathbf{CSEG}}^{opt} = \mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}$), NEG will be zero as shown below.

$$
\begin{aligned}
\mathrm{NEG}(\bar{\mathbf{d}}^{\mathbf{opt}}) &= norm\left(\mathbf{T}^{\mathbf{T}} \cdot \mathbf{T}^{\mathbf{T}\dagger} \cdot \overline{\mathbf{CTEG}}^{\mathbf{opt}} - \overline{\mathbf{CTEG}}^{\mathbf{opt}}\right) \Big/ norm\left(\overline{\mathbf{CTEG}}^{\mathbf{opt}}\right) \\
&= norm\left(\mathbf{T}^{\mathbf{T}} \cdot \mathbf{T}^{\mathbf{T}\dagger} \cdot \mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}} - \mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}\right) \Big/ norm\left(\mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}\right) \\
&= norm\left(\mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}} - \mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}\right) \Big/ norm\left(\mathbf{T}^{\mathbf{T}} \cdot \bar{\boldsymbol{\lambda}}\right) \\
&= 0
\end{aligned}
\tag{4.20}
$$

NEG is depicted as a function of $\bar{\mathbf{d}}$. To calculate NEG for any $\bar{\mathbf{d}}$, the condition $\overline{\mathbf{CTEG}} = \overline{\mathbf{CSEG}}$ is first met by choosing V_{DD}s and V_{th}s for every module as follows. To solve $CTEG_i = CSEG_i$ for the ith module, the delay, d_i, for that module is used in Equation 4.1 to write V_{thi} in terms of V_{DDi}. This makes $CTEG_i$ and $CSEG_i$ functions of V_{DDi} only and the equation $CTEG_i = CSEG_i$ can be solved easily (for e.g. using MATLAB's FZERO function) to get the V_{DDi} and V_{thi} values. NEG can then be computed using the V_{DD} and V_{th} values.

Note that the NEG metric can be computed for any digital system if the delay and energy equations for the modules in the system are known. The value of the NEG metric can be used as a measure of how good a design is in terms of energy consumption. Designs with high NEG values are energy inefficient for their delay constraint whereas those with NEG values close to zero are near optimal for their delay constraint.

An iterative gradient search based algorithm is used to get to the delay assignment at which NEG is zero. Any other global optimization algorithm could also be used. The inputs to the algorithm are the initial parameters of all the N modules, such as the V_{DDi}s, the V_{thi}s, the module delays (d_is) and the circuit- and process-dependent parameters k_{0i}s, k_{1i}s, k_{2i}s, k_{3i}s, k_{4i}s, k_{5i}s, k_{6i}s and k_{7i}s.

First, intermediate values for module delays, $\overline{\mathbf{d}}_{\mathbf{int}}$, are obtained that make all path delays as close to T_d as possible. This step also makes sure that all modules have zero slack, so that the starting point is optimal. A method similar to the Zero Slack Algorithm (ZSA) [55] is used in this step. It removes the slack of nodes in an energy-aware way (as described in Appendix B). Let $\overline{\mathbf{T}}_{\mathbf{int}} = \mathbf{T} \cdot \overline{\mathbf{d}}_{\mathbf{int}}$ be the vector of path delays after this step. Next, E_{total} is minimized by doing a gradient search on the delay vector, $\overline{\mathbf{d}}$.

The delay vector, $\overline{\mathbf{d}}_{\mathbf{new}}$, for a new iteration is obtained from the current delay vector, $\overline{\mathbf{d}}_{\mathbf{curr}}$, as follows:

$$\overline{\mathbf{d}}_{\mathbf{new}} = \overline{\mathbf{d}}_{\mathbf{curr}} + k \cdot \overline{\nabla}_{\mathbf{T}} E_{total} \tag{4.21}$$

where $\overline{\nabla}_{\mathbf{T}} E_{total}$ is the gradient of E_{total} along the nullspace vectors of **T**. k is chosen in such a way that the new energy ($E(\overline{\mathbf{d}}_{\mathbf{new}})$) is minimum in the direction of gradient vector. The search is stopped when a NEG value near 0 is obtained. The overview of the optimization algorithm is given in the flowchart in Figure 4.2.

4.4. Clustering heuristic for limited number of supply and threshold voltages

The algorithm described in the previous section yields optimum values of supply and threshold voltages for each module that minimize the overall circuit energy. But these voltages might all have different values, in which case a practical implementation of the optimized circuit is difficult in current technologies. In this subsection, a heuristic algorithm is described that clusters the optimum supply and threshold voltage values

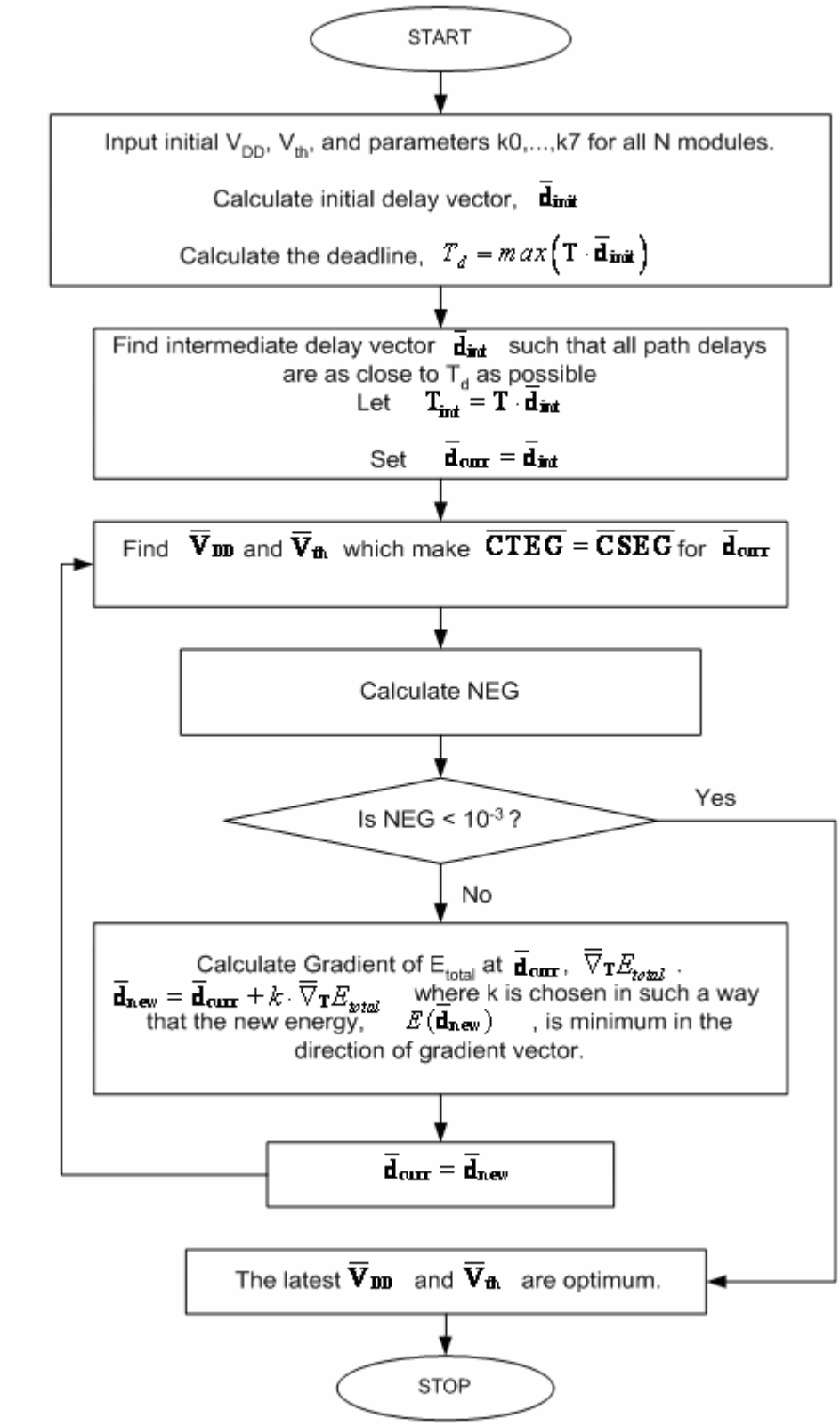


Figure 4.2. Algorithm for minimum energy consumption

obtained into a limited number of supply and threshold voltages. The final solution meets the delay constraint at the expense of slightly higher total energy consumption than the optimum case.

Assume only n supply voltage planes and m threshold voltages are available (n<N, m<N). Note that the values of the available voltages are not fixed at the beginning, although their number is fixed. Let $\overline{V}_{DD\_opt}$ and $\overline{V}_{th\_opt}$ be the optimum supply and threshold voltage values (obtained in the previous section), respectively. Let $\overline{V}_{DD\_n}$ and $\overline{V}_{th\_m}$ be supply and threshold voltage vectors holding values for the limited number of supply and threshold voltages (n supply voltages, m threshold voltages) initialized as follows:

$$\overline{V}_{DD\_n}(p) = min\left(\overline{V}_{DD\_opt}\right) + \left(\frac{max\left(\overline{V}_{DD\_opt}\right) - min\left(\overline{V}_{DD\_opt}\right)}{n+1}\right) \cdot p \qquad for\ p = 1 \cdots n \tag{4.22}$$

$$\overline{V}_{th\_m}(q) = min\left(\overline{V}_{th\_opt}\right) + \left(\frac{max\left(\overline{V}_{th\_opt}\right) - min\left(\overline{V}_{th\_opt}\right)}{m+1}\right) \cdot q \qquad for\ q = 1 \cdots m \tag{4.23}$$

These vectors will finally hold the n supply voltage values and m threshold voltage values that will be used in the circuit. For any module i, the function “Map” finds the nearest pair [$V_{DD\_n}(p)$, $V_{th\_m}(q)$] to the pair [$V_{dd\_opt}(i)$, $V_{th\_opt}(i)$] and assigns it to [$V_{DD\_new}(i)$, $V_{th\_new}(i)$].

$$[\overline{V}_{DD\_new}, \overline{V}_{th\_new}] = Map(\overline{V}_{DD\_opt}, \overline{V}_{th\_opt}, \overline{V}_{DD\_n}, \overline{V}_{th\_m}) \tag{4.24}$$

In any iteration, the delay of the circuit (T_c) is calculated using $[\overline{V}_{DD\_new}, \overline{V}_{th\_new}]$. $E_{total} \cdot T_c^2$ is used as the cost function if T_c exceeds T_d by a fixed fraction (say 0.01).

Doing this forces the critical path delay to go down in the next iteration, possibly increasing E_{total}. If T_c is less than T_d by a fixed fraction, E_{total} is used as the cost function. Doing this decreases the energy in the next iteration, possibly by increasing T_c. These cost functions were chosen because they yielded good results in experiments. The gradient, $\overline{\nabla}(Cost\_fn)$, is obtained by changing the entries of $\overline{V}_{DD\_n}$ and $\overline{V}_{th\_m}$ by a small amount, mapping these to new $[\overline{V}_{DD\_new}, \overline{V}_{th\_new}]$ and calculating the difference in the cost function. The new values of $\overline{V}_{DD\_n}$ and $\overline{V}_{th\_m}$, which lower the cost function, are obtained by searching in the direction of the gradient. The search terminates when the circuit delay is in 1% proximity of the deadline, T_d. The flowchart of the algorithm is given in Figure 4.3.

4.5. Experimental results

The hierarchical Verilog descriptions of the combinational ISCAS'85 circuits and a 16-bit Wallace Tree Multiplier were synthesized using Synopsys Design Compiler (with the TSMC 0.25μ library) to get the delay, dynamic energy and static energy consumption values for the modules at the top level of design hierarchy. The modules at the top level of hierarchy in the Verilog description were directly mapped to the modules used in the optimization*. The values of the process-dependent parameters (k_3, k_4, k_6, k_7) were obtained from SPICE simulations as explained in Section 4.2. SPICE simulation of simple gates showed that k_5 is 6 orders of magnitude smaller than k_2 for this technology. Since k_2 and k_5 scale almost linearly with number of gates [56][57], k_5 can be taken to be

* A power-aware partitioning of the circuit into modules could further improve the results, but that by itself is a very difficult problem to solve and is not handled in this work.

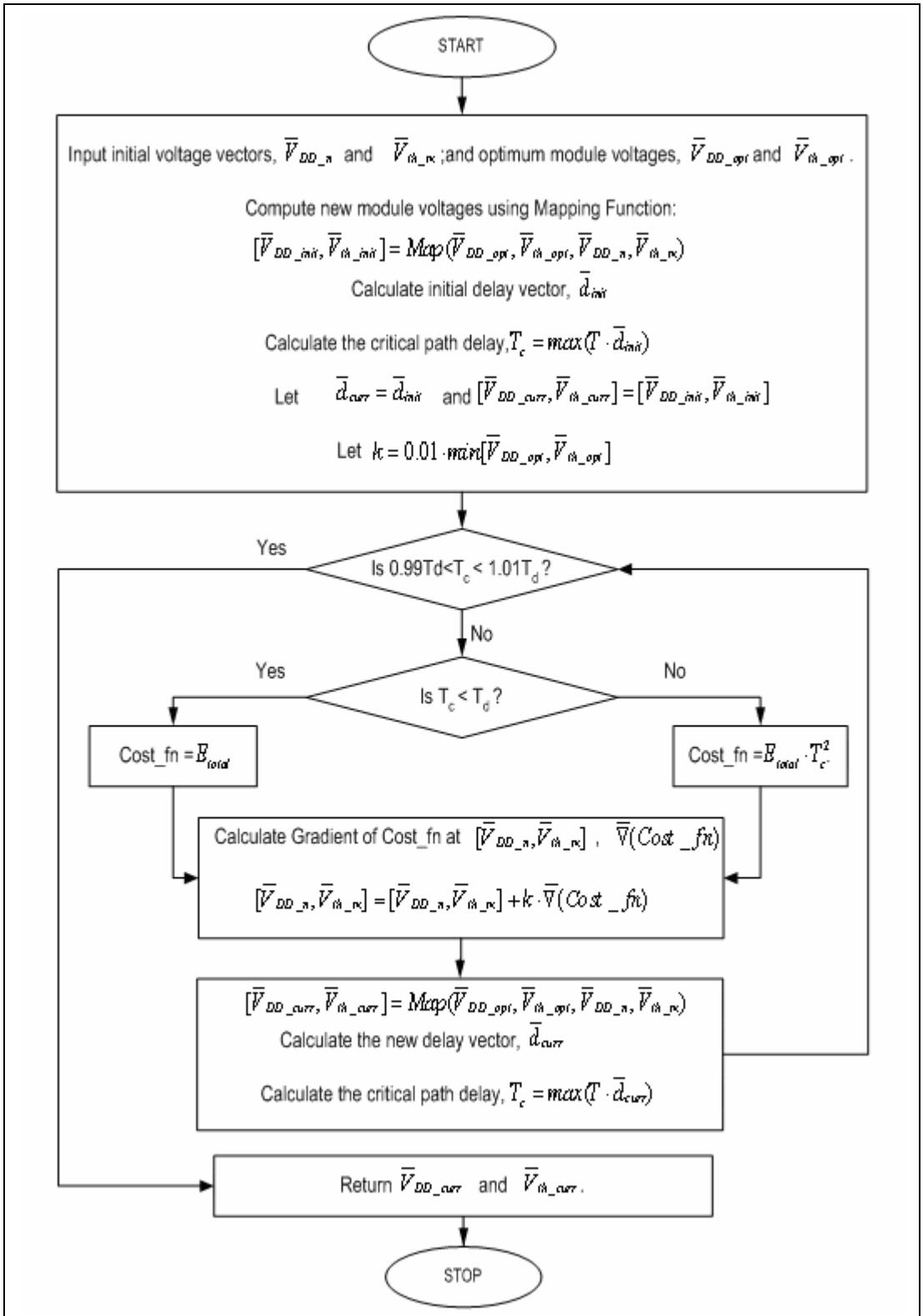
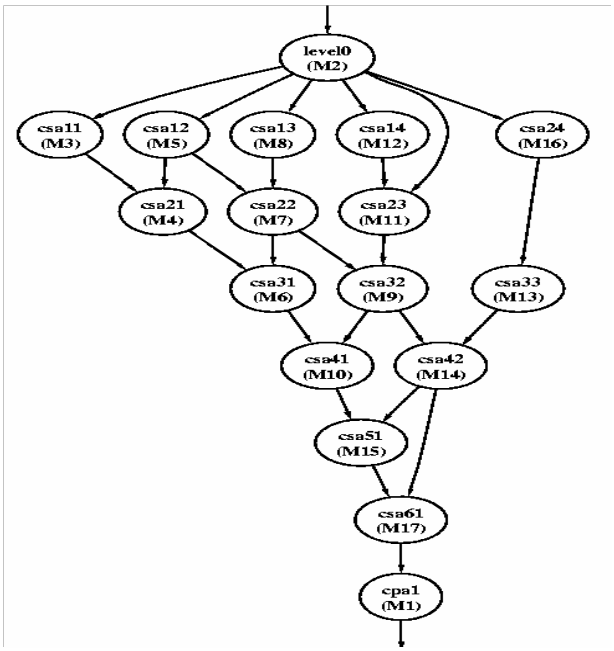


Figure 4.3. Algorithm for clustering

10^{-6} times k_2 for any module. The circuit-dependent parameters (k_0, k_1, k_2) were then calculated for each module by using the delay, dynamic energy and static energy values obtained from Synopsys and the process-dependent parameters.

The following notation is used for describing the results: The symbol "I" denotes the initial circuit which has the standard 0.25μ TSMC voltages (V_{DD} = 2.5V, V_{th} = 0.5V). The delay of the initial circuit is obtained using Synopsys Design Compiler and this value is used as the time constraint for the optimization i..e the optimized circuits (II, III, IV) will have the same delay as I. "II" denotes the baseline circuit (for energy comparisons) that has the *single V_{DD} and V_{th} values that give the minimum energy consumption for the given deadline*. "III" denotes the circuit having optimum (and possibly all different) V_{DD}s and V_{th}s for the modules. "IV" denotes the circuit in which the V_{DD}s and V_{th}s in III have been clustered into two V_{DD}s and one V_{th}. Only one V_{th} is used in the final circuit because it was found that having more V_{th}s only saved an additional 2-3% of energy in the benchmark circuits designed using 0.25μ technology. The need for multiple V_{th}s will become more pronounced as technology shrinks.

For the experiments, various switching activities were used for the input ports to observe their effects on the energy savings and the optimum voltages obtained. It was noticed that for switching activities above 0.05, the optimum V_{th}s were of the order of 10 mV. This is due to the fact that the static energy in 0.25μ technology is very small compared to the dynamic energy for high switching activities. So for these cases, the optimization algorithm scales down V_{DD} aggressively and to achieve the delay constraint, it reduces V_{th} to very small values without incurring a significant increase in static energy. Since such small V_{th} values are not currently feasible, for these cases V_{th} was



$$T^T = \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \\ 1&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&1&1&0&0&0&0&0&0 \\ 1&1&1&0&0&1&0&0&0&0&0 \\ 0&0&1&1&1&1&1&1&0&0&0 \\ 0&0&0&0&0&1&1&1&0&0&0 \\ 0&0&0&1&1&0&1&1&1&1&0 \\ 1&1&1&1&0&1&1&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&1&0&0&1&0&1&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

Figure 4.4. Wallace tree multiplier and associated T matrix

Table 4.1. Optimization results for Wallace tree multiplier

II (Baseline System)		III (unlimited V_{DD}s, V_{th}s)		IV (2 V_{DD}s, 1 V_{th})	
i	ii	i	ii	i	ii
		$[\overline{V}_{DD} \quad \overline{V}_{th}]$	$[\overline{V}_{DD} \quad \overline{V}_{th}]$	$[\overline{V}_{DD} \quad \overline{V}_{th}]$	$[\overline{V}_{DD} \quad \overline{V}_{th}]$
SA=0.01	SA=0.0001	2.23 0.10	2.80 0.33	1.84 0.09	2.42 0.34
		1.38 0.09	1.75 0.32	1.84 0.09	2.42 0.34
V_{DD}=1.62V	V_{DD}=2.18V	0.85 0.12	1.43 0.38	0.91 0.09	1.45 0.34
V_{th}=0.11V	V_{th}=0.35V	0.83 0.11	1.48 0.36	0.91 0.09	2.42 0.34
		0.85 0.12	1.43 0.38	0.91 0.09	1.45 0.34
T_d=13.7 ns	T_d=13.7 ns	0.77 0.08	1.39 0.34	0.91 0.09	1.45 0.34
		0.84 0.11	1.49 0.37	0.91 0.09	2.42 0.34
E=71.6 pJ	E=0.35 pJ	0.85 0.12	1.40 0.36	0.91 0.09	1.45 0.34
Ed=63.4 pJ	Ed=0.33 pJ	0.80 0.09	1.44 0.36	0.91 0.09	1.45 0.34
Es=8.2 pJ	Es=0.02 pJ	0.94 0.08	1.41 0.32	1.84 0.09	1.45 0.34
		0.84 0.11	1.46 0.37	0.91 0.09	2.42 0.34
		0.86 0.12	1.47 0.38	0.91 0.09	2.42 0.34
		0.50 0.09	0.97 0.33	0.91 0.09	1.45 0.34
		1.00 0.10	1.47 0.35	1.84 0.09	2.42 0.34
		0.63 0.06	1.07 0.30	0.91 0.09	1.45 0.34
		0.54 0.12	1.02 0.36	0.91 0.09	1.45 0.34
		0.64 0.06	1.07 0.30	0.91 0.09	1.45 0.34
		E=36.9 pJ	E=0.22 pJ	E=49.4 pJ	E=0.28 pJ
		Ed=31.8 pJ	Ed=0.20 pJ	Ed=41.9 pJ	Ed=0.25 pJ
		Es=5.1 pJ	Es=0.02 pJ	Es=7.5 pJ	Es=0.03 pJ
		Saving=48%	Saving=39%	Saving=31%	Saving=22%

fixed at 0.1V and the optimum V_{DD}s found. This phenomenon is not expected to occur for deep sub-micron technologies, where static energy is significant.

Figure 4.4 shows the top level of the Verilog design hierarchy for a Wallace Tree Multiplier. The modules are a partial product generator (level0), Carry Save Adders (CSAs), and a Carry Propagate Adder (CPA). Also shown is the **T** matrix corresponding to the entire circuit. The first column of Table 4.1 gives the V_{DD}, V_{th} and energy consumption values for the baseline Wallace Tree circuit (II) for two different input switching activities (SA=0.01 and SA=0.0001). Note that the delay for the baseline circuit is same as the delay of the initial circuit (I), which had V_{DD} = 2.5V, V_{th} = 0.5V. The second and third columns give the voltages for each module as well as the energy consumptions for circuits III and IV respectively.

Figure 4.5 shows the energy savings obtained for the various benchmark circuits as a percentage of the baseline energy consumption for an input switching activity of 0.01. The dynamic and static components of energy are also shown. It is observed that in II and III, static energy is ~10% of the total energy. This validates the fact that at the optimum, static energy is a fixed fraction of the total energy [58], although this fraction depends on the technology used. Figures 4.6 and 4.7 show the savings for different input switching activities for circuits III and IV, respectively. The results show that the energy savings tend to increase as the input switching activity increases. Thus, accurate estimation of the input switching activity is crucial for obtaining good energy savings.

Table 4.2 summarizes the results of the experiments. V_{DD1} and V_{DD2} are the two voltages applied to the circuit after clustering. Savings up to 48.4% savings were obtained for circuit III and up to 36% for circuit IV for switching activities below 0.05

(V_{th}s variable). For switching activities above 0.05 (V_{th}s fixed at 0.1V), savings up to 58.4% savings were obtained for circuit III and up to 31.6% savings for circuit IV. The average saving, for switching activities above 0.05, was 29% for circuit III and 18% for circuit IV. For switching activities below 0.05, the average saving was 28% for circuit III and 15% for circuit IV.

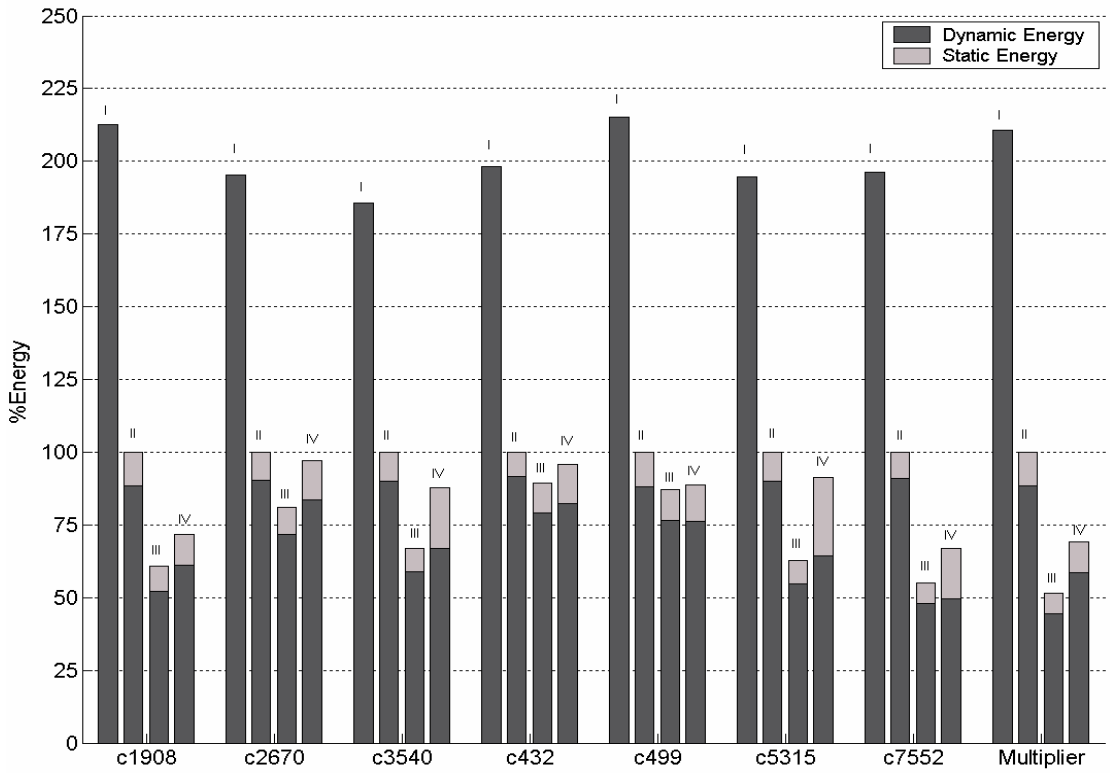


Figure 4.5. Energy consumption of benchmark circuits as a percentage of the baseline energy consumption when the input switching activity is 0.01.
I: Energy consumption of circuit with standard 0.25μ TSMC voltages (V_{DD} = 2.5, V_{th} = 0.5)
II: Energy consumption of circuit with optimum single V_{DD} and single V_{th} (baseline case)
III: Energy consumption of circuit with unlimited V_{DD}s and V_{th}s
IV: Energy consumption of circuit with two V_{DD}s and one V_{th}

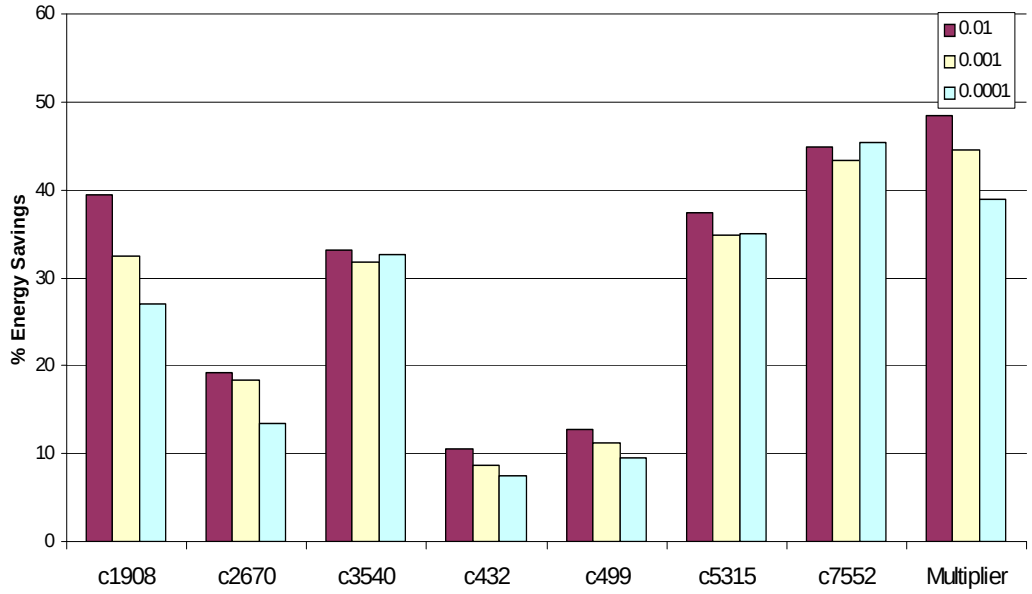


Figure 4.6. Percent energy savings with unlimited V_{DD}s and V_{th}s (III) for different input switching activities

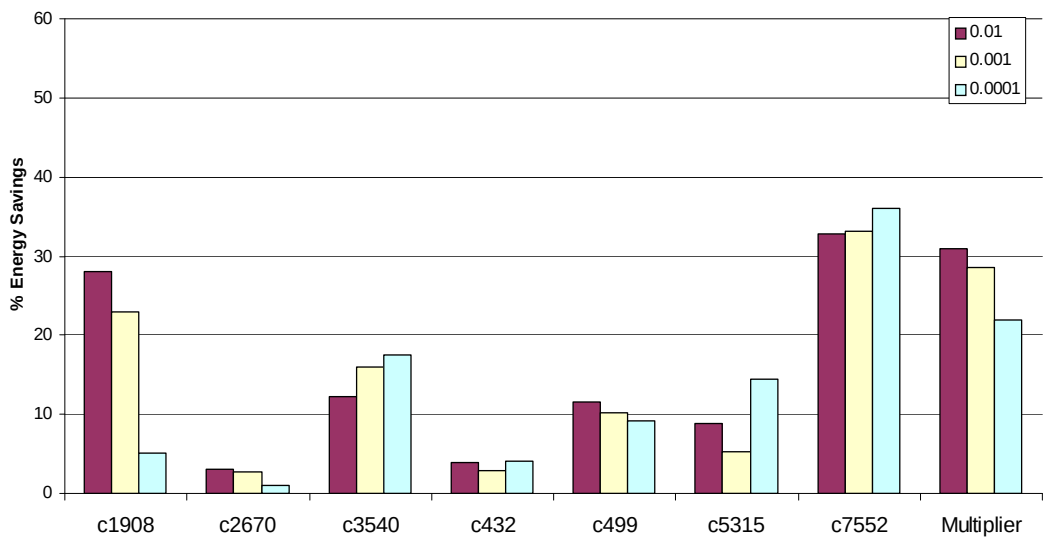


Figure 4.7. Percent energy savings with two V_{DD}s and one V_{th} (IV) for different input switching activities

Table 4.2. Optimization Results

Circuit	Input Switching Activity	E (I) pJ	E (II) pJ	E (III) pJ	E (IV) pJ	V_{DD} (II)	V_{th} (II)
c1908	0.5	106	44.2	25.9	30.6	1.6	0.10
	0.1	44.1	18.7	11.2	13.2	1.6	0.10
	0.01	5.85	2.75	1.67	1.98	1.6	0.10
	0.001	0.62	0.37	0.25	0.28	1.8	0.20
	0.0001	0.07	0.05	0.04	0.05	2.0	0.28
c2670	0.5	238	100	92.5	100	1.6	0.10
	0.1	78.1	33.5	31.2	33.5	1.6	0.10
	0.01	8.95	4.58	3.71	4.45	1.7	0.14
	0.001	0.85	0.55	0.45	0.53	1.9	0.23
	0.0001	0.09	0.07	0.06	0.07	2.1	0.32
c3540	0.5	414	175	120	139	1.6	0.10
	0.1	130	57.0	39.1	45.3	1.6	0.10
	0.01	14.4	7.75	5.18	6.81	1.7	0.16
	0.001	1.29	0.87	0.60	0.73	2.0	0.25
	0.0001	0.09	0.07	0.05	0.06	2.2	0.36
c432	0.5	23.5	9.81	9.05	9.47	1.6	0.10
	0.1	6.77	2.88	2.65	2.77	1.6	0.10
	0.01	0.74	0.37	0.33	0.36	1.7	0.14
	0.001	0.09	0.053	0.049	0.052	1.9	0.22
	0.0001	0.01	0.0084	0.0078	0.0081	2.1	0.30
c499	0.5	81.4	34.0	26.9	27.6	1.6	0.10
	0.1	34.4	14.5	11.8	12.0	1.6	0.10
	0.01	4.81	2.24	1.95	1.98	1.6	0.10
	0.001	0.49	0.29	0.26	0.26	1.8	0.19
	0.0001	0.05	0.039	0.035	0.035	2.0	0.28
c5315	0.5	438	184	110	153	1.6	0.10
	0.1	143	61.5	37.3	50.7	1.6	0.10
	0.01	16.7	8.59	5.38	7.83	1.7	0.14
	0.001	1.59	1.03	0.67	0.97	1.9	0.23
	0.0001	0.15	0.12	0.07	0.10	2.1	0.33
c7552	0.5	861	361	259	285	1.6	0.10
	0.1	283	121	84.7	86.0	1.6	0.10
	0.01	32.3	16.4	9.04	11.00	1.7	0.14
	0.001	3.34	2.12	1.20	1.42	1.9	0.22
	0.0001	0.42	0.32	0.17	0.20	2.1	0.31
Multiplier	0.5	2890	1210	502	834	1.6	0.10
	0.1	1180	500	245	342	1.6	0.10
	0.01	151	71.6	36.9	49.40	1.6	0.11
	0.001	11.7	7.21	4.00	5.16	1.9	0.21
	0.0001	0.43	0.35	0.22	0.28	2.2	0.35

Table 4.2. (Continued)

Circuit	V_{DD1} (IV)	V_{DD2} (IV)	V_{th} (IV)	% Energy Savings (III)	% Energy Savings (IV)
c1908	1.2	2.1	0.10	41.5	30.9
	1.2	2.1	0.10	40.2	29.4
	1.2	2.1	0.10	39.4	28.1
	1.4	2.4	0.19	32.5	22.9
	1.7	2.1	0.27	26.9	5.1
c2670	1.6	1.6	0.10	7.5	0
	1.6	1.6	0.10	6.9	0
	1.3	1.7	0.12	19.2	3.0
	1.5	1.9	0.22	18.4	2.6
	1.8	2.1	0.31	13.4	1.0
c3540	1.2	1.6	0.10	31.5	20.3
	1.2	1.7	0.10	31.4	20.6
	1.3	1.7	0.12	33.2	12.2
	1.5	2.0	0.22	31.8	16.0
	1.7	2.3	0.35	32.6	17.6
c432	1.5	1.8	0.10	7.7	3.5
	1.5	1.7	0.10	8.0	3.9
	1.5	1.8	0.11	10.6	4.0
	1.7	2.1	0.20	8.7	3.0
	1.9	2.4	0.29	7.5	4.1
c499	1.2	1.9	0.10	20.8	18.8
	1.2	1.9	0.10	18.5	17.1
	1.3	1.8	0.09	12.8	11.5
	1.5	2.0	0.19	11.1	10.2
	1.7	2.3	0.28	9.5	9.1
c5315	0.5	1.6	0.10	40.0	16.8
	0.5	1.6	0.10	39.3	17.5
	0.5	1.6	0.09	37.3	8.9
	0.6	1.8	0.18	34.8	5.3
	0.8	2.1	0.29	35.0	14.5
c7552	1.0	1.6	0.10	28.1	21.0
	0.6	1.6	0.10	29.9	28.9
	0.7	1.6	0.10	44.9	32.8
	1.0	1.9	0.20	43.4	33.2
	1.2	2.1	0.31	45.4	36.0
Multiplier	1.0	1.8	0.10	58.4	30.0
	1.0	1.8	0.10	51.0	31.6
	0.9	1.8	0.09	48.4	30.9
	1.1	2.1	0.20	44.5	28.5
	1.5	2.4	0.34	39.0	22.0

4.6. Conclusion

This chapter presented exact mathematical conditions on the supply and threshold voltages of modules in combinational circuits that minimize the total circuit energy under delay constraints. A procedure was described that used a DAV based search method to obtain the values of the optimal supply and threshold voltages. DAV based search was also used to finally cluster the many different voltage values obtained into a small number of voltages. For switching activities below 0.05, savings up to 48.4% savings were obtained if each module could have a separate supply and threshold voltage. For switching activities above 0.05 (V_{th}s fixed at 0.1V), savings up to 58.4% savings were obtained with each module having a separate supply voltage. The average savings for the above two cases were 28% and 29% respectively.

Chapter V

Gate Level Power Optimization

5.1 Introduction

This chapter discusses the application of DAV based optimization to the problem of power minimization using sizing, multiple V_{DD}s and V_{th}s at the gate level. The approach described in Chapter IV cannot be straightforwardly extended for optimization at the gate level because of several reasons:

a) Gate sizes are important optimization variables. However, they are not included in the module level equations.

b) Inclusion of gate sizes as variables makes the gate delays dependent on successor gate sizes (and hence successor delays). At the gate level, an analytical delay model will have to include the sizes of the successor gates whereas this was not the case at the module level. This makes analytical derivation of the exact minimum energy condition impossible at the gate level.

c) The overhead of level-shifters could be neglected at the module level, but this is not possible at the gate level.

d) The delay and energy equations in Chapter IV are simplistic and do not take into account second and third order effects that are present in current deep-sub-micron (DSM) technologies.

To tackle these problems, a DAV search based approach for gate level power optimization is developed in this section. It allows selection of optimal gate sizes, V_{DD}s

and V_{th}s for minimizing power consumption. Furthermore, delays, energy consumptions, output ramps and input capacitances of gates are modeled using SPICE simulation based look-up tables. This allows very accurate modeling of leakage energies and capacitances in the DSM domain as compared to analytical models. Section 5.2 gives the methodology developed for delay and energy modeling of gates in the circuit netlist. Section 5.3 describes a method to formulate the energy consumption of a circuit as a function of the gate delays, which is needed to apply DAV based optimization as described in Section 3.3. Section 5.4 describes the application of DAV based optimization in a hierarchical fashion that allows optimization of large netlists efficiently. Section 5.5 analyzes the computational complexity of the optimization with and without hierarchical optimization. Section 5.6 provides details of the implementation of the optimization framework and gives results of experiments on ISCAS'85 benchmark circuits.

5.2 Delay, Energy, Output Ramp and Input Capacitance Modeling

Deep-sub-micron technologies have a lot of second and third order effects that closed form analytical equations cannot capture. For example, the input capacitance of a gate changes according to the input signal slope. It is also different for different values of input signal amplitude, gate V_{DD}, V_{th}, etc. Similarly, delay and energy at the gate level are not very accurately modeled using the simple equations 2.1, 2.2 and 2.3. These equations do not incorporate gate size in them, which is needed to accurately capture the effect of sizing on a gate. Output ramp is another parameter that is difficult to model using analytical equations. Furthermore, it is difficult to model gate leakage analytically. To get around these limitations of analytical models, SPICE tables were used for the modeling. SPICE simulations for 70nm CMOS technology [59] were carried out to get

the values of delay, input capacitance, dynamic energy consumption, static energy consumption and output signal ramp (average of rise time and fall time) for 2 to 4 input NAND and NOR gates, and inverters. The variables for each gate were its size, the applied supply voltage (V_{DD}), the threshold voltage (V_{th}) of the NMOS and PMOS transistors (assumed equal in magnitude), the input signal amplitude, the input signal ramp and the load capacitance at its output (see Figure 5.1). Tables were built for delay, input capacitance, dynamic and static energy consumption values, and output signal ramp for gate sizes of 1 to 10, V_{DD} values of 0.8V to 1.2V (in steps of 0.2V), V_{th} values of 0.1V to 0.4V (in steps of 0.1V), input signal amplitudes of 0.8V to 1.2V (same as for V_{DD}), several input signal ramps ranging from 5ps to 75ps, and several load capacitances ranging from 0.25fF to 50fF. Appropriate interpolation was used to obtain the values of delay, input capacitance, dynamic and static energy consumption, and output signal ramp for input variables that were inside the ranges given above. For example, to determine dynamic energy for an arbitrary V_{DD} value (say 1.9V), quadratic interpolation was used (because of the quadratic relation between dynamic energy and V_{DD}) between the dynamic energy values for 0.8V and 1V obtained from the table. Similarly, to determine delay for an arbitrary gate size (say 2.6), reciprocal interpolation was used (because of the reciprocal relation between delay and gate size) between the delay values for sizes 2 and 3 obtained from the table. The ranges for the variables given above were chosen in such a way that extrapolation would not be needed during the optimization.

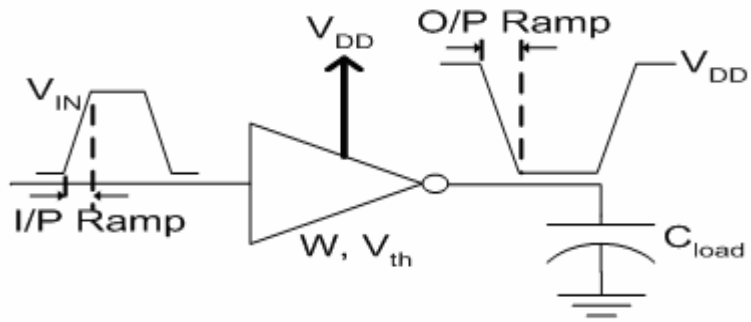


Figure 5.1. Variables for delay, energy, output ramp and input capacitance modeling

The measured delay in SPICE was the average delay for rising input and falling input (of amplitude V_{IN}, a variable) to the gate with the other gate inputs (in case of NAND and NOR) set to their sensitizing values. The input capacitance (C_{in}) of a gate was measured by applying a voltage pulse of amplitude V_{IN} with rise and fall time T to the gate and measuring the average current flowing into (I_{IN}) or out of (I_{OUT}) the gate. C_{in} is then taken as the average of $\frac{I_{IN} \cdot T}{V_{IN}}$ and $\frac{I_{OUT} \cdot T}{V_{IN}}$. The dynamic energy consumption for a 0-1 transition (E^{01}) and a 1-0 transition (E^{10}) at the output was measured by averaging the current supplied by the voltage source for a small time-period around the respective transitions and multiplying with V_{DD} and the time-period of measurement. The static power consumption was measured for output at 0 (P^0) and 1 (P^1) by measuring the average current drawn from V_{DD} with the output settled at the respective values and multiplying with V_{DD}. The logic "1" value of the input signal corresponded to voltage V_{IN}. Thus, for V_{IN} values less than V_{DD}, P^0 also included the effect of leakage in the gate PMOS network (due to the PMOS transistors not being completely off). The average energy consumption of a gate in a circuit with clock cycle of T_{clock} was obtained as follows:

$$E = T_{clock} \cdot \left(prob^0 \cdot P^0 + prob^1 \cdot P^1 \right) + \eta \cdot \left(\frac{E^{01} + E^{10}}{2} \right) \tag{5.1}$$

where $prob^0$ and $prob^1$ were the static probabilities of the output of the gate being 0 and 1 respectively and η was the switching activity at the gate output. The output signal ramp was measured as the average of rise times and fall times of the output signal.

5.3 Delay Assignment Variation (DAV) based Gate Level Power Optimization

Figure 5.2 shows an example combinational circuit driving a known output load C_{out}. The gates have been numbered in topological order (Appendix A). Suppose the topology matrix for the circuit is **T** and the initial delays corresponding to the gates are represented by vector $\overline{\mathbf{d}}_{\mathbf{init}} = [d_1 \quad d_2 \quad d_3 \quad d_4 \quad d_5 \quad d_6 \quad d_7 \quad d_8]^T$. The delay-constraint is $T_{dead} = \max(\mathbf{T} \cdot \overline{\mathbf{d}}_{\mathbf{init}})$. As described in Section 3.3, the delays can be

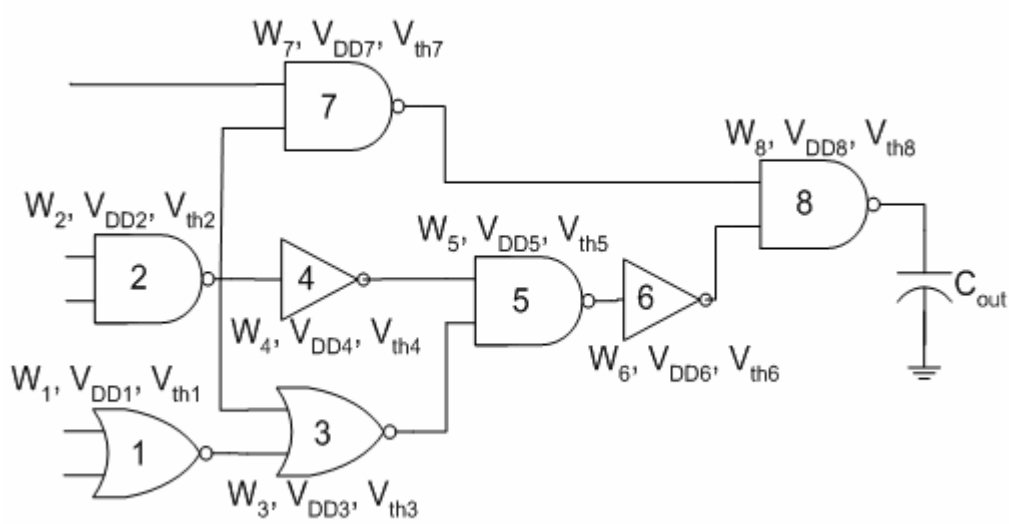


Figure 5.2. An example circuit with 8 gates and 4 paths (N=8, P=4).

perturbed by $\overline{\mathbf{\Delta}}$ in the null-space of **T** without violating the delay-constraint. Let the new delay vector be $\overline{\mathbf{d}} = \overline{\mathbf{d}}_{\mathbf{init}} + \overline{\mathbf{\Delta}}$. Then, the required sizes, V_{DD}s and V_{th}s for the gates to match the new delays can be computed by a backward traversal from primary outputs (POs) to primary inputs (PIs). The allowed sizes, V_{DD}s and V_{th}s to be used for the matching can be given as input.

For example, first, values of size, V_{DD} and V_{th} are chosen for gate 8 that match the assigned delay, d_8 (say 15ps), while minimizing the energy-load product (ELP) for the gate and its predecessors. The predecessors also have to be considered while matching the delay of a gate since the size chosen for the current gate determines the load of the predecessor gates. For example, if (size,V_{DD},V_{th}) values of (5,1.2,0.2) and (10,1,0.3) both yield a required delay of 15ps for gate 8 when driving a load C_{out}, the first set of values might be chosen over the second (even though the second set of values yields lower energy consumption for the gate) since the first set will reduce the capacitive loading of the predecessor gates (and hence make their delay matching possible with smaller sizes) yielding (possibly) a lower value of total ELP for the gate and its predecessors.

The load capacitances of gates 6 and 7 can now be determined from the size, V_{DD} and V_{th} of gate 8 by looking up its input capacitance in the SPICE tables. Once the load capacitances of gates 6 and 7 are known, the best sizes, V_{DD}s and V_{th}s can be found for them to match delays d_6 and d_7 respectively. This process can then be repeated till the PIs are reached. At the end, the gate sizes, V_{DD}s and V_{th}s for all gates to match the new delay assignment $\overline{\mathbf{d}}$ will be known. The total energy consumption of all the gates, the circuit area, the circuit delay, etc for this assignment can then be computed using the SPICE tables. Figure 5.3 gives a formal description of the algorithm for matching delays to gate

```
Procedure match_delay(d_vec, allowed_sizes, allowed_vdds, allowed_vts)

For i = N down to 1

        d = d_vec(i)
        If gate i is a Primary Output
                Cload = I/P capacitance of flip-flop
        Else
                Cload = 0
        End

        For all successors j of gate i
                Cload = Cload + I/P capacitance of gate j
        End

        Matching_set = Null
        For all combinations of sizes w, VDDs v and Vths u from the allowed set
                If Delay(w,v,u,Cload) is equal to d
                        Add (w,v,u) to Matching_set
                End
        End

        minELP = infinity
        For all entries (w,v,u) in Matching_set
                ELP = Energy(w,v,u,Cload)*Cload
                For all predecessors j of gate i
                        ELP = ELP + 0.5*Activity(j)*InputCap(w,v,u,Cload)^2*v^2
                End
                If ELP < minELP
                        minELP = ELP
                        min_set = (w,v,u)
                End
        End

        Assign the parameters in min_set to gate i
        Compute delay, energy and I/P capacitance of gate i using obtained parameters
        and the SPICE tables
End
```

Figure 5.3. Procedure for matching delays of gates to gate sizes, V_{DD}s and V_{th}s

sizes, V_{DD}s and V_{th}s. The inputs are the delay vector to be matched and the sets of sizes, V_{DD}s and V_{th}s allowed for the matching. Delay(w,v,u,Cload), Energy(w,v,u,Cload) and

InputCap(w,v,u,Cload) represent the SPICE look-up tables for delay, energy and input capacitance respectively.

Experimental comparison between an energy (E) metric and the ELP metric for choosing between different (size,V_{DD},V_{th}) sets showed that ELP was a better metric. The E metric almost always favors giving bigger sizes to a gate since that allows the usage of lower V_{DD} and higher V_{th} while matching the delay, hence reducing the energy a lot for the gate. However, the sizes of the predecessors then have to be increased even more to match their delay assignment. This can have a cascading effect which leads to a high energy matching of the delays for the overall circuit. On the other hand, the ELP metric favors giving smaller sizes to a gate even if it means higher energy consumption for the gate. However, this tends to lead to a lower energy matching of the delays for the overall circuit (as the cascading effect is reduced).

In some cases, the ELP metric can also lead to a cascading effect if while matching the delay of a gate, a bigger size still has lower ELP. For some delay assignments, it is better to use the minimum size gates that match the delay assignment. This is sure to minimize the cascading effect but might lead to usage of higher V_{DD}s and lower V_{th}s. A compromise would be to compare the circuit energies for a delay assignment matched using the ELP metric and the min-size metric and choose the lower of the two.

As an aside, it is interesting to note that the delay matching at the gate level is significantly more complicated than at the module level (as in Section 4.3). At the module level, the delay equation used did not have any term for load capacitance. Hence, the delays of a module could be matched independently of its successors. The V_{DD} and

V_{th} values chosen for a module were those that minimized its energy consumption for the delay assignment. Thus, for any delay assignment, the energy consumption of the overall circuit was guaranteed to be the minimum possible. At the gate level, however, the delay of a gate is highly dependent on its successor gates. The size, V_{DD} and V_{th} values chosen for a gate are not guaranteed to minimize its energy consumption (and are indeed chosen to minimize ELP or gate sizes) since the values depend on the parameters of the successor gates. Thus, for any delay assignment, there is no guarantee that the energy consumption of the overall circuit is the minimum possible for that delay assignment. Another assignment might exist that matches the delays while giving lower energy. However, matching the delays using the ELP metric and/or the min-size metric gives very good results. This brings in a heuristic element into the gate level optimization framework that is not the case at the module level.

The procedure described above formulates the energy consumption of a circuit in terms of the delays of the gates i.e. it formulates E($\bar{\mathbf{d}}$). This is the first step needed to apply DAV based optimization. As described in Section 3.3, E($\bar{\mathbf{d}}$) can be represented as E'($\bar{\mathbf{r}}$) where $\bar{\mathbf{r}}$ represents the co-ordinates of $\bar{\mathbf{\Delta}}$ in the null-space of **T**. The energy consumption of the circuit can now be minimized by finding the global minimum of E'($\bar{\mathbf{r}}$) using any global optimization algorithm. The corresponding vector $\bar{\mathbf{r}}_{min}$ can be used to find $\bar{\mathbf{d}}_{\mathbf{min}} = \bar{\mathbf{d}}_{\mathbf{init}} + \mathbf{U} \cdot \bar{\mathbf{r}}_{\mathbf{min}}$, the optimum delay assignment for the gates that minimizes energy. The optimum gate sizes, V_{DD}s and V_{th}s can then be found using the procedure described above.

There are a few practical issues with the procedure outlined above. First, the procedure described above could find different V_{DD}s for different gates from the allowed set of V_{DD}s. This would necessitate the use of a level-shifter whenever a low V_{DD} gate drove a high V_{DD} gate with a lot of accompanying overhead. Second, the size of the problem (equal to size of $\bar{\mathbf{r}}$) for big circuits can be too big to be handled by any global optimization algorithm in reasonable computation time. Third, the current delay assignment might not always be matched exactly because of the finite number of sizes, V_{DD}s and V_{th}s available, leading to a possible violation of the delay constraint.

The third problem can be handled by realizing that the delay assignment variation method is just a way to explore the design space. Hence, by formulating a cost function that is a weighted sum of the normalized circuit delay violation and the normalized energy consumption and then minimizing it, the best design in terms of circuit delay and energy consumption can still be obtained. The energy-delay product (EDP) of the entire circuit can also be used as a cost function.

The next section describes how the first two problems can be mitigated by partitioning the circuit into smaller sub-circuits and doing the optimization hierarchically.

5.4 Hierarchical Application of DAV based optimization

Figure 5.4 shows the example combinational circuit partitioned into two sub-circuits. All gates in a sub-circuit can be constrained to have a single value of V_{DD}. This reduces the need for level-shifters to just the boundaries between sub-circuits. The value

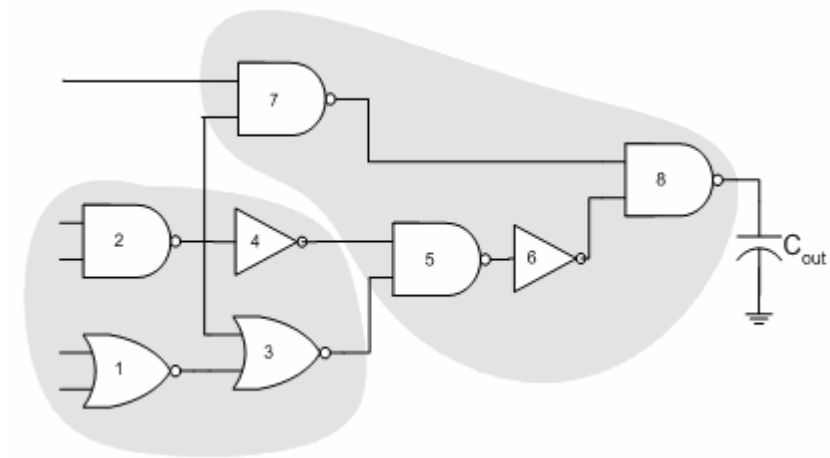


Figure 5.4. Example combinational circuit partitioned into 2 sub-circuits

of V_{DDi} for sub-circuit 'i' then also becomes an optimization variable and can be added to the problem i.e E'($\bar{\mathbf{r}}$) can be reformulated as E'($\bar{\mathbf{r}}_{\mathbf{subi}}$, V_{DDi}), where $\bar{\mathbf{r}}_{\mathbf{subi}}$ refers to the co-ordinates of a vector in the nullspace of the topology matrix for sub-circuit 'i', $\mathbf{U_{subi}}$. Furthermore, by partitioning the circuit such that a minimum numbers of edges are cut between sub-circuits, the overhead of level-shifters can be brought further down. Partitioning also reduces the size of $\bar{\mathbf{r}}$ for each sub-circuit and hence allows for faster optimization.

The topology matrix of sub-circuit 'i', $\mathbf{T_{subi}}$, can be computed easily from the topology matrix of the entire circuit, **T**. It is just those columns of **T** that correspond to the gates in sub-circuit 'i'. For example, the topology matrix for the example circuit is:

$$\mathbf{T} = \begin{array}{c} \begin{array}{cccc|cccc} \mathbf{T_{sub1}} & & & & \mathbf{T_{sub2}} & & & \end{array} \\ \left[\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \end{array} \tag{5.2}$$

If the sub-circuit with gates 1, 2, 3 and 4 is sub-circuit 1, and the other sub-circuit is 2, then the sub-circuit topology matrices are:

$$\mathbf{T}_{\mathbf{sub1}} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{T}_{\mathbf{sub2}} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{5.3}$$

The reduced topology matrix, $\mathbf{T}^{\mathbf{r}}$, is:

$$\mathbf{T}^{\mathbf{r}} = \begin{array}{c} \begin{array}{cc} \mathbf{T}^{\mathbf{r}}_{\mathbf{sub1}} & \mathbf{T}^{\mathbf{r}}_{\mathbf{sub2}} \end{array} \\ \left[\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & -1 & -1 & 1 & 0 \end{array}\right] \end{array} \tag{5.4}$$

$\mathbf{T}^{\mathbf{r}}$ is the same size as $\mathbf{T}$ since all the path equations are independent. The corresponding reduced sub-circuit topology matrices are:

$$\mathbf{T}^{\mathbf{r}}_{\mathbf{sub1}} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 0 \end{bmatrix}, \mathbf{T}^{\mathbf{r}}_{\mathbf{sub2}} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 \end{bmatrix} \tag{5.5}$$

Just as the null-spaces of $\mathbf{T}$ and $\mathbf{T}^{\mathbf{r}}$ are the same (since they represent the same set of equations), similarly the null-spaces of $\mathbf{T}_{\mathbf{subi}}$ and $\mathbf{T}^{\mathbf{r}}_{\mathbf{subi}}$ are also the same (they also represent the same set of equations). Hence, the computation of the null-spaces of the sub-circuit topology matrices can also be done from the reduced topology matrix.

Application of DAV based optimization to a sub-circuit implies that the delays of paths inside the sub-circuit do not change. This is an extra restriction that does not allow path delay variation between sub-circuits. This may not be desirable in the case when the energy consumptions of the sub-circuits are very different. In this case, the path delays of

the high energy consuming sub-circuits should be increased (from their initial values) compared to the low energy consuming sub-circuits so that the overall circuit can operate at lower energy. However, applying DAV based optimization to sub-circuits will not let this happen as it will keep the sub-circuit path delays at their initial values.

The solution to this problem is to first apply DAV based optimization between sub-circuits and then within each sub-circuit. To apply DAV based optimization between sub-circuits, each sub-circuit 'i' is assigned a number, Δ_{subi}, representing the amount of extra delay added to the gates in the sub-circuit. Assuming the circuit is partitioned into K sub-circuits, let $\bar{\mathbf{\Delta}}_{\mathbf{sub}} = [\Delta_{sub1}\ \Delta_{sub2}\ \ldots\ \Delta_{subK}]^T$ represent the perturbation vector for the K sub-circuits. Then, $\bar{\mathbf{\Delta}}_{\mathbf{sub}}$ is used to construct the perturbation vector, $\bar{\mathbf{\Delta}}_{\mathbf{pseudo}}$, that will be used to perturb the gate delay vector, $\bar{\mathbf{d}}$. First, a partition matrix, **P**, of K rows and N columns, representing the partitioning scheme is constructed as follows:

$$\begin{aligned} \mathbf{P}_{ji} &= 1 \ \ if\ \ gate\ \ i\ \ lies\ \ in\ \ partition\ \ j \\ &= 0 \ \ otherwise \end{aligned} \tag{5.6}$$

For the example circuit, the **P** matrix is:

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}_{2\ x\ 8} \tag{5.7}$$

Then, $\bar{\mathbf{\Delta}}_{\mathbf{pseudo}}$ is given by:

$$\bar{\mathbf{\Delta}}_{\mathbf{pseudo}} = \mathbf{P}^{\mathbf{T}} \cdot \bar{\mathbf{\Delta}}_{\mathbf{sub}} \tag{5.8}$$

For example, given the sub-circuit perturbation vector $\bar{\mathbf{\Delta}}_{\mathbf{sub}} = [\Delta_{sub1}\ \Delta_{sub2}]^T$ for the example circuit in Figure 5.4, the perturbation vector is $\bar{\mathbf{\Delta}}_{\mathbf{pseudo}} = [\Delta_{sub1}\ \ \Delta_{sub1}\ \ \Delta_{sub1}\ \ \Delta_{sub1}\ \ \Delta_{sub2}\ \ \Delta_{sub2}\ \ \Delta_{sub2}\ \ \Delta_{sub2}]^{\mathbf{T}}$. However, this $\bar{\mathbf{\Delta}}_{\mathbf{pseudo}}$

is not guaranteed to lie in the nullspace of **T, U** (hence the subscript pseudo). The actual perturbation vector, $\overline{\mathbf{\Delta}}$, is found by projecting $\overline{\mathbf{\Delta}}_{\mathbf{pseudo}}$ into the nullspace, **U**. This is done as follows:

$$\overline{\mathbf{\Delta}} = \mathbf{U} \cdot \mathbf{U}^{\mathbf{T}} \cdot \overline{\mathbf{\Delta}}_{\mathbf{pseudo}} = \mathbf{U} \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{P}^{\mathbf{T}} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}} = \mathbf{H} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}} \tag{5.9}$$

where $\mathbf{H} \equiv \mathbf{U} \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{P}^{\mathbf{T}}$ can be thought of as a higher level (or hierarchical) null-space matrix (in analogy with Equation 3.11). Recall from Section 3.3 that **U** is a N by Q matrix, where N is the number of gates in the circuit and Q is the number of vectors in the null-space of **T**. Hence, **H** is a N by K matrix. It is easy to see that $\mathbf{T} \cdot \overline{\mathbf{\Delta}} = \mathbf{T} \cdot (\mathbf{U} \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{P}^{\mathbf{T}} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}}) = (\mathbf{T} \cdot \mathbf{U}) \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{P}^{\mathbf{T}} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}} = \mathbf{0} \cdot \mathbf{U}^{\mathbf{T}} \cdot \mathbf{P}^{\mathbf{T}} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}} = \mathbf{0}$ showing that $\overline{\mathbf{\Delta}}$ indeed lies in the null-space of **T**.

The hierarchical DAV based optimization can now be formulated as a two step procedure as follows:

(i) Inter-partition DAV based optimization: This consists of the following sub-steps:

(a) Formulate E($\overline{\mathbf{d}}$) as E'($\overline{\mathbf{\Delta}}_{\mathbf{sub}}$) = E($\overline{\mathbf{d}}_{\mathbf{init}} + \mathbf{H} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}}$).

(b) Minimize E'($\overline{\mathbf{\Delta}}_{\mathbf{sub}}$) using any global optimization algorithm. Let the minimizing vector be $\overline{\mathbf{\Delta}}_{\mathbf{sub}}^{\mathbf{min}}$.

(c) Compute $\overline{\mathbf{d}}_{\mathbf{hier}}^{\mathbf{min}} = \overline{\mathbf{d}}_{\mathbf{init}} + \mathbf{H} \cdot \overline{\mathbf{\Delta}}_{\mathbf{sub}}^{\mathbf{min}}$. $\overline{\mathbf{d}}_{\mathbf{hier}}^{\mathbf{min}}$ as the optimal delay assignment for the gates that minimizes energy at this level of hierarchy. This becomes the initial delay assignment for the next stage of optimization.

(ii) Intra-partition DAV based optimization: Initialize the optimal delay assignment vector $\overline{\mathbf{d}}^{\mathbf{min}}$ to zero for all gates. Repeat the following sub-steps for each sub-circuit 'i':

(a) Let $\bar{\mathbf{d}}^{\mathbf{init}}_{\mathbf{subi}}$ be the delays corresponding to sub-circuit 'i' in $\bar{\mathbf{d}}^{\mathbf{min}}_{\mathbf{hier}}$.

(b) Formulate E($\bar{\mathbf{d}}$) as E'($\bar{\mathbf{r}}_{\mathbf{subi}}$, V_{DDi}) = E($\bar{\mathbf{d}}^{\mathbf{init}}_{\mathbf{subi}} + \mathbf{U}_{\mathbf{subi}} \cdot \bar{\mathbf{r}}_{\mathbf{subi}}$) . All gates in sub-circuit 'i' are constrained to have supply voltage V_{DDi}. The energies of gates not in sub-circuit 'i' are included in E'($\bar{\mathbf{r}}_{\mathbf{subi}}$, V_{DDi}) also.

(c) Minimize E'($\bar{\mathbf{r}}_{\mathbf{subi}}$, V_{DDi}) using any global optimization algorithm. Let the minimizing vector be $\bar{\mathbf{r}}^{\mathbf{min}}_{\mathbf{subi}}$ and the minimizing V_{DDi} be V^{min}_{DDi} .

(d) Compute $\bar{\mathbf{d}}^{\mathbf{min}}_{\mathbf{subi}} = \bar{\mathbf{d}}^{\mathbf{init}}_{\mathbf{subi}} + \mathbf{U}_{\mathbf{subi}} \cdot \bar{\mathbf{r}}^{\mathbf{min}}_{\mathbf{subi}}$ as the optimal delay assignment for the sub-circuit gates that minimizes total circuit energy.

(e) Set the entries in $\bar{\mathbf{d}}^{\mathbf{min}}$ corresponding to the gates in sub-circuit 'i' to $\bar{\mathbf{d}}^{\mathbf{min}}_{\mathbf{subi}}$.

The optimal delay vector $\bar{\mathbf{d}}^{\mathbf{min}}$ obtained after step (ii) can be used to compute the optimal gate parameters using procedure match_delay under the condition that all gates in a sub-circuit are only allowed to use the minimizing V_{DD} for that sub-circuit.

Although the above discussion assumed energy as the cost function, any other cost function could also be used. Whatever cost function is used, it should compute the cost at the circuit level even when sub-circuits are optimized (in step (ii) above). For example, if a weighted sum of total circuit energy (normalized to the initial energy) and circuit delay (normalized to the initial delay) is used as the cost function, a sub-circuit delay assignment that reduces sub-circuit energy at the expense of increased delay might have lower cost at sub-circuit level but higher cost at circuit level. For example, if a sub-circuit delay assignment reduces sub-circuit energy by 40% but the overall circuit delay

increases by 10% (due to limited library size), the sub-circuit cost would reduce by 15% (assuming equal weightage for energy and delay) but the overall circuit cost would increase by 1% (assuming sub-circuit energy is one-fifth of circuit energy, circuit energy reduces by 8% only).

5.5 Analysis of Optimization Complexity

The complexity without partitioning is first analyzed. Firstly, at the circuit level, the number of optimization variables is Q+1, where Q (length of $\bar{\mathbf{r}}$) is the number of vectors in the null-space of the topology matrix, **T**. This assumes that only one V_{DD} is allowed for the circuit and it is an optimization variable. Q is equal to N-R, where R is the rank of **T**. In the worst case, the rank of **T** might be 1, in which case, Q is N-1. Hence, in the worst case the number of optimization variables is N. Secondly, the calculation of the cost function is O(N+E), where E is the number of edges in the circuit, since for any delay assignment, the corresponding sizes and V_{th}s for the gates can be found in one backward traversal over the circuit and the energy consumption can be found in another traversal over the circuit. Finally, if the global optimization algorithm is restricted to $O((Q+1)^2)$ ($=O(N^2)$, in the worst case) cost function evaluations to locate the global optimum, the overall complexity in the worst case is $O(N^2(N+E))$. This level of hierarchy in which the whole circuit is flatly optimized is numbered 0 in Figure 5.5. This level only utilizes the nullspace matrix, **U**, of the overall topology matrix, **T**.

Now assume the circuit is partitioned into K equal sub-circuits. The problem size (number of optimization variables) for inter-partition optimization is K (length of $\bar{\mathbf{\Delta}}_{\mathbf{sub}}$). The **H** matrix needed for inter-partition optimization at this level of hierarchy is referred

to as $\mathbf{H_1}$ in Figure 5.5. The cost function evaluation is O(N+E). Hence, the optimization complexity for inter-partition optimization is $O(K^2(N+E))$. In intra-partition optimization, the problem size for a sub-circuit 'i' is N/K in the worst case (under the assumption that the number of vectors in the null-space of $\mathbf{T_{subi}}$ scale down directly). However, the cost function calculation is still O(N+E) since the total circuit energy/delay is computed even when optimizing sub-circuits. Hence, the optimization complexity for optimizing all the partitions in intra-partition optimization is K times $O((N/K)^2(N+E)) = O(N^2(N+E)/K)$. The null-spaces of the partition topology matrices are referred to as $\mathbf{U_1}$, $\mathbf{U_2}$, etc in Figure 5.5. Thus, the overall optimization complexity with partitioning is $O(K^2(N+E)) + O(N^2(N+E)/K) = O((K^2+N^2/K)(N+E))$. To see how the complexity varies with the number of partitions, the function K^2+N^2/K is plotted in Figure 5.6 for N=100 and K varying from 1 to 100. It is seen that there is an optimum number of partitions that minimizes optimization complexity. The optimum number of partitions can be calculated by differentiating K^2+N^2/K to get $K_{opt} = 0.794N^{2/3}$ which gives the minimum value of K^2+N^2/K to be $1.89N^{4/3}$. This yields an optimization complexity with an optimum number of partitions of $O(N^{4/3}(N+E))$ which is better than the optimization complexity without partitions (Level 0). This level of hierarchy (in which the partitions of the circuit are optimized flatly) is numbered 1 in Figure 5.5.

The partitioning scheme can be carried one step further. Instead of optimizing the partitions generated in the previous step flatly (in the intra-partition optimization step), each partition can be further partitioned into sub-partitions and the sub-partitions optimized flatly (hierarchy level 2 in Figure 5.5). The optimization complexity for inter-partition optimization would still be $O(K^2(N+E))$ from above. However, the optimization

complexity for intra-partition optimization would be different now since partitions have been further split into sub-partitions. Assuming the cost function at this level is computed for the whole partition only instead of the whole circuit, the cost function calculation would be O((N+E)/K). The optimization complexity for inter-sub-partition optimization would be $O(K^2(N+E)/K)$ or O(K(N+E)). The complexity for intra-sub-partition optimization for all sub-partitions in a partition would be K times $O((N/K^2)^2(N+E)/K)$ or $O(N^2(N+E)/K^4)$. Thus, the total complexity of intra-partition optimization now would be K times ($O(K(N+E)) + O(N^2(N+E)/K^4)$) or $O((K^2+ N^2/K^3)(N+E))$. Combining this with the inter-partition complexity finally gives the overall complexity of ($O(K^2(N+E))+ O((K^2+ N^2/K^3)(N+E))$) or $O((2K^2+ N^2/K^3)(N+E))$. The function $2K^2+N^2/K^3$ is plotted in Figure 5.7 for N=100 and K varying from 1 to 100. It is observed that less number of partitions are needed *per level* at this level (Level 2 in Figure 5.5) of optimization as compared to Level 1. However, the total number of lowest level partitions went up from 17 to 36 (6^2). The optimum number of partitions can be calculated as before by differentiating $2K^2+N^2/K^3$ to get $K_{opt} = 0.891N^{2/5}$ which gives the minimum value of $2K^2+N^2/K^3$ to be $2.97N^{4/5}$. This yields an optimization complexity with an optimum number of partitions of O($N^{4/5}(N+E)$) which is better than the optimization complexity at hierarchies 0 and 1.

Thus, hierarchical DAV optimization can result in significant computational speedup. In the next section, the energy-saving/speedup tradeoff is shown for the ISCAS'85 benchmark circuits.

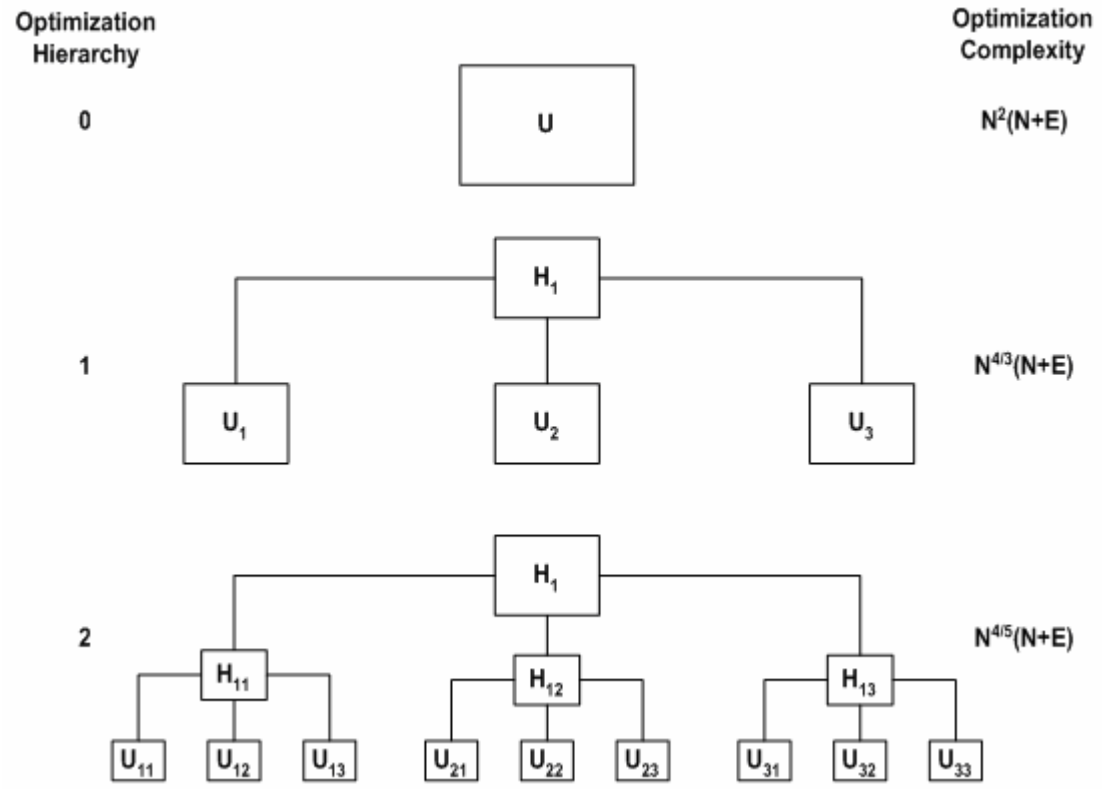


Figure 5.5. Pictorial representation of hierarchical DAV based optimization

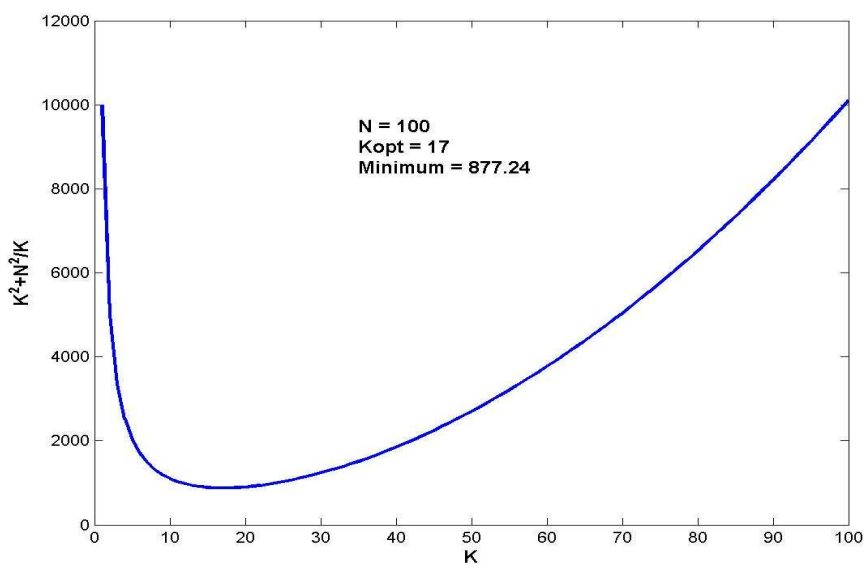


Figure 5.6. Plot of (K^2+N^2/K) versus K

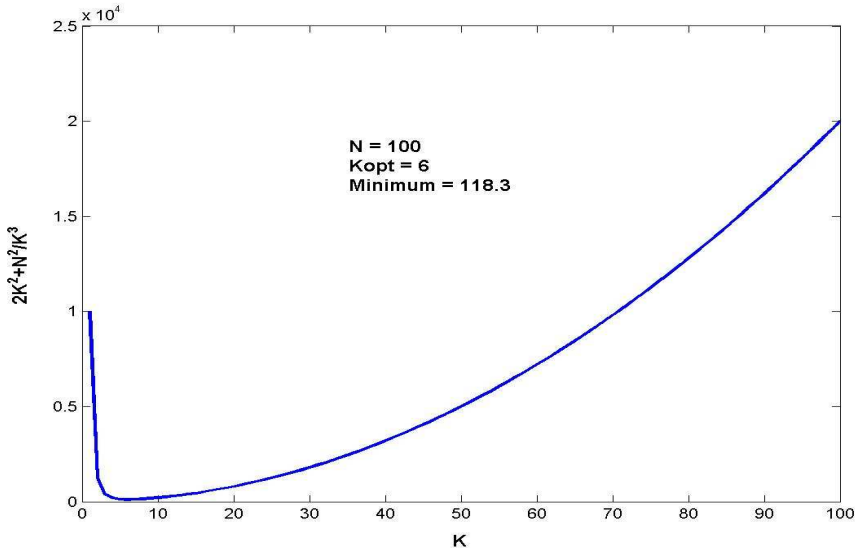


Figure 5.7. Plot of ($2K^2+N^2/K^3$) versus K

5.6 Implementation and Results

The DAV based optimization methodology was applied to ISCAS'85 benchmark circuits. Each benchmark circuit was first sized for speed using Synopsys Design Vision with a library of 2 to 4 input NAND and NOR gates, and inverters. For the optimum sizing obtained, the initial delay, dynamic and static energy consumption of the circuit was determined (with a V_{DD} of 1V, V_{th} of 0.2V for all gates and a capacitive load of 3 fF at POs) using SPICE models for a 70nm CMOS technology as described in Section 5.2. The switching activities and the static probabilities of all gate outputs (needed for the calculation of the dynamic and static energy consumption values) were also obtained using Synopsys Design Vision assuming switching activities of 0.1 and static probabilities of 0.5 at the PIs. A TSMC wire load model was used for net length estimation for different fan-outs and the distributed capacitance per unit length of local interconnects was computed taking into account coupling capacitances and ground plane

capacitances in a typical VLSI layout [60]. The width and spacing between interconnects was taken to be 0.1 micron, and the thickness of interconnect and dielectric was taken to be 0.2 micron respectively. The dielectric constant was taken to be 3.9 (SiO_2).

The optimization was carried out at hierarchy level 1 in Figure 5.5. For the ISCAS benchmark circuits, it was found that operating at this level gives good energy savings while consuming small computation time. For bigger circuits, going deeper in the hierarchy might be needed.

Two kinds of partitioning schemes were studied. In the first scheme, each benchmark circuit was partitioned using hMETIS [61] such that a minimum number of edges were cut between sub-circuits. Since, in the final optimized circuit, each sub-circuit can have different V_{DD}s, "min-cut" partitioning can minimize the number of level shifters needed. The size of each sub-circuit was kept almost equal and restricted to approximately 100 gates. Since each of the sub-circuits is optimized separately, the loads that the POs of a sub-circuit drive cannot be fixed if the sizes of the gates driven by POs (which are the PIs of the driven sub-circuit) are allowed to change during optimization. This problem was handled by fixing the sizes of the PIs of the sub-circuits at their initial values. Only the sizes of the gates that were not driven by gates in other sub-circuits were allowed to change from their initial values during the optimization.

The second partitioning scheme simply split the gates equally between the partitions in topological order. Figure 5.4 shows the example circuit "topologically" partitioned into two equal sub-circuits. The advantage of this scheme is that the partitions can also be guaranteed to be sorted topologically. If the sub-circuits are then optimized in reverse topological order, the load that the POs of a sub-circuit drive are fixed before that

sub-circuit is optimized (since the driven sub-circuits have already been optimized). Hence, the PIs of the sub-circuits do not have to be restricted to their initial values (as for min-cut partitioning). Furthermore, the V_{DD}s of the gates driven by the POs of a sub-circuit are also fixed before the sub-circuit is optimized. This allows the restriction of the V_{DD} of the sub-circuit during its optimization, to a value greater than or equal to the maximum V_{DD} of its successor sub-circuits. This, in turn, guarantees that level-shifters will not be needed in the optimized circuit. The "min-cut" and "topological" partitioning schemes represent two trade-offs present in any dual/multi-V_{DD} design. In the "min-cut" scheme, any V_{DD} can be used for a sub-circuit at the cost of additional level-shifters whereas in the "topological" scheme, no level-shifters are needed but the freedom in choosing sub-circuit V_{DD} is restricted.

Since some of the gates in the circuit have available slack after the sizing for speed, the first step in the optimization was to take up the slack so that the energy consumption of those gates could be reduced. This was done by using an "energy-aware" version of the Zero Slack Algorithm (ZSA) described in Appendix B. The slack of gates was taken up in the order of decreasing energy-slack product instead of uniformly as in the normal ZSA [55]. This allowed gates with high energy consumption to be slowed down while also reducing the number of iterations to get to the zero slack state. The V_{DD}s of the sub-circuits were kept at their initial values (1V) during this phase while the sizes and V_{th}s of the gates were chosen to match the new delays using the backward traversal method described in the previous section. Note that most optimization algorithms stop after this step i.e. after having taken up the slack. However, the results obtained after this step might still be far from optimal because there are innumerable ways to take up slack

out of which only one is the best. Another way to look at DAV based optimization is that it basically searches efficiently for the best way to take up slack amongst all possible ways.

The second step used the delay assignment of the gates after the ZSA step as input and found the optimal perturbation that minimizes the energy consumption. The global optimization method used to minimize the cost function was multi-level co-ordinate search (MCS) [62].

Tables 5.1 and 5.4 give the optimization results for ISCAS'85 benchmark circuits for the two different partitioning schemes studied. For the DAV optimization, all (continuous) sizes between 1 and 20 were assumed available. All voltages between 0.8V and 1.2V were assumed available for use as sub-circuit V_{DD}s. At the end, a post-processing step quantizes the sizes to a resolution of 0.2 and clusters the V_{DD}s into a maximum of 3 different values (to take into account a finite sized library and limited numbers of V_{DD}s allowed in practice). The threshold voltages allowed were 0.2V and 0.3V. Column 1 gives the circuit name, the number of gates in the circuit and the number of sub-circuits each circuit was partitioned into. Columns 2 and 3 give the initial delay and energy consumption of the circuit respectively. Columns 4 and 5 give the energy decrease and delay increase after the first step (ZSA) of the optimization. As mentioned before, in this step, all the sub-circuits were kept at the initial voltage of 1V while the slack of the gates was taken up (by decreasing the size and/or increasing V_{th}). Columns 6 and 7 give the total energy decrease and the delay increase after the second step (DAV). The DAV step assumed perfect level-shifters with zero delay and energy overheads. The first sub-columns of the 8th and 9th columns give the energy decrease and delay increase

respectively if the DAV-optimized sub-circuits are connected together without any level-shifters. These numbers show that there is not a very significant effect of low-voltage gates driving high-voltage gates (at sub-circuit boundaries) on the total circuit energy and circuit delay. This is due to the fact that there is little difference between the sub-circuit V_{DD}s obtained after DAV optimization (as shown in Tables 2 and 4). The second sub-columns of the 8th and 9th columns show the energy decrease and delay increase after quantizing the sizes to a resolution of 0.2 and clustering the sub-circuit V_{DD}s to 3 values. Clustering was done simply by grouping together V_{DD}s close to each other into their mean value. After clustering, the high V_{DD} gates that were driven by low V_{DD} gates were given a higher threshold voltage to reduce their leakage power. This is shown to be a better trade-off than inserting level-shifters, when the voltage difference is small [63]. The third sub-columns of the 8th and 9th columns show the effect of this on the energy decrease and delay increase of the circuit. Tables 5.2 and 5.3 give some statistics of the initial and final circuits. The final sub-circuit V_{DD}s used (after clustering) are also given.

As mentioned before, while most methods stop after taking up slack using some energy-aware heuristic (the one in Appendix B being one example heuristic), DAV based optimization searches for the best way to take up slack. This is how it is able to reduce energy beyond what other methods can give. For the ISCAS benchmark circuits, the energy savings after the ZSA step is 24% on the average. This represents the energy savings possible by using a heuristic to take up slack. DAV based search is however able to give 36% (using min-cut partitioning) on the average. The ZSA method in Appendix B has complexity O(N(N+E)) in the worst case since it could potentially do N calculations of the circuit slack before stopping and every slack calculation is O(N+E). The DAV

based optimization, on the other hand, is O($N^{4/3}(N+E)$) (with optimum number of partitions), which is only marginally higher.

Finally, to compare the speed-up obtained due to partitioning, the smallest ISCAS benchmark circuit 'c432' was optimized flatly (hierarchy level 0) and with 4 partitions (hierarchy level 1). Other benchmarks were too big too be optimized flatly in a reasonable time frame. For 'c432', it was found that the energy saving at level 0 was 39.1% with a 1.7% delay overhead. This can be compared to the results for level 1 in Tables 5.1 and 5.4. With min-cut partitioning, energy saving at level 1 was 37.3% with 2% delay overhead. With topological partitioning, energy saving at level 1 was 36.3% with 1.1% delay overhead. However, the optimization time at level 1 was 1000 seconds while the optimization time at level 0 was 20000 seconds, giving a 20X speedup. Note that the optimization assumes only one V_{DD} per partition. Hence only one V_{DD} was used for the whole circuit at level 0 whereas at level 1, each partition had a separate V_{DD}. The availability of extra V_{DD}s at level 1 seems to be able to compensate for the loss due to hierarchy.

Figure 5.8 gives the tradeoff between computational speedup and EDP savings at hierarchy levels 1 and 2 for the ISCAS circuits and a 16 bit ALU. The number of partitions that were optimized flatly was kept the same for both levels. For example, the circuit 'c2670' was optimized using 16 partitions at level 1 and 4 partitions (each with 4 sub-partitions) at level 2. At hierarchy level 2, only the partitions at level 1 were allowed to have different V_{DD}s (i.e. all the sub-partitions of a partition had the same V_{DD}). It is seen that there is significant reduction in optimization time at level 2 compared to level 1. Furthermore, the EDP savings obtained at level 2 are very close to those obtained at level

1. In fact, for some cases, the savings obtained at level 2 are more than the savings obtained at level 1. This is because there are a lot more V_{DD}s used at level 1 compared to level 2 (for example 16 at level 1 and 4 at level 2 for 'c2670') and this leads to some EDP degradation at level 1 due to V_{DD} clustering and level-shifting.

Figures 5.9 and 5.10 give the optimization times at levels 1 and 2 for the ISCAS circuits. It is seen that the optimization times at level 1 track the optimal complexity calculated in Section 5.5 closely (even though the number of partitions was not chosen to be K_{opt} in the experiments). The optimization times at level 2 do not track so closely. This is probably due to the fact that the implementation of the optimization algorithm at level 2 (in MATLAB and the C programming language) had complexity O(N+E) and not O((N+E)/K) as required in Section 5.5.

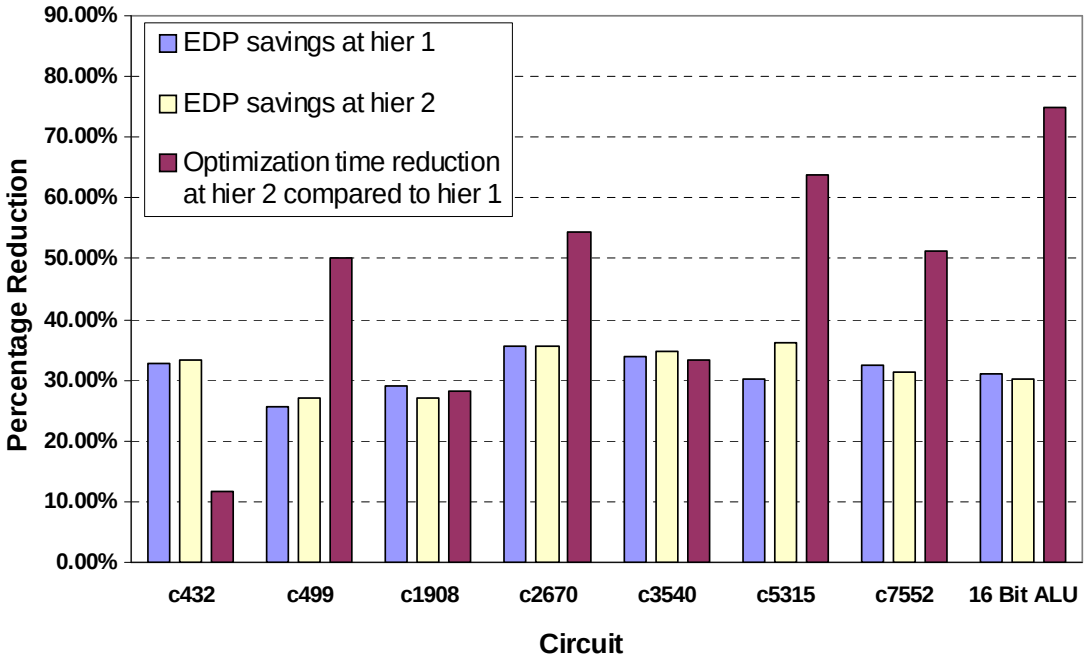


Figure 5.8. Comparison of EDP savings and optimization time reduction between hierarchy 1 and hierarchy 2

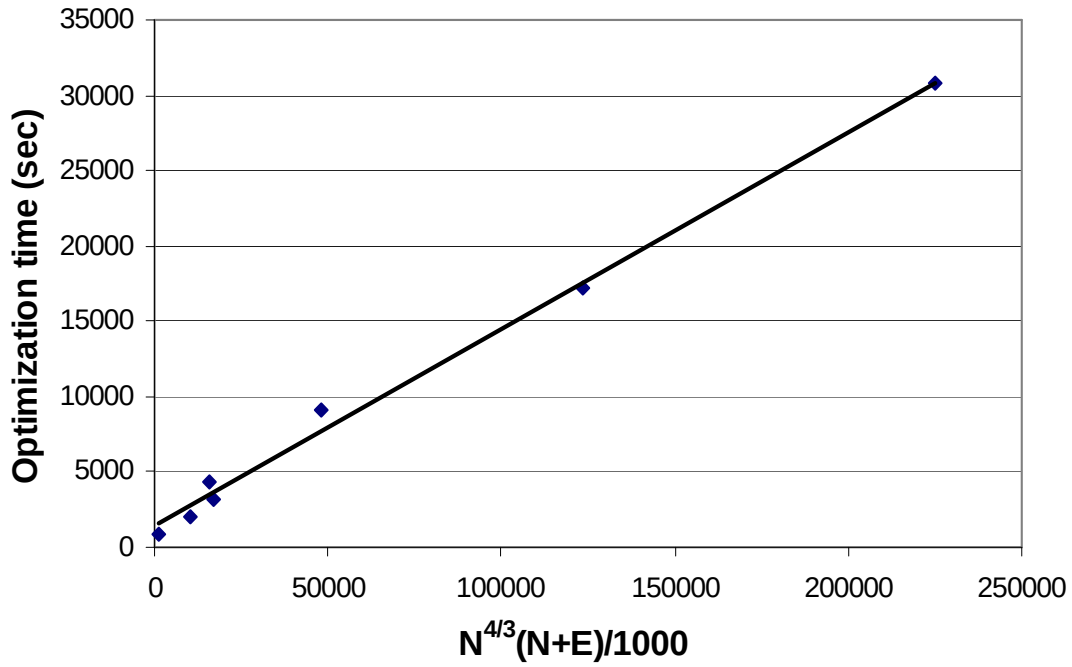


Figure 5.9. Plot of optimization times for ICSAS benchmark circuits (at hierarchy 1) versus $N^{4/3}(N+E)$

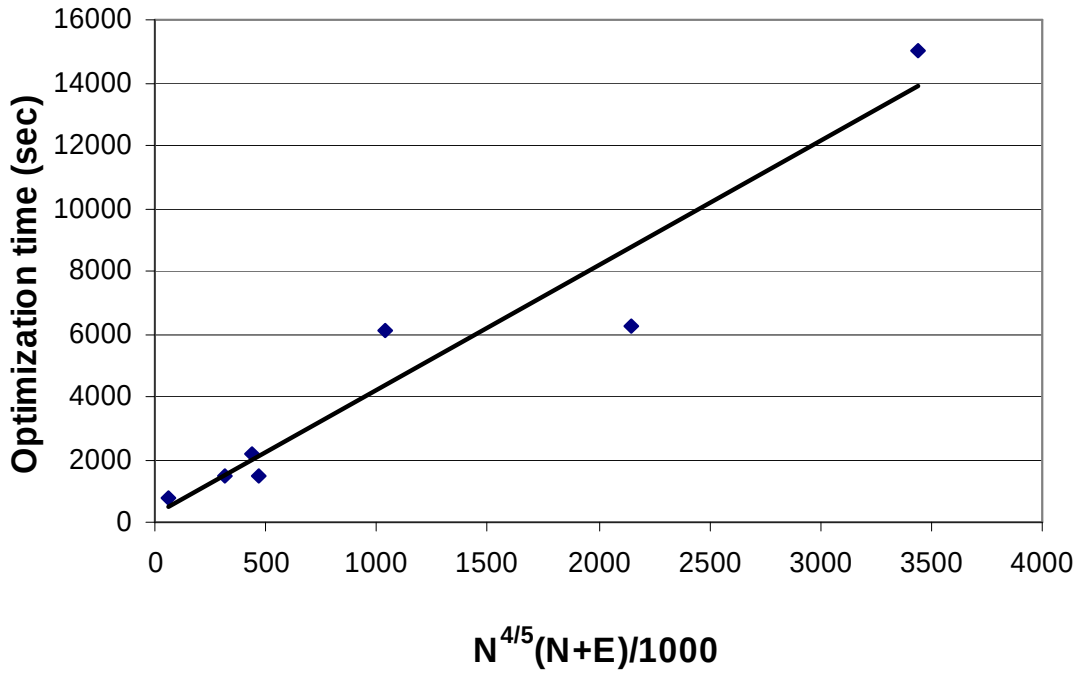


Figure 5.10. Plot of optimization times for ICSAS benchmark circuits (at hierarchy 2) versus $N^{4/5}(N+E)$

Table 5.1. Optimization Results (min-cut partitioning)

Circuit (#Gates, #Sub-circuits)	Initial Delay (ps)	Initial Energy (fJ)	% Energy Decrease (ZSA)	% Delay Increase (ZSA)	% Energy Decrease (DAV)	% Delay Increase (DAV)	% Energy Decrease (Final)			% Delay Increase (Final)		
c432 (267,4)	457	94.6	24.4	1.2	36.1	1.7	34.3	34.3	37.3	1.7	1.9	2
c499 (835,8)	351	351	24.3	1.6	29.7	1.7	29.5	29.7	30	1.7	2.3	2.3
c1908 (680,8)	523	262	21.8	1.25	28.1	4.3	26	26.2	27.9	4.4	4.5	4.5
c2670 (875,8)	326	333	23.7	1.7	38.3	1.5	36.6	37.4	42.1	1.7	1.9	2.9
c3540 (1319,8)	632	528	25.6	1.4	34.5	0.8	31.4	31.4	38.4	0.8	1.3	1.3
c5315 (1994,16)	510	784	25.3	0.7	34	0.4	29.4	34.2	40.8	4.6	3.9	3.9
c7552 (2538,16)	444	1088	21.8	1.9	32.7	2	30.9	29.4	35.1	2.4	2	2.5
Average	463	492	23.8	1.4	33.3	1.8	31.2	31.8	35.9	2.5	2.5	2.8

Table 5.2. Circuit Statistics (min-cut partitioning)

Circuit	Initial Circuit			Final Circuit			% Decrease in Circuit Size	% High V_{th} gates	Sub-circuit V_{DD}s (V)
	Min Size	Mean Size	Max Size	Min Size	Mean Size	Max Size			
c432	1	2.35	6	1	1.41	6	40.2	56.9	0.96, 0.98, 1
c499	1	2.52	6	1	1.65	6	34.7	25.6	1
c1908	1	2.23	6	1	1.68	6	24.5	47.5	0.93, 0.99, 1
c2670	1	2.6	6	1	1.54	6	40.9	67.1	0.85, 0.99, 1.01
c3540	1	2.25	6	1	1.58	6	29.8	68.6	0.89, 0.98, 1
c5315	1	2.27	7	1	1.54	9.6	32.3	71.6	0.82, 0.96, 1.14
c7552	1	2.24	8	1	1.57	8	29.8	71.2	0.88, 0.92, 1.01
Average	1	2.35	6.43	1	1.57	6.8	33.17	58.36	-

Table 5.3. Circuit Statistics (topological partitioning)

Circuit	Initial Circuit			Final Circuit			% Decrease in Circuit Size	% High V_{th} gates	Sub-circuit V_{DD}s (V)
	Min Size	Mean Size	Max Size	Min Size	Mean Size	Max Size			
c432	1	2.35	6	1	1.45	6	38.5	58.4	0.99
c499	1	2.52	6	1	1.71	6	32.2	24.8	1
c1908	1	2.23	6	1	1.53	9.2	31.6	68.4	1.05,1.2
c2670	1	2.6	6	1	1.56	6	40	67.7	0.97,1
c3540	1	2.25	6	1	1.62	11.2	28.1	71	1
c5315	1	2.27	7	1	1.49	6.6	34.3	81.2	0.98,1.01,1.05
c7552	1	2.24	8	1	1.54	6.4	31	74.3	1,1.02,1.04
Average	1	2.35	6.43	1	1.56	7.34	33.7	63.7	-

Table 5.4. Optimization Results (topological partitioning)

Circuit (#Gates, #Sub-circuits)	Initial Delay (ps)	Initial Energy (fJ)	% Energy Decrease (ZSA)	% Delay Increase (ZSA)	% Energy Decrease (DAV)	% Delay Increase (DAV)	% Energy Decrease (Final)			% Delay Increase (Final)		
c432 (267,4)	457	94.6	24.4	1.2	33	0.4	33	33.7	36.3	0.4	1	1.1
c499 (835,8)	351	351	24.3	1.6	26.8	1.1	26.8	26.1	26.6	1.1	1.8	2.1
c1908 (680,8)	523	262	21.8	1.2	8.2	2.3	8.3	8.3	11.3	2	2.5	2.5
c2670 (875,8)	326	333	23.7	1.7	33.2	0.9	33.2	33.2	36.9	0.9	1.8	2.8
c3540 (1319,8)	632	528	25.6	1.4	31	0.2	30.9	31	36.4	0.2	0.8	1.1
c5315 (1994,16)	510	784	25.3	0.7	26.8	0.7	25.4	27.7	34.5	0.4	0.7	1.2
c7552 (2538,16)	444	1088	21.8	1.9	25.5	1.1	25.5	25.1	30.8	1	1.2	2
Average	463	492	23.8	1.4	26.4	1.0	26.2	26.4	30.4	0.9	1.4	1.8

5.7 Conclusion

This chapter presented a technique for modeling delay and energy consumption of CMOS combinational circuits using SPICE look-up tables. A method was presented for using DAV based optimization to minimize the energy consumption of CMOS circuits at the gate level. A hierarchical method was developed to handle large netlists efficiently. It was shown that DAV based optimization can yield 12% better energy savings compared to traditional slack based optimization approaches. The computational complexity of DAV based optimization was shown to be only marginally higher than traditional methods.

Chapter VI

Gate Level Soft-Error Optimization

6.1 Introduction

Technology scaling roughly leads to a doubling of clock frequencies every generation, a 30% decreases in node capacitances every generation and a 30% reduction in supply voltages to reduce power consumption. All these factors are leading to a drastic increase in soft-error susceptibility of combinational and memory circuits to alpha-particle and neutron strikes. Because of the reduced node capacitances, a smaller injected charge is needed to induce a glitch at a circuit node. Thus, low-energy particle strikes that earlier had no effect on a circuit can now cause soft-errors. Because of the reduced supply voltages, noise margins are reduced, which also increases the susceptibility to particle strikes. Increasing clock frequencies increase the probability of a soft-error getting latched. Furthermore, due to super-pipelining, the number of gates in pipeline stages have been reducing, which in turn reduces the electrical attenuation of glitches as they propagate to the latches.

Although these factors affect both memory and combinational elements, the overall soft-error rate of memories is not increased as much as combinational logic because memories are protected by techniques such as error-correcting codes (ECC). There has not been a need to protect combinational circuits because combinational circuits have a natural tendency to mask glitches due to three phenomena [64]. First, due to *logical masking*, a glitch might not propagate to a latch because of a gate on the path

not being sensitized to facilitate glitch propagation. Second, due to *electrical masking*, a generated glitch might get attenuated because of the delays of the gates on the path to the output. Third, due to *latching-window masking*, a glitch that reaches the primary output might not still cause an error because of the latch not being open. The factors mentioned in the previous paragraph adversely affect all the above three factors in terms of soft-error tolerance. Due to decreasing number of gates in a pipeline stage, logical masking as well as electrical masking has been decreasing for new technology generations. Electrical masking has also been decreasing due to the reduction in node capacitances and supply voltages every generation. Furthermore, increasing clock frequencies have reduced the time window in which latches are not accepting data, thereby reducing latching-window masking also. Because of these factors, the soft-error rate (SER) of combinational logic is expected to rise 9 orders of magnitude from 1992 to 2011, when it will equal the SER of unprotected memory elements [4].

Generally, in mission-critical space applications combinational circuits are protected by using duplication/triplication and concurrent-error detection (CED) [46]. However, these methods have too high delay, area and power overheads to be used in commercial applications. Recently, low-cost methods for increasing soft-error tolerance of commodity applications using time-redundancy [49] and partial duplication [50] have been proposed. However, these methods still add additional delay overhead to the original circuit due to the presence of the checker circuit. Also, these methods have system level overheads (such as pipeline flushes) when an error is detected, either to correct the error or to do the computation again.

This chapter proposes a novel, low delay-overhead method for increasing the soft-error tolerance of nanometer CMOS combinational logic circuits using DAV based optimization. Using an optimal assignment of supply voltages, threshold voltages and sizes to gates and by adding optimal amounts of capacitive loads at the primary outputs, the electrical attenuation characteristics of the gates in the circuits are enhanced without incurring significant delay overhead. Multi-supply voltage and multi-threshold voltage designs are becoming increasingly common for low-power applications, however if these are infeasible, the method can still be used to just find optimal gate sizings for increased soft-error tolerance using the specified supply and threshold voltage. This method can be used along with any of the traditional methods described above to greatly decrease the overhead of error detection and correction.

The chapter is organized as follows. Section 6.2 describes characteristics of gates that affect the strike-induced glitches. Section 6.3 describes ASERTA, a tool for fast and accurate analysis of the soft-error tolerance of a circuit. Section 6.4 describes SERTOPT, a circuit optimization tool for enhancing the soft-error tolerance of circuits while meeting timing constraints. Section 6.5 gives experimental results. Section 6.6 concludes.

6.2 Glitch tolerance characteristics of individual gates

There are two characteristics of interest for a single gate in terms of soft error tolerance: *glitch generation* and *glitch propagation*. The glitch generation characteristics of a logic gate determine the shape and magnitude of the voltage glitch generated at the output of the gate due to a particle strike on the gate. The glitch propagation characteristics of a logic gate determine how the gate attenuates a glitch that is generated at some prior circuit node as it passes through the logic gate.

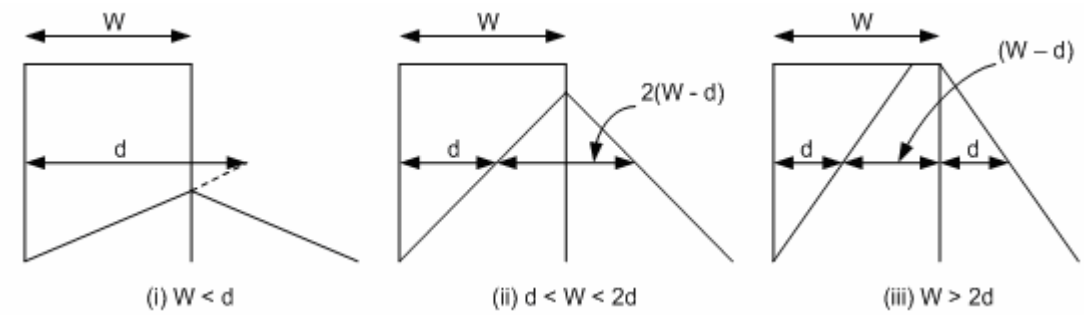


Figure 6.1. Input and output signals for different gate delays

When a particle strikes a circuit node, the voltage magnitude of the corresponding glitch is dependent on the total capacitance of the node. The duration of the generated glitch is dependent on the delay of the gate that is driving the node. If the gate driving the node is fast, it will quickly discharge (or charge) the node back to its original value. Therefore, faster gates have better glitch generation characteristics in terms of the generated glitch width. However, the behaviour is *opposite* for glitch propagation. Assuming a linear ramp at the output of the gate, for a gate propagation delay of d and glitch duration of w_i at the gate input, glitch duration at the output of the gate, w_o, can be approximated as follows (see Figure 6.1):

$$\begin{aligned} w_o &= 0 \;\; if \;\; w_i < d \\ w_o &= 2 \cdot \left(w_i - d \right) \;\; if \;\; d < w_i < 2 \cdot d \\ w_o &= w_i \;\; if \;\; w_i > 2 \cdot d \end{aligned} \tag{6.1}$$

This model is similar to the glitch amplitude attenuation model used in [65]. As seen from Equation 6.1, a slow gate will attenuate a glitch at its output more compared to a fast gate. Therefore, slow gates have better glitch attenuation characteristics.

Increasing a gate's output capacitance increases the delay of that gate. This makes the glitch attenuation characteristics of that gate better. Furthermore, if the capacitance is large enough, the particle may not have enough energy to create enough voltage

fluctuation for an error. So, a large enough output capacitance may improve both glitch generation and glitch propagation characteristics of a gate at a cost of increased gate delay.

Figures 6.2 and 6.3 show SPICE simulation results for generated glitch width and propagated glitch width, respectively, for an inverter for different values of gate size, gate supply voltage (V_{DD}), gate threshold voltage (V_{th}) and gate load (C_{load}). The SPICE models are for 70nm technology node [59]. The minimum and maximum values of the variables are indicated on the x-axis. Size of 1 means a gate width of 100nm. It is clear that if the output load is kept constant, the factors that slow down a gate (decrease in size, reduction in V_{DD}, and increase in V_{th}) increase generated glitch width but also increase the attenuation of propagating glitches. The generated glitch width first increases with output capacitance, then it starts to decrease. This behaviour is explained as follows: If

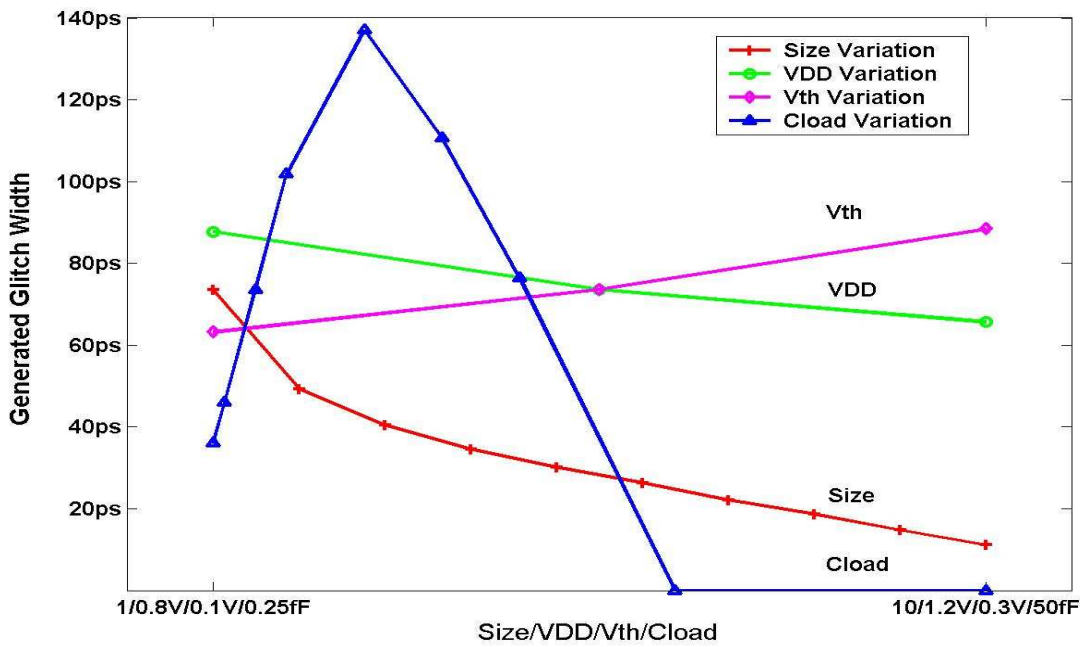


Figure 6.2. Glitch generation characteristics for an inverter for an injected charge of 16fC.

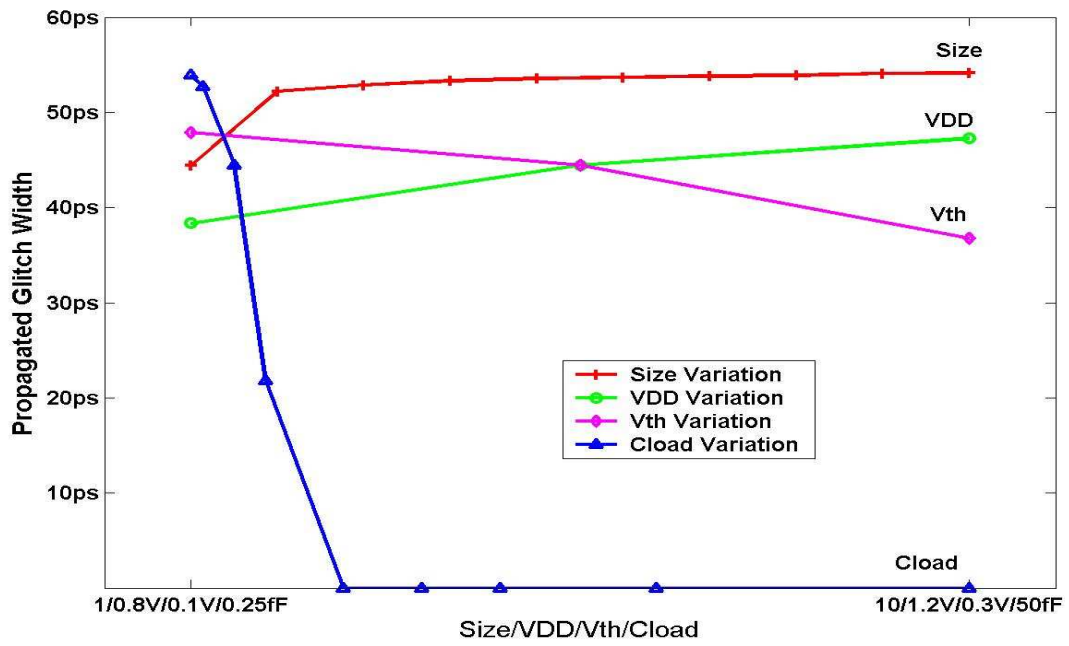


Figure 6.3. Glitch propagation characteristics of an inverter for an input glitch of duration 50ps.

the capacitance is small, the voltage generated at the gate's output is clipped by the diode between the source and the body of the transistor. For this case, smaller capacitance will hold less charge for the same voltage (Q = CV), making the discharge (recharge) time faster. This initially results in larger glitch widths for increasing values of output capacitance. However, if the output capacitance is large enough, the magnitude of the generated voltage glitch will reduce, and eventually become too small to cause an error.

There are two insights gained from the SPICE simulation data. First, only generated glitch width or propagated glitch width are not enough to characterize the "softness" of a gate as this might lead to erroneous conclusions. If only glitch propagation characteristics are considered as a measure of the "softness" of a gate (as in [66]), slowing down a gate would apparently always reduce the softness of the circuit; however, a slower gate will produce a bigger glitch at its output when it is subjected to a

particle strike. Such a glitch can easily propagate to the output and cause an error. The circuit must be considered as a whole and any soft error tolerance enhancement scheme should consider both glitch generation and glitch propagation characteristics of the gates as well as their location in the circuit.

The second insight is that the soft-error tolerance of a combinational circuit can be increased by increasing the capacitive loads of the gates at the primary outputs (POs) as this will attenuate all glitches reaching the POs (see propagated glitch width variation with C_{load} in Figure 6.3). The capacitive load should be increased beyond the critical point (peak in C_{load} curve in Figure 6.2) so that the generated glitch width at the POs is also small. However, a delay penalty will be incurred due to the increased load at the POs. This can be offset by appropriate selection of sizes, V_{DD}s and V_{th}s for the gates in the circuit as described in Section 6.4. The next section describes ASERTA, a tool for accurate estimation of the soft-error tolerance of a circuit.

6.3 Circuit soft-error tolerance analysis

ASERTA models a particle strike at a node as a current source injecting (or removing) a fixed amount of charge into (or from) that node. If the node is at low voltage, charge is injected into the node and if the node is at high voltage, charge is removed by the current source. The opposites of these two cases do can not cause a voltage glitch to be generated and are neglected. A SPICE look-up table is constructed for generated glitch width (due to charge injected at gate output) for different types of gates, fan-ins, sizes, channel lengths, V_{DD}s, V_{th}s and load capacitances.

The rising and falling behavior of the current waveform generated due to a particle strike is given by the following equation [67]:

$$\mathrm{I}(t) = \frac{2}{T \cdot \sqrt{\pi}} \cdot \sqrt{\frac{t}{T}} \cdot \exp\left(\frac{-t}{T}\right) \tag{6.2}$$

where T is a process dependent parameter. However, for simplicity, the current waveform in ASERTA is approximated as a trapezoid with duration of 24ps and with rise time and fall time of 4ps and 16ps respectively.

SPICE look-up tables are also constructed for delays, static energies, dynamic energies, output ramp and gate input capacitances for different types of gates, fan-ins, sizes, V_{DD}s, V_{th}s, input ramps and load capacitances as described in Section 5.2. ASERTA uses linear-interpolation inside the look-up tables to compute output values for arbitrary values of input parameters. Using look-up tables allows ASERTA to have better accuracy than analytical models while still being much faster than SPICE. To estimate the soft-error tolerance of a circuit, ASERTA injects charge into every gate output, looks-up the generated glitch width from the table and then propagates the generated glitch to the primary outputs (POs) taking into account the effects of logical and electrical masking. The sum total of the widths of the glitches reaching the POs is taken as a measure of the "Unreliability" of the circuit. The amount of charge injected (or removed) actually depends on the energy of the strike E and the probability of getting a strike of energy E falls of exponentially with E [68]. To model this, 3 different charge amounts (corresponding to 3 different strike energies) are actually injected (by varying the amplitude of the trapezoidal current waveform) at every node and the propagated glitches at the POs due to the 3 different charges are combined by weighting with the probability of that charge injection taking place. To model glitch generation and propagation, glitch width is used alone instead of a combination of width and amplitude since it was found

that the latter did not lead to enough increase in accuracy to justify the added complexity. The following sub-sections describe how ASERTA models logical, electrical and latching-window masking.

6.3.1 Logical masking

Since actual signal values are not known, for every node ASERTA calculates the probability that there is at least one sensitized path from that node to a primary output. Calculation of the sensitization probability values from the input signal statistics is easy for circuits which do not have reconvergent fan-out. Sensitization probabilities for such circuits can be calculated as in [66]. However, finding the values for circuits with reconvergent fan-out is an NP-complete problem [69]. ASERTA uses zero delay simulation of the circuit with 10000 random inputs applied (as in [50]) to compute the probability, P_{ij}, that there is at least one path sensitized from output of gate i to primary output j. For primary output j, P_{jj} is 1. The static probability, p_i, of a node i being at logic 1 is obtained for all nodes using a commercially available tool, Synopsys Design Compiler, given a static probability of 0.5 at the primary inputs.

For all successor gates s of gate i, the probability that a glitch at i will be able to propagate through gate s to primary output j is calculated as follows:

$$\pi_{isj} = \frac{S_{is} \cdot P_{ij}}{\sum_{k \in \Psi} S_{ik} \cdot P_{kj}} \tag{6.3}$$

where Ψ is the set of successors of gate i and S_{is} is the probability that gate s is sensitized to gate i (i.e. all other inputs of gate s have non-controlling values). S_{is} can be obtained by multiplying together the static probabilities of the other inputs being 1/0 for a AND/OR gate. Note that π_{isj} is not taken to be just $S_{is} \cdot P_{sj}$ since π_{isj} should have the property that

$\sum_{k \in \Psi} \pi_{ikj} \cdot P_{kj} = P_{ij}$. Also note that π_{isj} is an approximation to the actual probability value since in circuits with reconvergent fan-out, the probability that gate s is sensitized to gate i conditions the probability of gate s having a path sensitized to a primary output.

The next sub-section describes the procedure used in ASERTA for computing the glitch widths at POs for charge injected at every gate output.

6.3.2 Electrical masking

As mentioned before, ASERTA computes the expected output glitch width, W_{ij}, at primary output j for generated glitch width, w_i, at gate i. To do this efficiently in one pass over the circuit, for every gate, the expected output glitch widths, WS_{ijk}, for 10 sample glitch widths, ws_k (k between 1 and 10) are computed.

The output glitch widths are computed for all gates in reverse topological order (i.e. from POs to PIs) as follows:

(i) Let current gate be i.

(ii) If gate i is a primary output, set $WS_{iik}=ws_k$ for all k.

Set $WS_{ijk}=0$ for all other primary outputs j.

Also, since gate is primary output, it will propagate generate glitch width, w_i, directly. Hence, set $W_{ii}=w_i$ and $W_{ij}=0$ for all other primary outputs j.

(iii) If gate i is not a primary output, for all sample glitch widths, ws_k:

For all successors s of gate i:

Let d_s be the delay of gate s looked up from the SPICE tables.

Calculate the glitch width, wo_{sk}, propagated to the output of gate s for input width of ws_k using Equation 6.1.

For each primary output j, look up the expected output glitch width, WE_{sjk}, for generated glitch width of wo_{sk} from the table of expected output glitch widths for gate s, linearly interpolating if necessary.

Finally, Let $WS_{ijk} = \sum_{s \in \Psi} \pi_{isj} \cdot WE_{sjk}$

(iv) Compute W_{ij} by looking up the table of expected output glitch widths, WS_{ijk}, computed in step (iii), for a generated glitch width of w_i, again linearly interpolating if necessary. Now process the next gate.

At the end of this procedure, expected output glitch widths, W_{ij}, at primary output j for generated glitch width, w_i, for every gate i are known. The complexity of the procedure is O(V+E), where V is the number of gates and E is the number of circuit edges.

Lemma 1: For a very wide glitch ww_i generated at output of gate i, the above procedure correctly computes the expected output glitch width at primary output j as $WW_{ij} = ww_i \cdot P_{ij}$, if it is assumed that ww_i is one of the sample glitch widths used above (say sample 1).

Proof: Since the generated glitch is very wide, it will pass through all gates on any path from i to j without attenuation. WS_{jj1} is correctly computed as ww_i at primary output j. Assume that WS_{rj1} is correctly computed for all successor gates r of a gate p between i and j as $ww_i \cdot P_{rj}$. Then, the expected width WS_{pj1} will be computed as:

$$WS_{pj1} = \sum_{r \in \Psi} \pi_{prj} \cdot WS_{rj1} = \sum_{r \in \Psi} \pi_{prj} \cdot ww_i \cdot P_{rj}$$
$$= ww_i \cdot \sum_{r \in \Psi} \pi_{prj} \cdot P_{rj} = ww_i \cdot P_{pj}$$

where WS_{rj1} can be used instead of WE_{rj1} because ww_i is wide enough to propagate through gate r without attenuation. By induction, WS_{ij1} is also computed as $ww_i \cdot P_{ij}$. Since ww_i is the first sample glitch width, WS_{ij1} is WW_{ij}. □

6.3.3 Latching-window masking

A glitch must arrive within the setup and hold times of the latch at the primary output to be captured. Since the exact time of the particle strike is unknown, it can be assumed to be uniformly distributed within the clock cycle. The probability of a glitch being captured by a latch is directly proportional to its duration. Hence, by summing up the expected output glitch widths, W_{ij}, for all primary outputs j, the total contribution of gate i to the circuit unreliability is obtained. However, this ignores the fact that the size, Z_i of a gate plays an important role in determining the particle flux incident on the gate. Hence, the actual contribution of gate i to circuit unreliability is:

$$U_i = Z_i \cdot \sum_j W_{ij} \tag{6.4}$$

The total unreliability of the circuit is:

$$U = \sum_i U_i \tag{6.5}$$

Note that although the unreliability of a circuit becomes worse if its clock frequency is increased (due to technology scaling), the above definition of unreliability can ignore clock frequency because it is only used to compare designs with the same clock frequency.

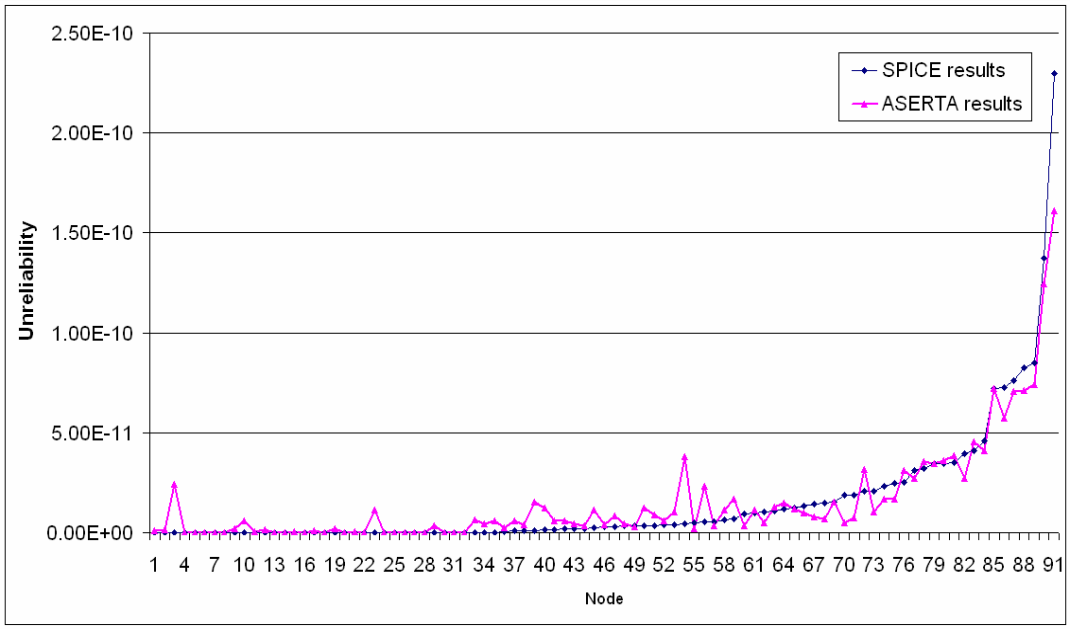


Figure 6.4. Unreliability values obtained by SPICE and ASERTA for nodes in c432.

Figure 6.4 shows the unreliability numbers, U_i, for the gates in benchmark circuit "c432" calculated by ASERTA plotted along with values calculated by SPICE for 70nm technology node. In SPICE, the unreliability was computed by applying 50 random input vectors, injecting charge at every gate output i and using the width of the glitch at primary output j as W_{ij} in Equation 6.4. Only the nodes that were at most five levels deep from the POs are plotted. It is seen that there is close matching. The correlation between the two series was computed to be 0.96. For the ISCAS'85 benchmark circuits, an average correlation of 0.9 was obtained.

The next section describes SERTOPT, a tool that uses the unreliability estimates generated by ASERTA to optimize nanometer circuits for increased soft-error tolerance by enhancing the electrical masking characteristics of gates in the circuits.

6.4 Circuit soft-error tolerance optimization

SERTOPT uses DAV based optimization to minimize a cost function that is a weighted sum of a circuit unreliability metric and circuit energy-delay product (EDP). A designer can easily change the optimization trade-offs by changing the ratio of the weights.

To find the circuit parameters (gate sizes, V_{DD}s, V_{th}s, loads) that are needed to match a delay assignment, SERTOPT traverses the circuit from POs to PIs in reverse topological order. For the gates at the POs, different capacitive loads yield the required delay for different values of size, V_{DD} and V_{th}. From these different sets of load, size, V_{DD} and V_{th} values, the set that yields the lowest value of generated glitch width is chosen. Once the parameters (load, size, V_{DD}, V_{th}) of the PO gates have been set, the capacitive loads offered by these gates to their predecessors can be found. For the internal gates (not primary outputs), appropriate values of size, V_{DD} and V_{th} are chosen that match the assigned delay while minimizing the energy-delay product (ELP) for the gate and its predecessors. The whole procedure is repeated until the primary inputs (PIs) are reached. The procedure is similar to the one in Figure 5.3 with the exception that PO load is also an additional variable now.

Once the circuit parameters (sizes, V_{DD}s, V_{th}s) have been determined, the circuit delay, energy (static + dynamic) and the EDP can be computed using SPICE libraries. The cost of the current delay assignment is computed as:

$$C = w_1 \cdot \left(\frac{EDP}{EDP_{init}} - 1 \right) + (1 - w_1) \cdot \left(1 - \frac{POD^{avg}}{POD_{init}^{avg}} \right) \tag{6.6}$$

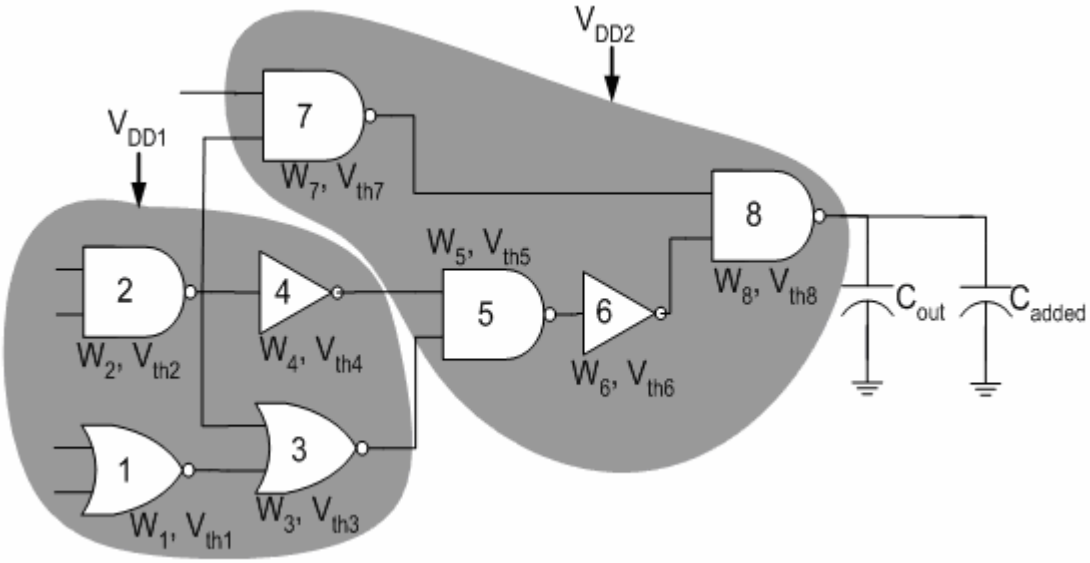


Figure 6.5. Example combinational circuit partitioned into 2 sub-circuits

where w_1 is the weight ($0 \leq w_1 \leq 1$) and EDP_{init} is the initial circuit EDP. POD^{avg} is the average primary output delay. The average POD is tried to be maximized to reduce glitch propagation through the POs. Matching the bigger average PO delay using big capacitances also reduces the glitch generation at the POs. POD^{avg} can be considered as a circuit reliability metric, with bigger values indicating a more reliable circuit.

The cost is minimized by using Multi-level Co-ordinate Search (MCS) to search for the optimal delay assignment giving lowest cost. However, simulated annealing, genetic algorithms or some other optimization algorithm can also be used.

The issues related to problem size and level-shifters (discussed in Section 5.3) are tackled similar to the approach in Section 5.4. The only major difference is the usage of additional capacitive loads to match the delays at the primary outputs as shown in Figure 6.5.

6.5 Experimental results

As in Section 5.6, gate sizes were first obtained for ISCAS'85 benchmark circuits by optimizing for speed using Synopsys Design Compiler. The gate sizes were then used with SPICE 70nm models [59] to compute the delays of the circuits for the 70nm technology. All the gates had a transistor channel length of 70nm, V_{DD} of 1V and V_{th} of 0.2V. The unreliability of the baseline circuits was estimated using ASERTA. A TSMC wire load model was used for net length estimation for different fan-outs and the distributed capacitance per unit length of local interconnects was computed taking into account coupling capacitances and ground plane capacitances in a typical VLSI layout. The width and spacing between interconnects was taken to be 0.1 micron, and the thickness of interconnect and dielectric was taken to be 0.2 micron respectively. The dielectric constant was taken to be 3.9 (SiO_2).

Then, SERTOPT was used to determine new gate sizes, V_{DD}s and V_{th}s (and added capacitances at the POs) for the circuits that would minimize unreliability while meeting the delay constraint of the baseline circuits. The optimization was done at hierarchy level 1 i.e. the circuit was partitioned into sub-circuits (using min-cut partitioning) and the sub-circuits were optimized flatly. The V_{th} values used were 0.2V and 0.3V. The results of the optimization are reported in Table 6.1. The fifth, sixth and seventh columns give the energy decrease, delay increase and unreliability decrease after DAV optimization (assuming perfect level-shifters with zero delay and energy overhead). The eight, ninth and tenth columns give the corresponding numbers after quantizing the sizes to a resolution of 0.2 and clustering the V_{DD}s into a maximum of 3 different values (to take into account a finite sized library and limited numbers of V_{DD}s allowed in practice). After

Table 6.1. Optimization Results

Circuit (#Gates, #Sub-circuits)	Initial Delay (ps)	Initial Energy (fJ)	Initial Unreliability (ns)	% Energy Increase (DAV)	% Delay Increase (DAV)	% Unreliability Decrease (DAV)
c432 (267,4)	457	94.6	2.32	42.6	6.7	71.9
c499 (835,8)	351	351	4.39	12.2	5.9	53
c1908 (680,8)	523	262	5.71	42.9	6.2	95.2
c2670 (875,8)	326	333	8.77	46.4	6.3	83.5
c3540 (1319,8)	632	528	10.97	26.1	5.6	79.9
c5315 (1994,16)	510	784	23.8	26.9	4.9	88.9
c7552 (2538,16)	444	1088	22.1	30.9	7.2	85.5
Average	463.3	491.5	11.2	32.6	6.1	79.7

Table 6.1. (Continued)

Circuit (#Gates, #Sub-circuits)	% Energy Increase (Final)	% Delay Increase (Final)	% Unreliability Decrease (Final)
c432 (267,4)	42.5	6.8	71.3
c499 (835,8)	12.2	6.2	53.1
c1908 (680,8)	41.2	6.3	95.2
c2670 (875,8)	45	6.3	83.5
c3540 (1319,8)	23.2	5.5	78.6
c5315 (1994,16)	23.5	4.9	88.6
c7552 (2538,16)	28.6	7.2	84.6
Average	30.9	6.2	79.3

clustering, the high V_{DD} gates that were driven by low V_{DD} gates were given a higher threshold voltage to reduce their leakage power. As mentioned before, the delay

Table 6.2. Circuit Statistics

Circuit	Initial Circuit			Final Circuit			% Increase in Circuit Size	% High V_{th} gates	Sub-circuit V_{DD}s (V)
	Min Size	Mean Size	Max Size	Min Size	Mean Size	Max Size			
c432	1	2.35	6	1	1.54	13.6	11.6	49.1	1.2
c499	1	2.52	6	1	1.6	7.2	12.3	35.9	1.2
c1908	1	2.23	6	1	1.46	7	36.6	49.6	1.2
c2670	1	2.6	6	1	1.48	20	43.5	49.3	1.2
c3540	1	2.25	6	1	1.35	12.4	-9.6	67.6	1.2
c5315	1	2.27	7	1	2.26	20	44.3	68.5	1.2
c7552	1	2.24	8	1	1.38	20	4.9	57.8	1.2
Average	1	2.35	6.43	1	1.58	14.3	20.5	54	-

constraint is exceeded due to the finite sized library used. Table 6.2 gives some circuit statistics before and after optimization. It is seen that although the maximum gate size goes up, the average gate size goes down. The circuit area was estimated as the sum of load capacitances of all gates. Although the average gate size went down, the circuit size still went up because of the added load capacitances at the primary outputs. The optimization results shown are actually for the case when only a V_{DD} of 1.2V was allowed for all the sub-circuits. If multiple V_{DD}s were allowed, it was found that the partition with PO gates got a much higher V_{DD} than the other partitions. This led to a lot of leakage energy dissipation that couldn't be reduced by just increasing the threshold voltages of the high V_{DD} gates driven by low V_{DD} gates.

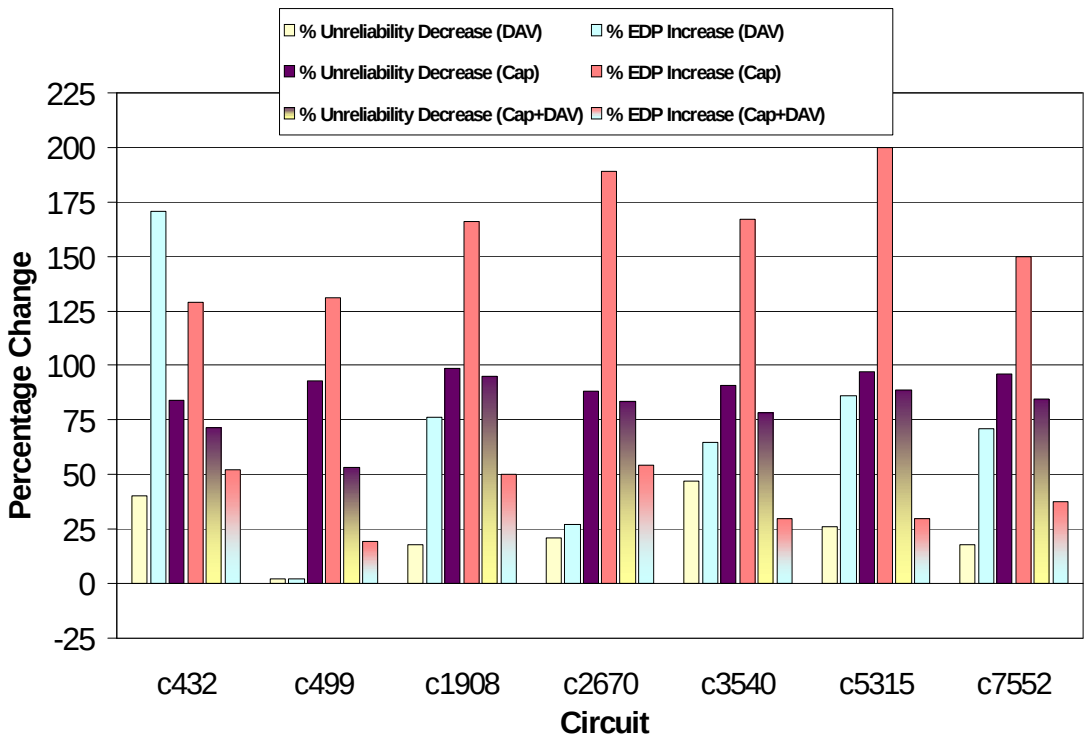


Figure 6.6. Comparison of methods (Cap+DAV), (DAV) and (Cap)

In [70], a similar method using DAV is presented but which does not use PO capacitances. The cost function is also a weighted sum of circuit unreliability, energy delay and area. Hence the circuit unreliability needs to be computed in every iteration of the search, with a lot of runtime overhead.In [71], PO capacitances are inserted and the delay overhead is tried to be reduced by simply increasing the supply voltages of the gates. Although this method leads to a large reduction in soft-errors, the energy overhead is very large. Figure 6.6 compares the unreliability decrease and EDP increase of the method in this chapter and the above two methods. The method in this chapter is referred as Cap+DAV, the method in [70] as DAV and the method in [71] as Cap. It is clearly seen that using capacitive loads at the outputs (for increasing reliability) while using DAV optimization for the rest of the gates (to reduce EDP) provides much better energy-delay-reliability trade-off than just using capacitive loads at output (and increasing V_{DD}

to offset delay penalty) or just using DAV. Just using DAV is not very effective in reducing unreliability. Cap is effective in highly reducing unreliability but the EDP cost is very high. Cap+DAV is able to reduce unreliability almost as much as Cap but with very low EDP overhead.

6.6 Conclusion

This chapter presented tools for the analysis and optimization of the soft-error tolerance of nanometer combinational circuits. The analysis tool, ASERTA, is able to accurately calculate (with average correlation of 0.9 with SPICE) the "unreliability" of circuits in orders of magnitude less computation time than SPICE. The optimization tool, SERTOPT, maximizes the average PO delay while simultaneously minimizing the energy-delay product (EDP) for the circuit. The bigger delays of the PO gates are matched by adding extra capacitances at the POs. This reduces both glitch generation and propagation through the PO gates. Experiments on ISCAS'85 benchmark circuits show that SERTOPT is able to reduce the unreliability of circuits by 79.3% on the average while incurring delay, energy and area penalties of 6.2%, 30.9% and 20.5% on the average respectively.

Chapter VII

Conclusions and Future Research

7.1. Summary of Research Contributions

The thesis developed an efficient, hierarchical and unified framework for the co-optimization of the energy consumption and soft-error robustness of digital circuits under specified timing constraints. The design variables that were used in the co-optimization were the supply and threshold voltages of the gates/modules in the netlist and/or the sizes of the transistors comprising the gates. The novelty in the co-optimization approach was the use of a technique called Delay-Assignment-Variation (DAV) based optimization. This technique allows delay constrained optimization to be formulated efficiently as a global optimization problem targeting any DAG based cost function. In this thesis, the cost functions studied were circuit energy and circuit unreliability. The next section describes some other problems that can also be tackled by this approach.

The following outlines the contributions of the research:

- A novel matrix representation, **T**, of a netlist was developed that can be used in various circuit optimizations which have delay as a constraint. Using the matrix representation, a generic and efficient Delay-Assignment-Variation (DAV) based optimization technique was formulated that can be applied to various delay-constrained optimization problems. The efficiency of DAV based optimization compared to brute-force parameter optimization results due to the implicit

consideration of the delay constraints that allows a big reduction in the optimization problem size.

- Using the DAV based approach, an exact technique has been developed that determines the values of the supply and threshold voltages to be used in a digital circuit such that the sum of the dynamic and static energy consumption of the circuit is minimized while meeting the timing constraint.
- A metric, the Normalized Energy Gradient (NEG) metric, was formulated that can be used to evaluate the "energy efficiency" of any design using a limited number of supply and threshold voltages as compared to the optimum design having unlimited number of available voltages. The NEG metric can also be used to rank various low power optimization techniques that make use of multiple supply and threshold voltages according to how close they bring the system to the lowest energy consumption possible.
- DAV based optimization was formulated for gate level netlists. To handle large netlists, a hierarchical application of the approach was developed. Depending on the size of the netlist, optimization could be done at different levels of hierarchy to trade-off computational complexity and optimality of the results. Hierarchical DAV based optimization was used to find optimal sizes and threshold voltages of gates and supply voltages of sub-circuits that minimize the energy consumption.
- Usage of a combination of SPICE look-up tables and analytical equations to model energy and delay of CMOS combinational circuits accurately and efficiently was described.

- The unreliability of individual gates in a CMOS circuit was characterized in a novel way using their “glitch generation” and “glitch propagation” characteristics. The effect of various gate parameters such as size, output load, and supply and threshold voltage on the unreliability of a gate was investigated.
- An accurate and efficient tool called ASERTA was developed for the analysis of soft-error tolerance of a CMOS combinational circuit. The tool estimates the soft-error tolerance with accuracy close to SPICE in orders of magnitude less computation time.
- Usage of gate sizing, multiple supply/threshold voltages and output loading for increasing the soft-error tolerance of digital circuits was investigated. The problem was formulated as a DAV based optimization problem and it was shown that a judicious combination of sizing, primary output loading and usage of multiple supply/threshold voltages can significantly increase the glitch attenuation characteristics of circuits with marginal delay-energy penalties. It was shown that this was a better approach for reducing unreliability than just using gate sizing or just using higher V_{DD}s combined with primary output loading.

7.2. Future Directions

The DAV based optimization approach given in this thesis is a general a approach that can be used to solve various delay-constrained DAG based optimization problems. A few examples are given below:

- This approach can be easily applied at the software level where the data flow graph (DFG) is a DAG. The nodes of the graph represent computations/instructions that execute on functional units (FUs). Under a dynamic voltage scaling (DVS) model, the

supply voltages of the FUs can change dynamically depending on the delay/energy requirements. Using DAV based optimization and appropriate models of FU delay and energy, the optimal voltages for the FUs that minimize the energy consumption for the entire flow graph (while meeting the timing constraints) can be found and stored in memory. A controller can then read the memory when the program is executing and correctly assign voltages to FUs dynamically to minimize energy.

- In this thesis, DAV based optimization was applied to the problem of minimizing energy consumption and increasing soft-error tolerance. The approach can be straightforwardly applied to the circuit sizing problem without any modifications. Instead of using energy and/or unreliability as the cost function, an estimate of the circuit size (computed using the gate sizes) can be solely used as the cost function to minimize area while meeting timing constraints.
- The DAV approach can be applied to solve the wire sizing and gate sizing problem simultaneously. The wires in a netlist can actually be represented as additional nodes between gates. This is shown in Figure 7.1. Each wire also has associated delays and energies similar to gates. Now, the DAV based optimization can be used to find optimal delay assignments for gates and wires. The optimal gate and wire sizes can be found from the optimal delay assignment by a backward traversal similar to the one described in Section 5.3. Simultaneous gate and wire sizing is likely to become a key optimization in future technologies when local wire delays also become significant.

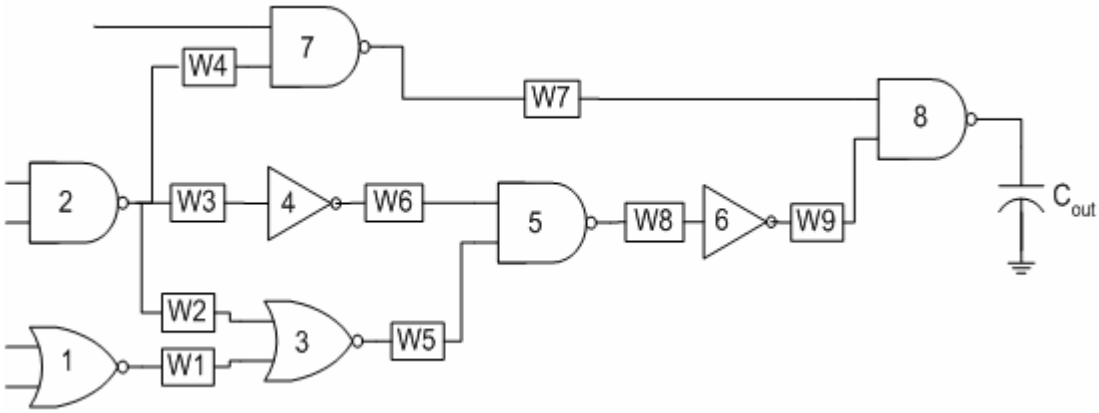


Figure 7.1. Representation for simultaneous gate and wire sizing

Improvements are also possible to the hierarchical application of DAV based optimization. A few are listed below:

- This work described and compared two different kinds of partitioning schemes viz. min-cut partitioning and topological partitioning. Min-cut partitioning has the advantage that is reduce the number of level-shifters required between partitions, thereby reducing the overhead of level-shifters that is inherent in any multi-V_{DD} optimization approach. However, min-cut partitioning imposes constraints on the sizes of the gates on the boundaries of the partitions (see Section 5.6), which might reduce the possible optimization benefits. Topological partitioning doesn't impose constraints on the sizes of the boundary gates but imposes conditions on the partition V_{DD}s (the V_{DD}s have to be increasing in value from POs to PIs) to eliminate the need for level-shifters. This might again reduce the optimization savings. The best way might be to have a partitioning scheme that reduces the number of edges cut between partitions while at the same time ensuring that the partitions can be sorted topologically (i.e. the partition graph is also a DAG).

- In this work, the partitions were generated without directly considering their impact on the energy savings (except trying to reduce the number of edges cut or trying to reduce the size constraints imposed). However, it might be possible to choose the sizes of partitions and the gates included in them in a more power-aware way. For example, in an ALU, the higher order bits have very low switching activity compared to the lower order bits. Hence, it might be a good strategy to cluster the logic corresponding to the higher order bits into one partition and the lower order bits into another. This will allow the inter-partition optimization stage to achieve higher energy savings by slowing down the higher activity partition while speeding up the lower activity partition.
- The mapping of gate delays to gate sizes, V_{DD}s and V_{th}s is a crucial step in DAV based optimization. At the module level (Chapter IV), the best V_{DD} and V_{th} could be chosen for a module for its delay assignment (using analytical equations), independent of the V_{DD}s and V_{th}s chosen for other modules. However, at the gate level, the choice of sizes, V_{DD}s and V_{th}s for a gate affects the choices for its predecessor gates. This makes optimal mapping of gate delays to sizes, V_{DD}s and V_{th}s very hard. This work explored two mapping schemes viz. the ELP scheme and the min-size scheme, but finding better schemes can be another research pursuit.

Appendix A

Topological Sort of Directed Acyclic Graphs

A topological sort of a DAG, G(V,E), is a numbering of the nodes of the graph in such a way that given any edge e(u→v) between nodes u and v, u is less than v. After a DAG has been topologically sorted, it is very easy to make forward and reverse traversals over it. A forward traversal can be done by visiting nodes in increasing order of the topological numbering i.e. from node 1 to node |V|. A backward traversal can be done by visiting nodes in decreasing order of the topological numbering i.e. from node |V| to node 1. A forward traversal is guaranteed to visit a node before visiting any of its successors whereas a backward traversal is guaranteed to visit a node before any of its predecessors. The algorithm for listing nodes in topological order is given in Figure A.1.

```
Algorithm Topsort(G(V,E))

        count = 1
        for each node n in V
                If n has no incoming edges, insert n into queue Q
        end
        while Q is nonempty
                Remove a node n from Q
                topnumber(n) = count
                count = count + 1
                for each node m with an edge e from n to m
                        Remove edge e from E
                        if m has no other incoming edges, insert m into Q
                end
        end
```

Figure A.1. Algorithm for obtaining topological numbering of nodes in a DAG.

Appendix B

Power Aware Zero Slack Algorithm

Given a DAG, G(V,E), with delays D and energies E associated with each node, five quantities can be computed for each node as follows:

(i) Early Start Time (EST):

If node n is a primary input (no incoming edges), EST(n) = 0.

Otherwise, EST(n) = max(EST(m)+D(m)) for all nodes m with edges to n.

(ii) Early Finish Time (EFT): For any node n, EFT(n) = EST(n)+D(n).

(iii) Late Finish Time (LFT):

If node n is a primary output (no outgoing edges), LFT(n) = max(EFT(m)) for all nodes m in V.

Otherwise, LFT(n) = min(LFT(m)-D(m)) for all nodes m with edges from n.

(iv) Late Start Time (LST): For any node n, LST(n) = LFT(n)-D(n).

(v) Slack: For any node n, Slack(n) = LST(n)-EST(n) = LFT(n)-EFT(n).

Computation of Early Start Times requires a forward traversal over G while computation of Late Finish Times requires a backward traversal over G. The procedure for computing the above five quantities for a DAG is given in Figure B.1. It uses the topological numbering of the nodes.

```
Algorithm Compute Slacks(G(V,E))
        for i = 1 to |V|
                Let n be the node with topological number i
                if n has no incoming edges, EST(n) = 0
                else EST(n) = max(EST(m)+D(m)) for all nodes m with edges to n
                EFT(n) = EST(n)+D(n)
        end
        Delay = max(EFT(n)) for all nodes n
        for i = |V| down to 1
                Let n be the node with topological number i
                if n has no outgoing edges, LFT(n) = Delay
                else LFT(n) = min(LFT(m)-D(m)) for all nodes m with edges from n
                LST(n) = LFT(n)-D(n)
                Slack(n) = LST(n)-EST(n)
        end
```

Figure B.1. Algorithm for computing slacks of nodes in a DAG.

After the slacks have been computed, the algorithm in Figure B.2 increases the delays of the nodes with non-zero slack until all nodes have zero slack. The nodes are chosen in the order of decreasing energy-slack product.

```
Algorithm Power Aware Zero Slack(G(V,E))
        Compute Slacks(G(V,E))
        while there is a node with non-zero slack
                Select node n that has maximum Energy-Slack product
                D(n) = D(n)+Slack(n)
                Compute Slacks(G(V,E))
        end
```

Figure B.2. Algorithm for removing slacks of nodes in a DAG in a power-aware manner.

References

[1] http://public.itrs.net/Files/2003ITRS/Home2003.htm, The International Technology Roadmap for Semiconductors, 2003.

[2] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, Vol. 19, Issue 4, pp. 23-29, August 1999.

[3] M. Horowitz, T. Indermaur, R. Gonzalez, "Low-power Digital Design," *ISLPED*, pp. 8-11, October 1994.

[4] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," *DSN*, pp. 389-398, June 2002.

[5] J.M. Rabaey, M. Pedram, "Low Power Design Methodologies," Kluwer Academic Publishers, 1996.

[6] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, R.W. Brodersen, "Optimizing power using transformations," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 12-31, Vol. 14, Issue 1, Jan 1995.

[7] M. Johnson, K. Roy, "Optimal Selection of Supply Voltages and Level Conversions during Data Path Scheduling under Resource Constraints," *ICCD*, pp.72 -77, 1996.

[8] A. Manzak, C. Chakrabarti, "A low power scheduling scheme with resources operating at multiple voltages," *IEEE Trans. on VLSI*, pp. 6-14, Vol. 10, Issue 1, Feb. 2002.

[9] A.U. Diril, Y.S. Dhillon, K. Choi, A. Chatterjee, "An O(N) Supply Voltage Assignment Algorithm for Low-Energy Serially Connected CMOS Modules and a Heuristic Extension to Acyclic Data Flow Graphs," *ISVLSI*, pp.173-179, February 2003.

[10] T. Ishihara, H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," *ISLPED*, pp. 197-202, August 1998.

[11] T. Pering, T. Burd, R. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scheduling Algorithms," *ISLPED*, pp.76-81, 1998.

[12] K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Kuroda, "A 300 MIPS/W RISC core processor with

variable supply-voltage scheme in variable threshold-voltage CMOS**,"** *CICC*, pp. 587-590, May 1997.

[13] K. Nose, T. Sakurai, "Analysis and future trend of short-circuit power," *IEEE Trans. on CAD*, vol.19, no.9, pp.1023-1030, Sept. 2000.

[14] T. Sakurai, A.R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, pp.584-594, April 1990.

[15] V. Sundararajan, K.K. Parhi, "Synthesis of low power CMOS VLSI circuits using dual supply voltages," *DAC*, pp. 72 -75, 1999.

[16] C. Chen, A. Srivastava, M. Sarrafzadeh, "On gate level power optimization using dual-supply voltages," *IEEE Transactions on VLSI Systems*, vol. 9, pp. 616-29, 2001.

[17] K. Usami, M. Horowitz, "Clustered voltage scaling technique for low-power design," *Low Power Design Symposium*, pp. 3-8, April 1995.

[18] M. Donno, L. Macchiarulo, A. Macii, E. Macii, M. Poncino, "Enhanced clustered voltage scaling for low power," *GLSVLSI*, pp. 18-23, April 2002.

[19] J. Chang, M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *IEEE Trans. on VLSI Systems*, vol.5, no.4, December 1997.

[20] C. Chen, M. Sarrafzadeh, "Power reduction by simultaneous voltage scaling and gate sizing," *ASP-DAC*, pp. 333-338, Jan 2000.

[21] L. Wei, Z. Chen, K. Roy, M.C. Johnson, Y. Ye, V.K. De, "Design and optimization of dual-threshold circuits for low-voltage low-power applications**,"** *IEEE Trans. on VLSI Systems*, pp.16-24, vol.7, no.1, March 1999.

[22] V. Sundararajan, K.K. Parhi, "Low power synthesis of dual threshold voltage CMOS VLSI circuits," *ISLPED*, pp.139-144, 1999.

[23] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, J. Yamada, "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 30, Issue 8, pp. 847-854, August 1995.

[24] A. Srivastava, D. Sylvester, "Minimizing total power by simultaneous V_{DD}/V_{th} assignment," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, Issue 5, pp. 665-677, May 2004.

[25] Y.S. Dhillon, A.U. Diril, A. Chatterjee, H.H.S. Lee, "Algorithm for achieving minimum energy consumption in CMOS circuits using multiple supply and threshold voltages at the module level," *ICCAD*, pp. 693-700, Nov 2003.

[26] R. Bai, S. Kulkarni, W. Kwong, A. Srivastava, D. Sylvester, D.Blaauw, "An implementation of a 32-bit ARM processor using dual power supplies and dual threshold voltages**,"** *ISVLSI***,** pp.149-154, 2003.

[27] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhary, R. Panda, D. Blaauw, "Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing," *DAC*, pp. 436-441, 1999.

[28] V. Sundararajan, K.K. Parhi, "Low power gate resizing of combinational circuits by buffer redistribution," Proc. 20th Anniversary Conference on Advanced Research in VLSI, pp. 170 – 184, 1999.

[29] D.S. Chen, M. Sarrafzadeh, "An Exact Algorithm for Low Power Library-Specific Gate Re-Sizing," *DAC*, pp. 783–788, 1996.

[30] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "A gate resizing technique for high reduction in power consumption," *ISLPED*, pp. 281 – 286, 1997.

[31] P. Pant, V.K. De, A. Chatterjee, "Simultaneous power supply, threshold voltage, and transistor size optimization for low-power operation of CMOS circuits," *IEEE Trans. on VLSI Systems*," Vol. 6, Issue. 4, pp. 538-545, December 1998.

[32] A. Srivastava, D. Sylvester, D. Blaauw, "Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment," *DAC*, pp. 773-778, June 2004.

[33] Y. Zhang, X. Hu, D.Z. Chen, "Cell selection from technology libraries for minimizing power," *ASP-DAC*, pp. 609-614, Feb 2001.

[34] M. Hamada, Y. Ootaguro, T. Kuroda, "Utilizing surplus timing for power reduction," *CCIC*, pp. 89-92, May 2001.

[35] T.K. Callaway, E.E. Swartzlander Jr., "Estimating the power consumption of CMOS adders,**"** *Proc. of 11th Symposium on Computer Arithmetic*, pp. 210-216, 1993.

[36] S. Iman, M. Pedram, "An approach for multilevel logic optimization targeting low power," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, Issue 8, pp. 889-901, Aug. 1996.

[37] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Transactions on VLSI Systems*, Vol. 2, Issue 4, pp. 426-436, Dec 1994.

[38] G.E. Tellez, A. Farrahi, M. Sarrafzadeh, "Activity-driven clock design for low power circuits," *ICCAD*, pp. 62–65, Nov. 1995.

[39] T. Chi-Ying, M. Pedram, A.M. Despain, “Power efficient technology decomposition and mapping under an extended power consumption model,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, Issue 9, pp. 1110–1122, Sept. 1994.

[40] K. Roy, S.C. Prasad, “Circuit activity based logic synthesis for low power reliable operations,” *IEEE Transactions on VLSI Systems*, Vol. 1, Issue 4, pp. 503-513, Dec.1993.

[41] T. Chi-Ying, M. Pedram, A.M. Despain, “Low-power state assignment targeting two- and multilevel logic implementations,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, Issue 12, pp. 1281-1291, Dec. 1998.

[42] A. Chandrakasan, R. Brodersen, “CMOS low power digital design," Kluwer Academic Publishers, 1995.

[43] S. Iman, M. Pedram, “Logic synthesis for low power VLSI designs,” Kluwer Academic Publishers, 1998.

[44] S. Devadas, S. Malik, “A survey of optimization techniques targeting low power VLSI circuits,” *DAC*, pp. 242-247, 1995.

[45] M. Pedram, “CAD for low power: status and promising directions,” *Proc. of International Symposium on VLSI Technology, Systems, and Applications*, pp. 331-336, 1995.

[46] S. Buchner, M. Baze, D. Brown, D. McMorrow, J. Melinger, “Comparison of error rates in combinational and sequential logic,” *IEEE Transactions on Nuclear Science*, Vol. 44, Issue 6, pp. 2209-2216, Dec. 1997.

[47] M. Nicolaidis and Y. Zorian, "On-line testing for VLSI - a compendium of approaches," *JETTA*, vol. 12, pp. 7-20, 1998.

[48] N.A. Touba, E.J. McCluskey, “Logic synthesis of multilevel circuits with concurrent error detection,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, Issue 7, pp. 783-789, July 1997.

[49] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," *VTS*, pp. 86-94, 1999.

[50] K. Mohanram, N.A. Touba, “Cost-effective approach for reducing soft error failure rate in logic circuits,” *ITC*, Vol. 1, pp. 893-901, 2003.

[51] Y.S. Dhillon, A.U. Diril, A. Chatterjee, A.D. Singh, “Sizing CMOS Circuits for Increased Transient Error Tolerance,” *IOLTS*, pp. 11-16, July 2004.

[52] Q. Zhou, K. Mohanram, “Transistor Sizing for Radiation Hardening,” *Proc. International Reliability Physics Symposium*, pp. 310-315, 2004.

[53] N. Kaul, B.L. Bhuva, S.E. Kerns, “Simulation of SEU transients in CMOS ICs,” *IEEE Transactions on Nuclear Science*, Vol. 38, Issue 6, pp. 1514-1520, Dec. 1991.

[54] M.P. Baze, S. Buchner, W.G. Bartholet, T.A. Dao, “An SEU analysis approach for error propagation in digital VLSI CMOS ASICs,” *IEEE Transactions on Nuclear Science*, Vol. 42, Issue. 6, pp. 1863-1869, Dec. 1995.

[55] R. Nair, C.L. Berman, P.S. Hauge, E.J. Yoffa, “Generation of performance constraints for layout,” *IEEE Trans. on CAD*, vol.8, no.8, pp.860-874, Aug. 1989.

[56] J.A. Butts, G.S. Sohi, “A static power model for architects,” *IEEE/ACM MICRO*, pp. 191-201, 2000.

[57] R. Kumar, C.P. Ravikumar, “Leakage power estimation for deep submicron circuits in an ASIC design environment,” *DAC*, pp.45-50, 2002.

[58] K. Nose, T. Sakurai, “Optimization of V_{DD} and V_{TH} for low-power and high-speed applications,” *ASP-DAC*, pp.469-474, 2000.

[59] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," *CICC*, pp. 201 – 204, 2000.

[60] S.C. Wong, G.Y. Lee, D.J. Ma, “Modeling of interconnect capacitance, delay, and crosstalk in VLSI,”, *IEEE Trans. on Semiconductor Manufacturing*, vol. 13, no. 1, pp. 108 – 111, 2000.

[61] G. Karypis, V. Kumar, “Multilevel k-way hypergraph partitioning,” *DAC*, pp. 343 - 348, 1999.

[62] W. Huyer, A. Neumaier, “Global optimization by multilevel coordinate search,” *Journal of Global Optimization*, vol. 14, no. 4, pp. 331 – 355, 1999.

[63] A.U. Diril, Y.S. Dhillon, A. Chatterjee, A.D. Singh, "Level-shifter free design of low power dual supply voltage CMOS circuits using dual threshold voltages," *ICVLSI*, 2005.

[64] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Trans. on Nuclear Science*, vol. 50, pp. 583-602, 2003.

[65] M. Oman, G. Papasso, D. Rossi, C. Metra, "A model for transient fault propagation in combinatorial logic," *International On-Line Testing Symposium*, pp. 111-115, July 2003.

[66] C. Zhao, X. Bai, S. Dey, "A scalable soft spot analysis methodology for compound noise effects in nano-meter circuits," *DAC*, pp. 894-899, 2004.

[67] L. B. Freeman, "Critical charge calculations for a bipolar SRAM array," *IBM J. Res. Develop*., vol. 40, pp. 119-129, 1996.

[68] P. Hazucha, C. Svensson, S.A. Wender, "Cosmic-Ray Soft Error Rate Characterization of a Standard 0.6-μm CMOS Process," IEEE Journal of Solid-State Circuits, vol. 35, no. 10, pp. 1422 – 1429, 2000.

[69] F. N. Najm and I. N. Hajj, "The complexity of fault detection in MOS VLSI circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, pp. 995-1001, 1990.

[70] Y. S. Dhillon, A. U. Diril, A. Chatterjee, "Soft-error tolerance analysis and optimization of nanometer circuits," *DATE*, pp. 288-293, 2005.

[71] A.U. Diril, Y.S. Dhillon, A. Chatterjee, A.D. Singh, "Design of adaptive nanometer digital systems for effective control of soft error tolerance," *VTS*, to be published, 2005.