

Copyright © 1997, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**ESTIMATION AND SYNTHESIS FOR LOW POWER,  
HIGH PERFORMANCE INTEGRATED CIRCUITS**

by

Premal Buch

Memorandum No. UCB/ERL M97/74

15 October 1997

COVER PAGE

**ESTIMATION AND SYNTHESIS FOR LOW POWER,  
HIGH PERFORMANCE INTEGRATED CIRCUITS**

Copyright © 1977

by

Premal Buch

Memorandum No. UCB/ERL M97/74

15 October 1997

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering  
University of California, Berkeley  
94720

## Abstract

### Estimation and Synthesis for Low Power, High Performance Integrated Circuits

by

Premal Buch

Doctor of Philosophy in Engineering - Electrical Engineering & Computer Sciences

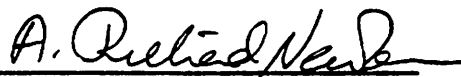
University of California, Berkeley

Professor A. Richard Newton, Chair

Power minimization is becoming very important for a number of reasons ranging from an increasing demand for portable computing to the problem of *hot chips* due to increasing clock frequencies and device counts of integrated circuits. Minimizing power dissipation of chips has an impact not only on energy savings, but also helps create more reliable chips. In this context, the advent of deep submicron technologies creates a moving target for CAD algorithms, which now need to reduce power in the existing design methodology as well as consider delay, power and area minimization from the deep submicron perspective. This thesis presents a set of algorithms for the characterization and synthesis of high performance integrated circuits with a focus on low power design.

Algorithms for fast vector-dependent power simulation and vector-independent power estimation at the transistor level are presented along with a fast mixed-signal simulator used to drive the estimation. A mixed-abstraction methodology is outlined for chip-level power estimation.

The logic synthesis problem is approached from two directions: optimizing a circuit for new design criteria like power dissipation in the current design methodology, and a new methodology for next generation circuit design targeting delay, power and area optimization in deep submicron technology. Statistical properties of functions and minterm probabilities in the Boolean space are analyzed and algorithms to reduce power dissipation without compromising the traditional design criteria like delay and area are presented at the technology independent and dependent level in the current static CMOS standard cell based framework. Pass transistor logic (PTL) is proposed as a promising alternative to static CMOS for deep submicron design and decomposed BDDs are proposed as a suitable logic level representation for synthesis of PTL networks. A comprehensive new synthesis flow based on decomposed BDDs is outlined for PTL design. It is shown that the proposed approach allows logic-level optimizations similar to the traditional multi-level network based synthesis flow for static CMOS, and also makes possible optimizations with a direct impact on delay, power and area of the final circuit implementation which do not have any equivalent in the traditional approach. Heuristic algorithms to synthesize PTL circuits optimized for delay, power and area in the new methodology are presented.



Professor A. Richard Newton  
Dissertation Committee Chair

# Table of Contents

<b>1. Introduction.....</b>	<b>1</b>
1.1. The New Design Challenges.....	1
1.2. The Traditional Design Methodology.....	3
1.3. The Low Power Design Problem.....	6
1.4. The Proposed Solution: Dissertation Overview.....	10
<b>2. Power Characterization.....</b>	<b>12</b>
2.1. Introduction.....	12
2.2. SYMPHONY: A Fast Mixed Signal Circuit Simulator.....	16
2.2.1. Introduction.....	16
2.2.2. Simulator Architecture.....	18
2.2.3. Dynamic Partitioning for Simulation of BiMOS Circuits.....	20
2.2.4. A PWL Model for Simulation of Digital Bipolar Circuits.....	22
2.2.5. Results.....	25
2.2.6. Conclusions.....	28
2.3. Power Simulation.....	28
2.3.1. Introduction.....	28
2.3.2. Power Simulation using SYMPHONY.....	30
2.3.3. Results.....	31
2.3.4. Conclusions.....	34
2.4. Power Estimation.....	34
2.4.1. Introduction.....	34
2.4.2. A Measure of Switching Activity for Analog Waveforms.....	35
2.4.3. A Monte Carlo Approach to Transistor Level Power Estimation.....	39
2.4.4. Partitioning for Estimation.....	42
2.4.5. Information Propagation Between Partitions.....	43
2.4.6. Results.....	44
2.4.7. Conclusions.....	50
2.5. Summary.....	51
<b>3. Logic Synthesis for Low Power Design.....</b>	<b>53</b>
3.1. Introduction.....	53
3.2. Preliminaries.....	57
3.3. Technology Independent Logic Synthesis for Low Power.....	62
3.3.1. Introduction.....	62
3.3.2. Previous Work.....	63
3.3.3. Partitioning the Boolean Space.....	63
3.3.4. Power Sensitive Minterms.....	68
3.3.5. Guided Logic Synthesis for Low Power.....	70
3.3.6. Results.....	74
3.3.7. Conclusions.....	76
3.4. Technology Dependent Logic Synthesis for Low Power.....	78
3.4.1. Introduction.....	78

3.4.2. Previous Work.....	80
3.4.3. Algorithm Overview .....	81
3.4.4. Sensitivity and Flexibility in Synthesis for Low-Power .....	82
3.4.5. Power Optimization Through Engineering Change.....	86
3.4.6. Results.....	91
3.4.7. Conclusions .....	92
3.5. Summary .....	94
<b>4. Logic Synthesis for Pass Transistor Circuits.....</b>	<b>96</b>
4.1. Introduction.....	96
4.2. Pass Transistor Logic Networks and BDDs.....	99
4.3. PTL Networks and Decomposed BDDs .....	103
4.4. A Synthesis Flow for PTL Design.....	106
4.5. Decomposition Techniques for BDD-based PTL Networks .....	108
4.5.1. Area Minimization .....	108
4.5.2. Performance .....	109
4.5.3. Low Power .....	112
4.6. Results.....	114
4.7. Enhancing the Decomposed BDD-based Approach.....	121
4.7.1. Don't Care Optimization.....	121
4.7.2. Synthesis of Mixed static CMOS/PTL Circuits .....	121
4.8. Conclusions.....	122
4.9. Summary .....	123
<b>5. Conclusions.....</b>	<b>124</b>
5.1. Future Work.....	127
<b>Bibliography .....</b>	<b>133</b>

# List of Figures

<b>CHAPTER 1.</b> .....	<b>1</b>
Figure 1-1 : The deep submicron trend [Tau95].....	2
Figure 1-2 : The traditional design flow .....	4
Figure 1-3 : Power dissipation in CMOS circuits .....	8
Figure 1-4 : Proposed Solution.....	11
<b>CHAPTER 2.</b> .....	<b>12</b>
Figure 2-1 : Event Scheduling.....	20
Figure 2-2 : BiCMOS adder: sum bit .....	27
Figure 2-3 : BiCMOS register file: writelines .....	27
Figure 2-4 : Power simulation in SPICE .....	29
Figure 2-5 : Power dissipation distribution of the 16-bit carry look ahead adder .....	47
Figure 2-6 : Predicted and actual error for the 16-bit carry look ahead adder .....	47
Figure 2-7 : Power dissipation distribution of the 32-bit counter.....	47
Figure 2-8 : Predicted and actual error for the 32-bit counter.....	47
Figure 2-9 : Runtime vs. err or trade-off as a function of number of partitions: 16-bit 2-to-1 multiplexor.....	49
Figure 2-10: Runtime vs. err or trade-off as a function of number of partitions: 8-bit wallace multiplier .....	49
<b>CHAPTER 3.</b> .....	<b>53</b>
Figure 3-1 : The traditional static CMOS synthesis flow .....	55
Figure 3-2 : Power dissipation at node vs. node signal probability.....	61
Figure 3-3 : Error vs. Number of Classes: $ I_p  = 10$ .....	66
Figure 3-4 : Actual Global Error vs. Product of Errors Estimate from (EQ 3.10) .....	67
Figure 3-5 : Proportion of Boolean Space vs. Proportion of Total Probability .....	69
Figure 3-6 : Power / Area Optimization Phases, $p(f_n) > 0.5$ .....	72
Figure 3-7 : Pseudo-code for Favoring Power Reduction during Area Optimization . .....	73
Figure 3-8 : Reducing switching activity via rewiring: an example. $F = \bar{a} \bar{c} + \bar{b} \bar{c} + \bar{c} \bar{d}$ , DC = $c\bar{d}$ , $p(a) = 0.4$ , $p(b) = 0.6$ , $p(c) = 0.2$ , $p(d) = 0.8$ . transition probability $f = p(\text{high})(1-p(\text{high}))$ , where $p(\text{high})$ for a variable is the probability of the variable being high in any clock period [She92] .....	78
Figure 3-9 : Function size probability profile.....	85
<b>CHAPTER 4.</b> .....	<b>96</b>
Figure 4-1 : Comparing pass transistor and static CMOS implementations of an example function $F = \bar{A} + B\bar{C}$ .....	97
Figure 4-2 : A PTL circuit with a sneak path .....	100
Figure 4-3 : Implementing a BDD node in PTL .....	100
Figure 4-4 : Comparing pass transistor implementations of the example function of Fig. 4-1 with its BDD .....	100
Figure 4-5 : Alternative BDD-based implementation of the example function from Fig. 4-1 .....	100

Figure 4-6 : Comparing monolithic and decomposed ROBDDs.....	105
Figure 4-7 : The traditional static CMOS synthesis flow vs. the proposed decomposed BDD synthesis flow .....	106
Figure 4-8 : BDD size reduction due to Boolean simplification via composition .....	108
Figure 4-9 : High performance heuristics (ordering: $A, B, C, D, x, y, z$ ).....	110
Figure 4-10: Reducing occurrences of high switching activity node.....	113
Figure 4-11: Low Power heuristic to minimize glitching: $F = A + BC, p(A=1) \cdot 1$ fi $p(F=1) \cdot 1$ .....	113
Figure 4-12: HSPICE results on timing and power dissipation of a full adder circuit implemented in static CMOS and PTL .....	120
<b>CHAPTER 5.</b> ....	124
Figure 5-1 : A Framework for synthesis in deep submicron technologies [Nex97]....	130



# List of Tables

<b>CHAPTER 1.</b>	1
<b>CHAPTER 2.</b>	12
Table 2-1: Results on BiMOS benchmark circuits	26
Table 2-2: Results on digital bipolar circuits (A “-” indicates that the program could not converge)	26
Table 2-3: Circuits for power simulation benchmarking	33
Table 2-4: Power simulation accuracy comparison: HSPICE, PowerMill, SYMPHONY	33
Table 2-5: Power simulation runtime comparison: HSPICE, PowerMill, SYMPHONY	33
Table 2-6: Circuits for power estimation benchmarking	45
Table 2-7: Power estimation accuracy and runtime comparison	46
Table 2-8: Comparison between direct simulation with worst case primary input switching, and the statistical approach	49
<b>CHAPTER 3.</b>	53
Table 3-1: Comparison between <i>script.rugged</i> and <i>script.power</i>	75
Table 3-2: Power reduction by rewiring on some area-optimized MCNC/ISCAS-89 benchmarks	93
<b>CHAPTER 4.</b>	96
Table 4-1: Comparing the best area static CMOS vs. PTL and the best delay static CMOS vs. PTL (Area is measured in l2 and delay is measured as the length of the critical (longest topological) path)	118
Table 4-2: Comparing static CMOS optimized using local minimizations and full Don’t Cares (using <i>script.rugged</i> for area and <i>script.delay</i> for delay) vs. PTL (which does not use local optimizations or any Don’t Cares in this implementation). (Area is measured in l2 and delay is measured as the length of the critical path). A “-” indicates that the program could not complete due to space out.	118
Table 4-3: Comparing static CMOS vs. PTL in runtime (measured in seconds) (Note that the output of static CMOS algorithms is a mapped logic network while the output of the PTL algorithm is an HSPICE netlist). A “-” indicates that the program could not complete due to space out.	119
Table 4-4: Comparing static CMOS of Table 4-1 vs. PTL for cell counts and average cell size (measured in l2)	119
Table 4-5: Experimental data for a full adder circuit	120
<b>CHAPTER 5.</b>	124

# Acknowledgments

I would like to thank my advisor Prof. A. Richard Newton for his constant support, guidance and encouragement during the course of this research. I would also like to thank Prof. Ernest Kuh for guiding me through the initial stages of my research and Prof. Ilan Adler for his feedback on the dissertation. Thanks also to Prof. Alberto Sangiovanni-Vincentelli and Prof. Robert Brayton for all the guidance and help they provided through my graduate studies, Prof. Jan Rabaey for his input as a member of my qualification examination committee, and Dr. Vijay Nagasamy for mentoring me on parts of this work. Working with Chris Lennard, James Cherry, and John Lillis, was a pleasure.

I would also like to thank all the members of my research group - Charles Hough, Henrik Esbensen, Hiroshi Murata, Janet Wong, Julie Hu, Narsimha Bhat, and the folks who shared my graduate school experience in 550 Cory Hall - Adrian Isles, Alok Agrawal, Amit Mehrotra, Edoardo Charbon, Gitanjali Swamy, Jagesh Sanghavi, Luca Carloni, Marco Sgroi, Mukul Prasad, Rajeev and Neelakshi Murgai, Rajeev Ranjan, Ravi Gunturi, Renu Mehra, Shaz Qadeer, Sriram Krishnan, Sunil Khatri, Wilsin Gosti, and Yuji Kukimoto. Thanks also to the wonderful administrative staff who help create a great work environment for all us in the CAD group - Brad Krebs, Carol Sitea, Corey Schaffer, Flora Oviedo, Judd Reiffin, Justin Adler, Kia Cooper, and Tahani Sticpewich.

Looking back at the years in graduate school, there are a few people who appear in some of the fondest memories. Amit Narayan and David Xue - having them to share the ups and downs in these years was both a pleasure and a privilege. Then there are people who could only share the ups and downs over a long distance phone line - my parents, Rekha and Vinayak Buch - I know how hard it was when five years back I left for Berkeley and I know how proud you have been of

every little milestones I achieved at every stage of my Ph.D. Thank you. And there is Tanvi, my fiancée - you made the last two years wonderful and I look forward to many more together.

This work was supported in part by the Semiconductor Research Corporation under contract DC-324-032, by the Digital Equipment Corporation, and by Motorola. Their support is gratefully acknowledged.

---

# 1 Introduction

---

## 1.1. The New Design Challenges

Over the last several decades, the semiconductor industry has enjoyed an exponential growth as predicted by Moore's law. During this growth, device sizes have continued to shrink, chip densities have been increasing by an order of magnitude every six years and clock frequency has been increasing by an order of magnitude every eight years [Cam97](Fig. 1-1). The 1994 Semiconductor Industry Association roadmap predicted that in 1998 the process technologies will be at  $0.25\mu$  and the clock speeds at 450 MHz; and that they will reach  $0.13\mu$  and 800 MHz respectively in 2004. The technology growth is already ahead of this predicted curve in 1997, with  $0.18\mu$  deep submicron processes used in several advanced designs and the 600 MHz DEC Alpha microprocessor already in mainstream production, causing the roadmap to be revised upwards.

Traditionally, circuit design has been driven by two main cost criterion: chip area, which has been used as a measure of the expected production yield, and speed, which has been the performance metric strongly influencing the market success of a product. In the deep submicron technologies, with on-chip frequencies approaching a gigahertz and the device counts increasing to several tens of millions of transistors, there are now several new challenges that designers must address. The power dissipation of DEC Alpha, the fastest available microprocessor today, is 72 watts, and more mainstream microprocessors like the Pentium compatible series from Intel, AMD, and Cyrix, are not far behind with power dissipations in the order of 40 watts. These numbers are expected to rise to 140 watts even at the conservative prediction levels of the 1994 roadmap. On-

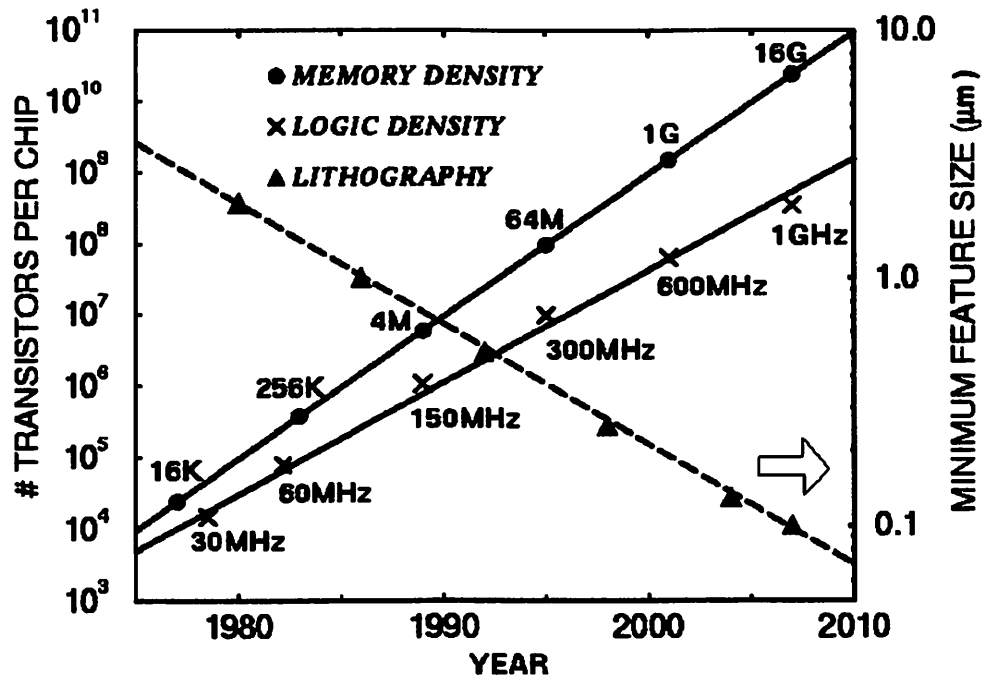


Figure 1-1 : The deep submicron trend [Tau95]

chip energy generation at this level can seriously affect the chip performance and reliability. Even at the current levels of power dissipation, these designs are long beyond the range of the cheap plastic, or even ceramic packaging. The expensive packaging and on-chip heat sinks required with these designs also drives up the production cost. This has created an urgent need for power minimization algorithms for high performance circuits.

Apart from high performance circuits, power minimization has also assumed great importance in the context of the increasing demand for portable computing and telecommunication equipment and the associated problem of battery life. With the existing battery technologies and the current power dissipations of even stripped-down versions of the state-of-the-art microprocessors, the battery life for a typical portable computing device is of the order of a few hours. The need to extend the battery life and thus the usability of these devices has been another big driver in the need for low power design techniques.

Along with the emergence of power as a major design concern, the advent of deep submicron process technologies also impacts the traditional design flow in another way: smaller geometries imply that the dominant delay moves from gates to wires and interconnect. Narrowing wire widths increases the resistances of the wires. This is partially compensated by using metals with higher conductivity (e.g. copper) and by increasing the height of the wires, but the global interconnect still scales quadratically with the scaling of technology. At  $1.0\mu$  technology in an automatically placed and routed layout, intrinsic gate delay contributed to 70% of the path delay. This has changed to a delay contribution of 10% in the  $0.18\mu$  technology, with the fanout load contributing another 25% and the remaining 65% of the delay due to the interconnect<sup>1</sup> [Keu97]. With the increasing importance of wires in the design, physical effects which were considered second-order before (e.g. cross-talk, noise and signal integrity) cannot be ignored either. This creates a need for algorithms which consider interconnect and physical design issues at all stages in the design process.

## **1.2. The Traditional Design Methodology**

The traditional static CMOS standard cell based design methodology has been aimed at minimizing the overall gate area and delays with no regard to power dissipation or wire planning. The design process is usually carried out in a top-down fashion with several distinct, relatively decoupled phases like high level synthesis, logic synthesis and physical design (Fig. 1-2).

High level synthesis generates a register transfer level structure which realizes the given behavioral description. Temporal scheduling, and allocation and binding of hardware are the issues considered at this stage.

---

1. For manually optimized layouts, gate delay remains the dominant component today. However, the trend is in the same direction.

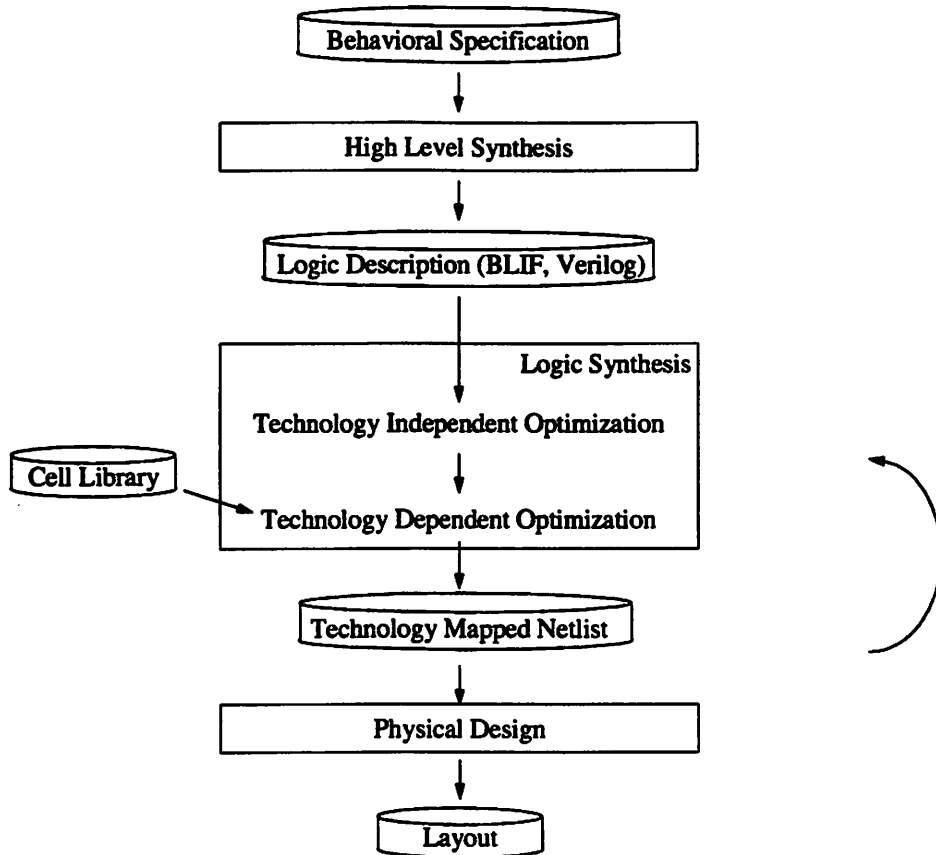


Figure 1-2: The traditional design flow

The input to the logic synthesis phase is the register transfer level description of the circuit, and a cell library. The circuit is typically represented as a multi-level logic network, which is then optimized for various design objectives like area and delay to generate a gate level netlist implemented with elements from the given cell library. The optimization phase itself consists of two sub-phases: technology independent and technology dependent optimization ([Bar87][Bra87]). The objective of the technology independent phase is to simplify the logic level netlist without making any assumptions about the underlying technology to be used for the actual implementation of the circuit. Each node of the multi-level logic network at this stage represents an arbitrarily complex function. The network is then optimized using Boolean and algebraic operations on nodes like node factoring, substitution, elimination, node simplification using don't cares, etc. ([Sav90b]). The

technology dependent phase takes this netlist as input and transforms it to implement and optimize it for a particular technology.

The mapped netlist is then input to the physical design tools which place and route the netlist to realize the physical layout of the circuit which is optimized for area and delay. This layout can then be shipped out for manufacturing.

In light of the design challenges described in the previous section, this methodology has several shortcomings. First, all phases of this flow currently target area and delay optimization. With power dissipation becoming a major concern in designs, there is a need for power characterization and optimization tools at all phases in this methodology. Another shortcoming of this methodology is that it is geared towards optimizing the gate delay of static CMOS standard cell circuits. In the deep submicron technologies, interconnect delay is the dominant factor in path delays. This is a major limitation of this methodology since the logic synthesis phase is entirely decoupled from the physical design phase, and has very little information on the wires in the final layout. Moreover, in the logic synthesis phase, the technology dependent phase can completely randomize and make irrelevant the optimizations performed in the technology independent phase. In order to meet the market demand for ever increasing clock speeds, it is necessary to look at alternatives to the static CMOS technology as well as develop algorithms which consider the wire delay when optimizing circuits. A solution to this problem would need a complete overhaul of the existing design methodology.

In the last few years, power optimization has been a major focus of research interest and synthesis in deep submicron technologies is now starting to get a great amount of attention. This dissertation addresses some of the above needs by looking at power characterization techniques at the transistor level of abstraction and power optimization algorithms for logic synthesis. It also presents a new logic synthesis flow for pass transistor logic (PTL), a promising alternative to static



CMOS logic, and a logic level representation for PTL circuits which tightly couples the logic level optimizations to the final circuit implementation. This will be a part of the initiative on a new synthesis framework for deep submicron technologies underway at the University of California, Berkeley [Nex97].

### 1.3. The Low Power Design Problem

Power dissipation is influenced by decisions made at every stage of the design process starting from the behavioral, to logic and transistor levels of abstraction. The design has to meet the power budget at every stage. In order to achieve this without time consuming iterations of implementation and evaluation, power estimation tools are needed at each level of design abstraction. Currently, the problem of designing low power systems is tackled on an *ad hoc* basic, with designers usually relying on experience and intuition. In order to design power efficient systems, formal techniques have to be developed to minimize power dissipation. The following analyzes the power dissipation in static CMOS circuits to put the power characterization and optimization problem in perspective.

In CMOS circuits, there are two components that contribute to power dissipation [Cha92a]: static dissipation (due to leakage, substrate injection and sub-threshold currents) and dynamic dissipation (due to switching transient current and charging and discharging of load capacitance). The total power dissipated by the circuit,  $P_{total}$ , is the sum of the two components:  $P_s$ , the static power dissipation, and  $P_d$ , the dynamic power dissipation.

$$P_{total} = P_s + P_d \quad (\text{EQ 1.1})$$

In most CMOS ASICs the contribution due to static dissipation is small compared to dynamic dissipation. The static power dissipation  $P_s$  of a circuit is given by the equation:

$$P_s = \sum_i^n I_l \times V_{dd} \quad (\text{EQ 1.2})$$

where  $I_l$  is the sum of the leakage, substrate injection and sub-threshold current of the CMOS gate,  $V_{dd}$  is the supply voltage and  $n$  is the number of devices in the circuits.

Ideally CMOS circuits dissipate no static power since in the steady state there is no direct path from  $V_{dd}$  to ground. However, in practice, leakage, substrate injection and sub-threshold currents give rise to a static component of CMOS power dissipation. For a submicron NMOS device with an effective  $W/L = 10/0.5$ , the substrate injection current is on the order of 1-100 $\mu$ A for a  $V_{dd}$  of 5V [Wat89]. Since the substrate current reaches its maximum for gate voltages near  $0.4V_{dd}$  and since gate voltages only reside in this range during the switching transients, the actual power dissipation due to substrate injection contribution is several orders of magnitude below other power dissipation factors. Similarly, reverse-bias junction leakage currents associated with the parasitic diodes in the MOS devices are on the order on nanoamps and have little effect on the overall power dissipation. Sub-threshold currents arise from the fact that even when  $V_{GS} = 0$ , MOS transistors are not perfect open switches and still carry a small current. At the current technologies this is a negligible factor, but is expected to become more important as the  $V_{dd}$  and  $V_T$  scale down [Gra94].

The dynamic component power dissipation is the sum of the short circuit power dissipation of a gate and the capacitive power dissipation, and is proportional to the switching activity at the output of a CMOS gate. The short circuit power dissipation is due to the fact that at some point during the switching transient, both the NMOS and PMOS networks are turned on. During this time, a short circuit exists between  $V_{dd}$  and ground. This component can be kept below 10-15% of the total power dissipation by designing the circuit such that the rise and fall times of the signals throughout the design are within a fixed range and as close to equal as possible [Vee84].

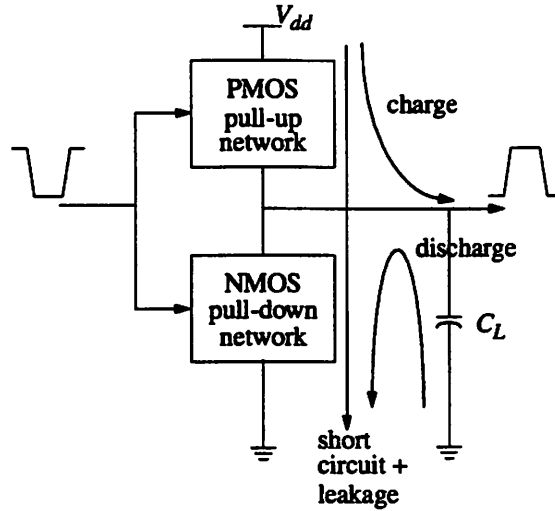


Figure 1-3 : Power dissipation in CMOS circuits

The dominant component of power dissipation (and also the over-all power dissipation) is the capacitive power dissipation, which results from the charging and discharging of parasitic capacitances in the circuit as circuit nodes change values. For the circuit in Fig. 1-3, where all the parasitic capacitances are lumped at the output in the equivalent load capacitor  $C_L$ , as the input switches from high to low, the NMOS pull-down network is cut off and the PMOS pull up network is activated, charging  $C_L$  up to  $V_{dd}$ . This charging process draws an energy equal to  $C_L V_{dd}^2$  from the power supply. Half of this is dissipated in the PMOS transistors while the other half is stored on the load capacitance. Then, when the input makes a transition from zero back to one, the process is reversed and the capacitance is discharged with its energy dissipated in the NMOS network. The capacitive power dissipation  $P_{d_i}$  for a gate then directly depends on the switching activity of the gate and is given by:

$$P_{d_i} = \frac{1}{2} \cdot \alpha_i \cdot C_{L_i} \cdot V_{dd}^2 \cdot f \quad (\text{EQ 1.3})$$

where  $C_{L_i}$  is the output load capacitance on the gate  $i$ ,  $V_{dd}$  is the supply voltage,  $f$  is the clock

frequency and  $\alpha_i$  is the expected number of number of switching transitions per clock cycle for gate  $i$ . This component of power dissipation typically contributes to almost 90% of the total power dissipation. The dynamic power dissipation  $P_d$  of a circuit with  $n$  gates is then given by the summation:

$$P_d = \frac{1}{2} \cdot V_{dd}^2 \cdot f \cdot \sum_i^n \alpha_i \cdot C_{L_i} \quad (\text{EQ 1.4})$$

From the power characterization perspective, the above analysis means that estimating the load capacitance and the switching activity are the two main challenges in performing accurate power estimation. Early in the design process, the capacitance can be estimated by treating circuit blocks as pre-characterized black boxes, and the switching activity can be estimated from high-level simulation data. However, while such estimates serve the purpose of guiding early design decisions, they are not very useful for accurate characterization of a circuit. A very accurate analysis of the power dissipation can only be performed at a stage in the design process when the transistor level description of the circuit with all the parasitic information is available. Such an accurate estimation methodology is one of the topics addressed in this dissertation.

From the power optimization perspective, the above analysis means that power savings can be achieved by reducing the power supply voltage, clock speed, circuit capacitances or switching activity. The supply voltage value and the system clock frequency are usually fixed in the beginning of the design process based on market and design process driven considerations (there are still some optimizations possible which reduce the supply voltage and local clock frequency for parts of the circuit not in the critical path. These are also typically performed at the behavioral level [Cha92a]). The gate and transistor level optimization techniques should then be targeted towards minimizing the circuit capacitances and switching activity. Circuit capacitances are typically a strong function

of the active gate area (although wiring capacitance may become a factor as technology scales). As a result, the minimum area implementation of a circuit often corresponds to the minimum power implementation as well, and power savings can be gained by applying area minimization algorithms to a circuit. In general however, this is not true because the power dissipation is a strong function of the switching activity, and there is a need for algorithms specifically directed at reducing switching activity as well. Such optimization techniques at the logic level of abstraction is another topic addressed by this dissertation.

#### **1.4. The Proposed Solution: Dissertation Overview**

This dissertation addresses the three problems described above: power characterization, power optimization, and synthesis for next generation technologies. The solutions presented here can be used as point tools for estimation and synthesis, or as a part of a larger framework for the estimation and synthesis for next generation designs (Fig. 1-4).

Chapter 2 addresses the power characterization problem. A methodology for vector-dependent and vector-independent transistor level power estimation is described, which consists of faster core circuit simulation algorithms, their application to power simulation, and a statistical model for the average power estimation problem.

Chapter 3 addresses the problem of power optimization. Specifically, logic level power optimization techniques are described at the technology independent and dependent level. These techniques exploit the statistical properties of the minterm probability distributions in the Boolean space to minimize the power dissipation of a circuit without trading off area or delay.

Chapter 4 addresses the problem of logic synthesis for pass transistor networks. A new flow for pass transistor logic synthesis is outlined. This flow allows logic-level optimizations similar to the traditional multi-level network based synthesis flow for static CMOS, and also makes possible

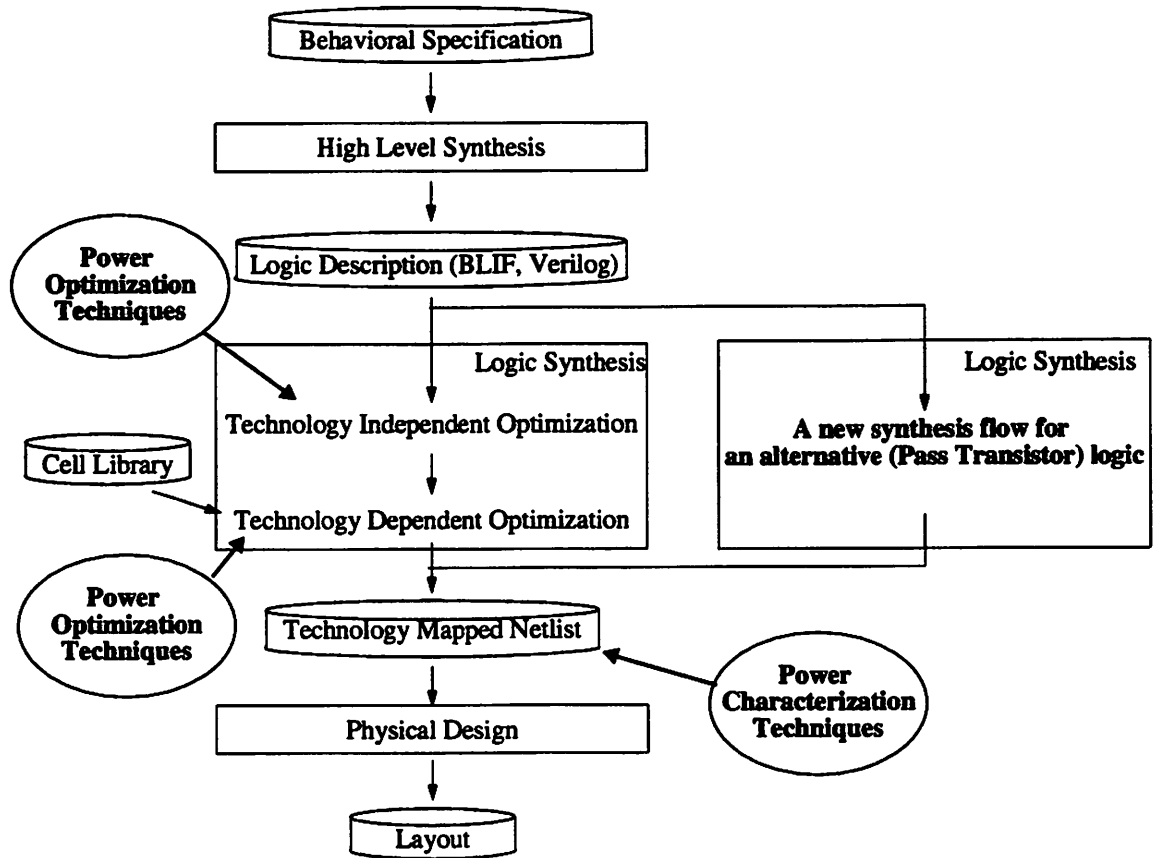


Figure 1-4 : Proposed Solution

optimizations with a direct impact on area, delay and power of the final circuit implementation. Decomposed BDDs are proposed as a suitable logic level representation for such a flow. Finally, in Chapter 5, a summary of the dissertation is presented and some of the possible directions of future research on the topics presented here are outlined.

---

# 2 Power Characterization

---

## 2.1. Introduction

Power characterization is essential to systematically guiding the design process to meet its power goals as the design moves through the register transfer, logic and transistor levels of abstraction. In the initial stages of the design, power characterization is required to obtain feedback about design decisions, while in the later stages power characterization can be used to help identify potential hot spots in the design before it is fabricated. A challenging problem in this context is how to efficiently obtain power estimates which meet the accuracy and the run-time constraints of the designer. In this work, we focus on the problem of power characterization at the transistor level of design abstraction.

The power characterization problem encompasses the problems of estimating the average power dissipation of a circuit, estimating the worst-case sustainable power dissipation and computing instantaneous power dissipation. The worst-case power dissipation is of interest from the electromigration and chip reliability perspective, while average power dissipation is useful from the battery life and packaging selection perspective. In this work, we focus on the problem of estimating the average power dissipation of the circuit. The proposed estimation technique uses a new, fast power simulator at its core which can be directly applied to compute the instantaneous power dissipation of the circuit as well.

As presented in Section 1.3, CMOS and BiCMOS circuits draw a very small amount of

power supply current in their steady state. The bulk of the power supply current is drawn when a circuit node changes its value. Consequently, the power estimation problem is strongly vector dependent. All power characterization techniques targeting this problem can be classified under three broad classes: vector-independent techniques (e.g. statistical/empirical techniques [Lan93]), weakly vector-dependent techniques (e.g. probabilistic techniques [Naj94a][Tsu93b]), and vector-dependent techniques (e.g. simulation based techniques [Cir87][Den94][Kan86]). Each of these class of methods offers a different runtime-error trade-offs, with increasing runtimes and decreasing errors in the order above.

Empirical techniques generally make use of statistical measures of the circuit. They read a description of the design, compile various statistical measures (e.g. chip area, transistor count, feature size, number of pins, design functionality etc.), and calculate the power consumption based on these measures. Although these methods can be very fast, since very little of the implementation details are accounted for, the errors can be very high. The main use of these methods is to obtain rough estimates of power dissipation at early stages of the design.

The main advantages of the probabilistic techniques are their small runtimes and weak vector dependence. The probabilistic techniques use a stochastic model of logic signals of a circuit and propagate the probabilities of logic values through the combinational logic modules in order to compute the average switching rate of the circuit. This measure is in turn, used to obtain the average power consumption of the circuit. It can potentially be accurate; however, for high accuracy, the spatial and temporal correlation between internal node values must be modeled. As this proves to be expensive, most approaches trade off accuracy for speed, resulting in highly inaccurate estimates at times. Methods which attempt to model these effects suffer from blow-ups in time and memory requirements. This class of methods is weakly vector dependent since these methods use input probabilities only and do not require input vectors.



Simulation based methods offer the most direct approach of obtaining the power dissipation of a circuit. These methods consist of simulating the circuit under user-provided input vectors and monitoring its power dissipation. While offering good accuracy, this approach suffers from the drawback of large runtimes. Furthermore, the strong pattern-dependence is a severely limiting constraint since it is often impossible to obtain a large enough input pattern set such that the power estimate is statistically meaningful. Even in cases where such inputs are available, it is not easy to determine the size of the vector set required to obtain a meaningful estimate of the power dissipation in a typical operating environment. A digital design with  $n$  inputs can have  $2^n$  possible combinations at the primary inputs. For an analog design, or when considering temporal correlations between input values in different cycles in the digital case, the size of the possible input vector set would increase exponentially.

Characterizing a transistor-level design for power requires very accurate estimation tools which can model all physical effects in the device models (particularly so for deep submicron technology) and identify main sinks of power in the design. Thus, while empirical and probabilistic techniques are useful in obtaining rough estimates of power dissipation at early stages of the design, due to above disadvantages they are not suitable for very accurate estimation. On the other hand, industry standard circuit simulators such as SPICE3e [Nag75] and HSPICE [HSP92], while offering good accuracy, suffer from a drawback of limited capacity and large run times. Moreover, it is clearly not feasible to perform exhaustive simulation of all possible input combinations to obtain a power estimate of the design. Thus a technique is needed for fast estimation via simulation, but with some approach to limit the required number of input patterns.

In this work a solution is proposed to this problem of transistor level power estimation which consists of a new, fast circuit simulator SYMPHONY [Buc97a], a power simulator using the SYMPHONY as its core engine [Buc95b], and a statistical power estimation technique which uses

this power simulator to yield power estimates with a given confidence and error margin [Buc96a]. The resulting approach combines the accuracy of the simulation-based approaches with the weak vector dependence of probabilistic approaches.

**SYMPHONY** is a new mixed signal circuit simulator that alleviates some of the problems faced by tools such as HSPICE, thus making it realistically feasible to perform power estimation via simulation of the entire design. **SYMPHONY** combines the stepwise equivalence conductance approach [Lin93b] for digital MOS devices, a new piecewise-linear model for digital bipolar devices, and a SPICE-like engine for analog subcircuits for fast and accurate circuit simulation. **SYMPHONY** can be 2X-250X faster than SPICE3e depending upon the amount of analog circuitry present in the design.

From the power simulation standpoint, the advantage of this approach is that the power can be measured directly, without any additional overhead, by monitoring the conductance and the voltage waveform during each time-step. Using the stepwise equivalent conductance for each non-linear device and the piecewise linearity of the voltage waveforms, computing the power dissipation in a device in each time-step is reduced to a constant-time evaluation.

For power estimation, a stochastic model is developed for power dissipation at the transistor level. The concept of transitional density [Naj93] (used for gate level power estimation in [Bur93]) is extended to analog waveforms. It is shown that a similar approach can be adopted for power estimation at the transistor level. Specifically, it is proved that for each device voltage variable, a companion stochastic process can be constructed which converges to the device power dissipation everywhere and is strict-sense stationary and mean-ergodic. Power estimation is then reduced to a mean estimation problem and a Monte Carlo approach is applied to solve it. This consists of applying random inputs to the system and monitoring its output. A formal stopping

criterion is derived to guarantee a desired error bound at a specified confidence level, under the assumption that power dissipation in a clock cycle is normally distributed.

To further speed-up the estimation process, a divide-and-conquer approach is used to break down the problem in sizes that become practically feasible for transistor-level simulation. Each sub-problem can still be further partitioned to gain speed-ups in the circuit simulation process itself, as in [Kle82]. The main advantage of partitioning for estimation is that the multi-rate behavior and stiffness of a circuit from the power perspective can be exploited. Partitioning the circuit allows using different input vector sets of appropriate size for different parts of the circuit. This not only speeds-up the estimation process by reducing computation, but also improves the accuracy of the stopping criteria of the Monte Carlo estimation in case of circuits where a single power dissipation model does not fit well. A statistical model is used to propagate signal information between partitions of a circuit. This model can also be used to bias the input vector generation if any information about external inputs is provided by the user.

The rest of the chapter is organized as follows: SYMPHONY is introduced in Section 2.2 and its application to the problem of power simulation described in Section 2.3. A statistical technique for transistor-level power estimation is presented in Section 2.4 and summary of the work presented in this chapter on power characterization in Section 2.5.

## **2.2. SYMPHONY: A Fast Mixed Signal Circuit Simulator**

### **2.2.1. Introduction**

The growing use of mixed signal circuit design has given rise to a new set of challenges to CAD tools. To characterize and verify the functionality of the design at the transistor and layout level, circuit simulation is still one of the most important tools. While it is always possible to simulate the entire transistor level circuit with time trusted tools like SPICE3e [Nag75] and HSPICE [HSP92],

for large state-of-the-art circuits this has long ceased to be a feasible option. The common approach to mixed signal simulation is to use fast tools (which trade-off accuracy for speed) to simulate digital parts of the circuit, more accurate simulators for the analog parts and hope that there are no problems when both are put together in the same design. There are some simulation frameworks that interface fast, event-driven digital simulators with time point-driven analog simulators [Acu90][Cha92c][Vis91] but they can have problems handling strong feedback from analog to digital subcircuits (and vice versa).

With the push towards faster and faster designs, the use of bipolar devices in digital designs is increasing again. Most of the research in recent years has been concentrated on developing fast circuit simulators for CMOS circuits. Existing mixed-signal simulators either cannot handle BiMOS circuits or perform poorly on them since they do not exploit any special characteristics of these circuits.

In this work, we present some techniques for efficient simulation of BiMOS mixed signal circuits, and SYMPHONY, a mixed signal circuit simulator which embodies them. SYMPHONY combines a fast simulator for digital circuits with a SPICE-like nonlinear solver for analog subcircuits. The typical switching behavior of bipolar devices in digital setting is exploited by using a simplified model to approximate the bipolar device characteristics to speed-up the simulation. The problem of minimizing the worst case approximation error is formulated. As the analytical solution of this problem can be very complex, a heuristic is proposed to achieve this by using a PWL model with expanded Chebyshev points as the breakpoints. Dynamic circuit partitioning is combined with an event-driven approach to exploit the latency and multi-rate behavior, and efficiently handle tight coupling and feedback in the circuit.

In the following, Section 2.2.2 details the digital and analog simulation platforms and the event management in SYMPHONY, and Section 2.2.3 describes dynamic partitioning for BiMOS

circuits and its implementation in SYMPHONY. In Section 2.2.4, a new PWL model for bipolar device characteristics is introduced. Section 2.2.5 presents some experimental results on a set of BiMOS circuits and Section 2.2.6 concludes with a summary of the contributions of this work.

## 2.2.2. Simulator Architecture

SYMPHONY is an event-driven mixed signal simulator which combines a fast simulation engine for digital circuits with a traditional nonlinear solver *a la* SPICE for the analog subcircuits. Static and dynamic circuit partitioning (apart from any user-input) are used to identify analog and digital subcircuits in the circuit to be simulated.

The digital simulator uses stepwise equivalence conductance [Lin93b] to model nonlinear device conductances and piecewise linear voltage waveforms. This approach for the digital engine in SYMPHONY is employed in this work since it provides good speed-ups while maintaining high accuracy and guaranteeing consistency, stability, convergency. In the following, a brief review of this approach is presented (for more details please refer to [Lin93b])

Assuming for the sake of simplicity, that there are no inductors and only constant capacitors in the circuit, the circuit equations are of the form

$$\mathcal{F}(V(t)) + C\dot{V}(t) = I_S(t) \quad (\text{EQ 2.1})$$

where  $V(t)$  is the node voltage vector,  $\mathcal{F}(V(t))$  is the current flowing through resistive devices,  $C$  is the constant capacitance matrix, and  $I_S(t)$  is the vector of inputs. The nonlinear system of EQ. 2.1 can be transformed to a linear-time variant system below without any loss of generality

$$G(t)V(t) + C\dot{V}(t) = I_S(t) \quad (\text{EQ 2.2})$$

where  $G(t)$  represents the equivalent conductance matrix for every branch in the circuit at time  $t$ ,

with  $G(t)V(t) = \mathcal{F}(V(t))$  at every time instant  $t$ .  $G(t)$  can be expanded in a Taylor series around  $t = t_n$ . Retaining only the first two term gives,

$$[G(t_n) + \dot{G}(t_n)(t - t_n)]V(t) + C\dot{V}(t) = I_S(t) \quad (\text{EQ 2.3})$$

This can then be further approximated as,

$$\mathcal{G}V(t) + C\dot{V}(t) = I_S(t) \quad (\text{EQ 2.4})$$

where, for  $h_n = t_{n+1} - t_n$ ,  $\mathcal{G} = G(t_n) + \dot{G} \cdot (t_n) \cdot \frac{h_n}{2}$ . EQ. 2.4 can be solved using sparse Gaussian elimination or LU decomposition, and no Newton-Raphson iterations are needed.

The analog subcircuits can require very high accuracy (even at the expense of runtime), which is best provided by traditional, *exact* simulation techniques. Hence, for the analog subcircuits (which are obtained by circuit partitioning or user input), we use the traditional Newton-Raphson iteration-based approach [Nag75].

SYMPHONY uses an event-driven mechanism to take advantage of circuit latency. The time-step is controlled by the validity of the stepwise equivalent conductance approximation for digital subcircuits and an LTE based error control formula for analog subcircuits.

For a MOSFET spanning two regions, the gate voltage is computed by curve-fitting when evaluating the region containing the source/drain nodes. For safe event scheduling, it is required that the gate voltage can be computed correctly any time the corresponding source/drain regions are evaluated.

Since the simulation of analog subcircuits is iteration based, the time step may have to be reduced and the event discarded if the iterations do not converge. Due to this back-tracking in time,

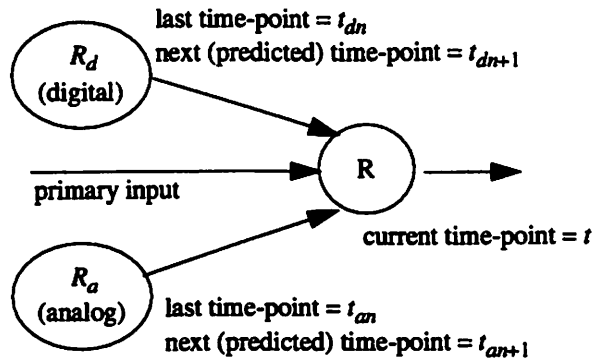


Figure 2-1 : Event Scheduling

voltage waveforms cannot be predicted by extrapolation. Thus, in Fig. 2-1 the voltage of node  $n_a$  (in  $R_a$ ) can be safely determined only up to time  $t_{an}$ .

In case of digital subcircuits, time-steps are selected such that validity of the predictor is guaranteed by construction (using LTE criteria applied to the stepwise equivalence conductance approach). Thus, in Fig. 2-1, the voltage of node  $n_d$  (in  $R_d$ ) can be consistently predicted (by curve-fitting) for any time up to  $t_{dn+1}$ .

Safely evaluating a region  $R$  at time  $t$ , requires that the voltages and slopes of all inputs to the region at time  $t$  be available. For the input from a digital region, this requires that  $t_{dn+1} > t$ . For an input from an analog region, this requires that  $t_{an} > t$ .

Thus, the time counter (event time of the subcircuit under evaluation) does not increase monotonically. Local queues to store events for each region, and a global queue to store events satisfying the safe evaluation criteria, are used. Local and global clocks are used to keep track of the simulation time and limit the window of voltage history that need be maintained.

### 2.2.3. Dynamic Partitioning for Simulation of BiMOS Circuits

Circuit partitioning to improve the speed-up simulation is a well-known technique. Weak coupling

between the gate and drain/source of a MOSFET and unidirectionality of signal propagation can be used to *a priori* (statically) partition a circuit in channel-connected components [Bry84][Kle82].

Bipolar devices have strong coupling between the base and the collector/emitter nodes. Also, unidirectionality of signal propagation cannot be assured. Consequently, static circuit partitioning cannot be applied to bipolar circuits.

While there are several works on dynamic partitioning under different assumptions [Boy78][Sav90a][Vid86][Yu94], the problem of dynamic partitioning for arbitrary BiMOS circuits in an event-driven framework has not been addressed before.

A common feature of BiMOS designs is the large number of predominantly MOS subcircuits with a few bipolar devices as output drivers etc. These subcircuits often represent independent functional blocks in the design. If only static partitioning is used, bipolar devices can act as a glue between two functionally independent regions, resulting in large regions containing many functional blocks. Thus, the latency and multi-rate behavior between these functional blocks under the input stimuli cannot be exploited.

Dynamic partitioning can be used to speed-up the simulation in such cases by taking advantage of the special behavior of bipolar devices in digital subcircuits. These bipolar devices toggle between the *on* and *off* state of the device, with little time spent in the transitions. Since when the bipolar device is *off*, there is no coupling between the base and the collector/emitter nodes, the solution for the base node can be decoupled from the solution for the collector/emitter nodes.

SYMPHONY starts by initially partitioning the circuit at MOSFET gates (if any), and labels the partitions with BJTs as *super-regions*. For each *super-region*, a signal flow graph  $G$  is constructed by inserting a directed edge between nodes  $i$  and  $j$  if there is any coupling between nodes  $i$  and  $j$  in the circuit. At each time point, some of the edges may be *active* (if the bipolar device



connecting the two nodes is *on*) or *inactive* (if the bipolar device is *off*). The signal flow graph  $G$  may have one or more disconnected components after the removal of all *inactive* edges. Each such disconnected component represents a *dynamic* partition of the analog subcircuit. Since there is no coupling between these partitions, each can be simulated independently with different time-points.

During the simulation, whenever a bipolar device changes state, the signal flow graph  $G$  is updated to reflect the deletion/addition of *active* edges. This can result in one dynamic partition being divided in one/more partitions or two dynamic partitions being merged. Merging two partitions can be a problem if both have a different time-stamp. In this case, both regions are synchronized by scheduling them for evaluation at the current simulation time.

#### **2.2.4. A PWL Model for Simulation of Digital Bipolar Circuits**

The stepwise equivalent conductance approximation is best suited for devices whose conductances are not very sensitive to voltage changes. Due to the exponential nature of the bipolar device characteristics, it cannot be applied to simulation of bipolar devices. Using the Newton-Raphson iterations based solver for regions containing bipolar devices can introduce avoidable runtime degradation in cases of BiMOS designs with digital behavior. However, if we are interested in only the typical region of operation of the device, a polynomial model (of user-specified order) can be fitted to the device characteristics and the stepwise equivalent conductance approach applied to it. In the following, this problem is formulated. The exact analytical solution for the best fit-polynomial given the degrees of freedom of the model is non-trivial. Using results from the numerical analysis literature, a heuristic is proposed to generate a good approximation to the device characteristics via a PieceWise Linear (PWL) model. In practice, a 3 to 4 order model (i.e. 3 to 4 PWL segments) was found to be sufficient for good accuracy.

The heuristic proposed here has applicability beyond just applying the stepwise equivalent

conductance to bipolar devices. PWL modeling is applied to approximate a complex, non-linear system in many problems. While there has been a lot of work on algorithms to simulate PWL systems, these assume the existence of a PWL model of the system. The problem of automatic generation of a PWL model has not been addressed before. Obtaining a good approximation to the system function is very critical to accuracy and convergence of these algorithms. The formal method proposed here can be used in such cases to obtain the breakpoints for the PWL approximation, such that the worst case approximation error is minimized.

**Definition 2.1 (approximation error):** Let  $\pi_n$  be the set of all polynomials of degree  $\leq n$ , and  $p_n(x) \in \pi_n$  approximate a given function  $f(x)$  uniformly well on some interval  $a \leq x \leq b$ . The error in the approximation of  $p_n(x)$  to  $f(x)$  is measured by the norm [Con80]:

$$\|f - p\|_\infty = \max_{a \leq x \leq b} |f(x) - p(x)| \quad (\text{EQ 2.5})$$

Ideally, we would want a best uniform approximation from  $\pi_n$ , that is a polynomial  $\hat{p}_n(x)$  of degree  $\leq n$  for which:

$$\|f - \hat{p}_n\|_\infty = \min_{p \in \pi_n} \|f - p\|_\infty \quad (1)$$

We denote the number  $\|f - \hat{p}_n\|_\infty$  by  $\text{dist}_\infty(f, \pi_n)$  and call it the *uniform distance* on the interval  $a \leq x \leq b$  of  $f$  from polynomials of degree  $\leq n$ .

**Theorem 2.1 :** A function  $f$  which is continuous on  $a \leq x \leq b$  has exactly one best uniform approximation on  $a \leq x \leq b$  from  $\pi_n$ . The polynomial  $p \in \pi_n$  is the best uniform approximation to  $f$  on  $a \leq x \leq b$  if and only if there are  $n+2$  points  $a \leq x_0 \leq \dots \leq x_{n+1} \leq b$  so that

$$(-1)^i [f(x_i) - p(x_i)] = \epsilon \|f - p\|_\infty \quad i = 0, \dots, n+1 \quad (\text{EQ 2.6})$$

with  $\varepsilon = \text{signum}[f(x_0) - p(x_0)]$ .

A proof of this theorem can be found in [Con80]. The construction of a best uniform approximation from  $\pi_n$  is, in general, a non-trivial task. By proper *interpolation*, *almost best* approximations to nonlinear functions can be obtained with much less computation.

**Theorem 2.2** : Let  $p_n(x) \in \pi_n$ , interpolate  $f(x)$  at the points  $x_0 < x_1 < \dots < x_n$  in the interval  $a \leq x \leq b$  of interest. Then

$$\text{dist}_\infty(f, \pi_n) \leq \|f - p_n\|_\infty \leq (1 + \|\Lambda_n\|_\infty) \text{dist}_\infty(f, \pi_n) \quad (\text{EQ 2.7})$$

where  $\|\Lambda_n\|_\infty$  is the uniform norm of the Lebesgue function  $\Lambda_n(x)$  given by,

$$\Lambda_n(x) = \sum_{i=0}^n \left| \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} \right| \quad (\text{EQ 2.8})$$

A proof of this theorem can be found in [Con80]. It is thus desirable to choose the interpolation points  $x_0, \dots, x_n$  in  $a \leq x \leq b$  such a way that  $\|\Lambda_n\|_\infty$  be as small as possible. This is *almost* accomplished by the *expanded Chebyshev points* for the interval  $a \leq x \leq b$ , given by

$$x_i = \frac{1}{2} \left[ a + b + (a - b) \cos \frac{2i + 1}{2n + 2} \pi / \cos \frac{\pi}{2n + 2} \right] \quad i = 0, \dots, n \quad (\text{EQ 2.9})$$

It can be shown that the  $\|\Lambda_n^e\|_\infty$ , corresponding to the *expanded Chebyshev points* is within 2% of the smallest possible value of  $\|\Lambda_n\|_\infty$  for all  $n$ . From Theorem 2 and by computing the values of  $\|\Lambda_n^e\|_\infty$ , it can be shown that for  $n \leq 47$ , the error in the polynomial interpolating  $f(x)$  at the *expanded Chebyshev points* is never bigger than 4 times the best possible error  $\text{dist}_\infty(f, \pi_n)$  (obtained by using the best uniform approximation polynomial  $\hat{p}_n(x)$ ), and is usually smaller (by contrast, if  $\Lambda_n^u$  denotes the Lebesgue function for a *uniform* spacing of interpolation

points, then  $\|\Lambda_n^u\|_\infty \geq e^{n/2}$ , which grows very rapidly with  $n$ ).

We use the *expanded Chebyshev points* as the break points for the PWL approximation of the bipolar  $i$ - $v$  characteristics. Since at every instant, the bipolar devices is still modeled by an effective conductance (which is now a PWL function of the terminal voltages) the structure of the system of equations from EQ. 2.4 remains unchanged. Standard iteration-based techniques are used to solve the PWL system [Kat65].

### 2.2.5. Results

The speed-up obtained by SYMPHONY over SPICE3e is due to three techniques: the stepwise equivalence conductance approach, the PWL model for digital bipolar devices, and dynamic partitioning for analog bipolar subcircuits. Experimental results demonstrate the efficiency of each of these techniques. In the following, all runtimes are on a DEC 5100/25 platform with a 24 Mbyte main memory.

Table 2-1 presents experimental results on industrial analog/digital BiMOS circuits. These circuits have bipolar and MOS devices in the analog subcircuits and MOS devices elements in the digital subcircuits. Thus, the speed-up is due to the stepwise equivalence conductance technique for the digital subcircuits and dynamic partitioning for analog subcircuits. Column 1 has the names of the industrial circuits. Column 2 contains the number of devices in the circuit and Column 3 indicates the percentage of these devices in the digital subcircuits. Column 4 and Column 5 compare the runtimes of SYMPHONY and SPICE3e for the input stimuli provided with the circuits and Column 6 contains the relative speed-up achieved by SYMPHONY over SPICE3e for these simulations. The results indicate that, SYMPHONY yields 2x-250x speed-up over SPICE3e. The speed-up increases with the increase in percentage of digital subcircuits in the design. The waveforms comparisons in Fig. 2-2 and Fig. 2-3 demonstrate the high accuracy of SYMPHONY.

Circuit	# devices	% digital subckt	runtime (sec)		Speed-up
			SPICE3e	SYMPHONY	
bicmos inverter	6	0.0	1.3	0.6	2.2
register file	204	11.6	192.1	68.8	2.8
adder	24	33.4	10.8	3.5	3.1
5 bit counter	170	97.9	288.8	13.5	21.4
16 bit multiplier	7034	99.9	305656.4	1224.3	249.7

Table 2-1 : Results on BiMOS benchmark circuits

Circuit	# nodes	SPICE3e	SYMPHONY	Speed-up
bipolar inverter1	4	0.8	0.3	2.7
bipolar inverter2	37	4.2	2.1	2.0
bipolar flip-flop	170	41.8	19.6	2.1
ring oscillator	99	-	38.4	-

Table 2-2 : Results on digital bipolar circuits (A "-" indicates that the program could not converge)

Note that the user can further increase the accuracy in parts of the circuit by specifying them as analog.

Table 2-2 presents experimental results on some MCNC benchmark circuits. These circuits consist of (digital) bipolar devices only. Thus, the speed-up over SPICE is due only to the PWL bipolar device model. Column 1 has the circuit names and Column 2 contains the number of nodes in the circuit. Column 4 and Column 5 compare the runtimes of SYMPHONY and SPICE3e for the input stimuli provided with the circuits and Column 6 contains the relative speed-up achieved by SYMPHONY over SPICE3e for these simulations. The results show that just the PWL bipolar device model can yield speed-ups of 2x-3x over SPICE3e.

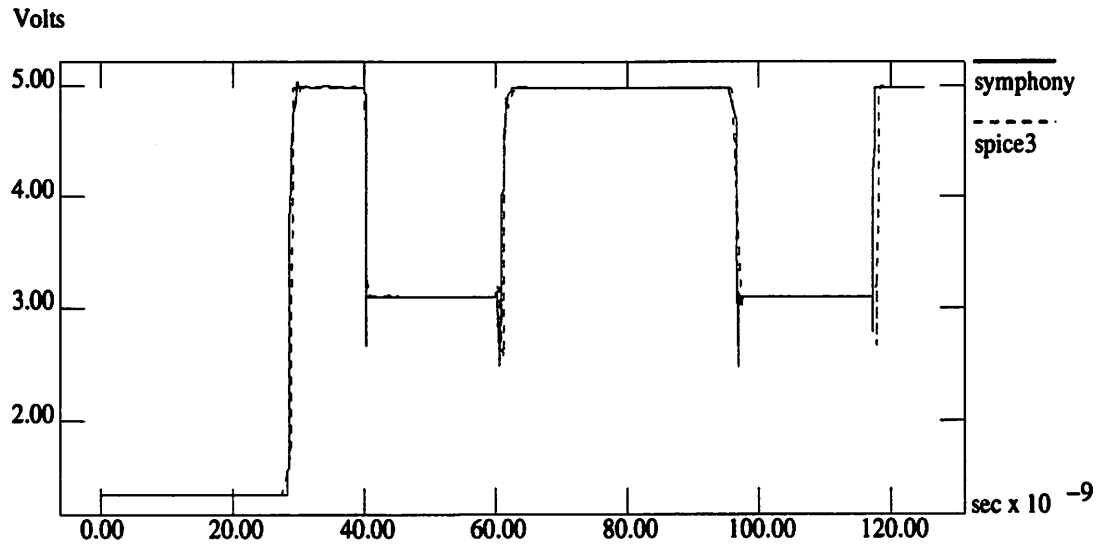


Figure 2-2 : BiCMOS adder: sum bit

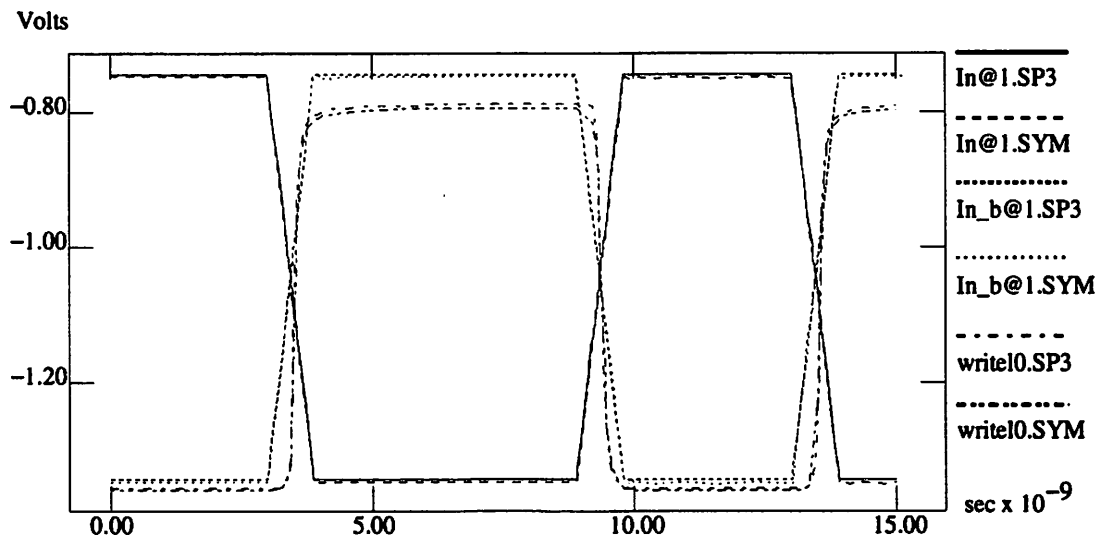


Figure 2-3 : BiCMOS register file: writelines

## 2.2.6. Conclusions

SYMPHONY, an efficient mixed signal simulator was presented. SYMPHONY combines a fast simulator for digital circuits with a traditional nonlinear solver *a la* SPICE for the analog subcircuits. The digital simulator uses stepwise equivalence conductance to model nonlinear device conductances and achieves additional speed-up by modeling the voltages by PWL waveforms. The simulator is implemented in an event-driven framework with local and global clocks for event management.

The main contributions of this work are:

- A new PWL model for device characteristics of bipolar elements in digital subcircuits based on Extended Chebyshev points which minimizes the worst case approximation error.
- Dynamic circuit partitioning to fully exploit the latency and multirate behavior of the circuit.

Experimental results over a suit of BiMOS circuits show that SYMPHONY yields substantial speed-ups over industry-standard circuit simulators.

## 2.3. Power Simulation

### 2.3.1. Introduction

Instantaneous power dissipation of a circuit under a given input sequence can be computed by simulating the circuit for the given input vectors and monitoring its power dissipation. The power dissipation of the circuit is given by:

$$P(t) = \sum_{\text{all elements}} V_i(t) \cdot I_i(t) = \sum_{\text{all supplies}} V_{DD} \cdot I_{DD}(t) \quad (\text{EQ 2.10})$$

where  $V_i(t)$  and  $I_i(t)$  are the instantaneous voltage drop and the current through element  $i$  in the

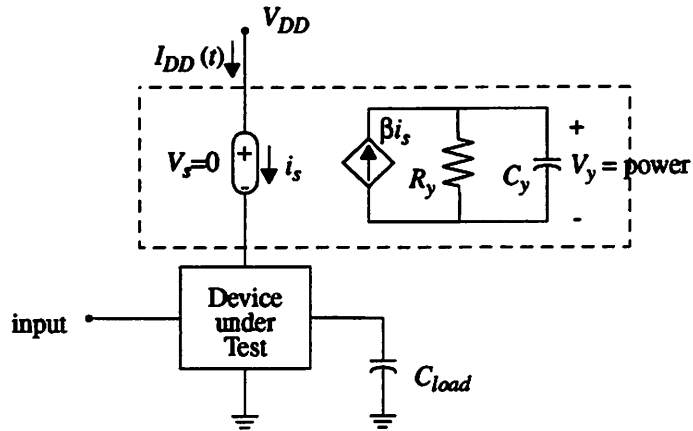


Figure 2-4 : Power simulation in SPICE

circuit,  $V_{DD}$  is the power supply voltage and  $I_{DD}(t)$  is the current drawn through the supply  $V_{DD}$ . Since the circuit simulator computes the instantaneous voltage waveforms for all nodes and current waveforms for all branches, computing the total current drawn from the power supply is a straightforward task using EQ. 2.10. To obtain the power dissipation in a circuit element, the product of  $V_i(t)$  and  $I_i(t)$  in EQ. 2.10 is computed for each time-step using non-linear numerical integration. Instantaneous power dissipation can also be computed by adding a non-invasive power meter element shown in Fig. 2-4 whose parameter ( $\beta, R_y, C_y$ ) values are adjusted such that the voltage drop across the capacitor  $C_y$  is equal to the instantaneous circuit power dissipation [Kan86]. The key to efficiency gains in power simulation then lies in the speed and accuracy of the techniques used in the circuit simulation engine.

In the following, the power simulation algorithm to compute the instantaneous power dissipation and the power dissipation in circuit elements within the SYMPHONY framework is described in Section 2.3.2, the results of applying SYMPHONY to power simulation of industrial circuits are presented in Section 2.3.3, and the contributions of this work are summarized in Section 2.3.4.



### 2.3.2. Power Simulation using SYMPHONY

The stepwise equivalence conductance and the piecewise linear waveform approximation are ideally suited for efficient power computation. The power can be measured directly by monitoring the conductance and the voltage waveform during each time-step. Using EQ. 2.10, The power dissipated in each device during a time step  $h_n$  (from  $t_n$  to  $t_{n+1}$ ) is given by

$$P_d = \frac{1}{h_n} \int_{t_n}^{t_{n+1}} V(t) \mathcal{F}(V(t)) dt \quad (\text{EQ 2.11})$$

Recall that, during each time-step, each nonlinear device conductance is approximated by an equivalent conductance  $\mathcal{G}$  (EQ. 2.4). Thus, EQ. 2.11 can be simplified as:

$$P_d = \frac{1}{h_n} \int_{t_n}^{t_{n+1}} \mathcal{G} \cdot V^2(t) dt \quad (\text{EQ 2.12})$$

Since the voltage waveforms are piecewise linear,  $dV/dt$  is a constant for a given time step. Thus, the power dissipated in each device from  $t_n$  to  $t_{n+1}$  can be obtained by simply computing the area under the power waveform curve given by:

$$P_d = \frac{\mathcal{G}}{h_n} \int_0^{h_n} \left[ V(t_n) + t \cdot \frac{dV}{dt} \Big|_{h_n} \right]^2 dt \quad (\text{EQ 2.13})$$

The power in capacitors can be computed similarly. Specifically, one can directly measure the power consumption during simulation using the piece-wise linearity property of the waveforms in SYMPHONY. For each event during the course of a simulation, we perform the following calculations. Suppose an event changes the voltage across a capacitor  $C_i$  from  $v_0$  at time  $t_0$  to  $v_1$  at time  $t_1$ . Then, the power dissipated from  $t_0$  to  $t_1$  is given by:

$$P_c = \frac{1}{h_n} \cdot C_i \cdot V_{avg} \cdot \frac{dV}{dt} \quad (\text{EQ 2.14})$$

where  $V_{avg}$  is the average value of  $v_0$  and  $v_1$ .  $dV/dt$  is a constant as before. Inductors in the circuit are handled similarly, with the inductor current (computed for transient simulation using the modified nodal analysis) as the controlling variable.

Then average power up to the time  $t_1$  is updated as follows.

$$P_{t_1} = \frac{P_{t_0} \cdot t_0 + P_{t_1-t_0} \cdot (t_1 - t_0)}{t_1} \quad (\text{EQ 2.15})$$

where  $P_{t_1-t_0}$  denotes the power consumed from  $t_0$  to  $t_1$  and  $P_{t_i}$  denotes the power consumed from  $t = 0$  to  $t_i$ . We perform this calculation for every event of the simulation, and finally sum up the power dissipated at every node to obtain the power consumption of the circuit.

### 2.3.3. Results

The power estimator described in the previous section has been implemented in the SYMPHONY framework [Buc95b]. HSPICE [HSP92], the industry standard circuit simulator, and PowerMill [Den94], the industry standard fast power simulator are used along with this program to compare benchmark results. Industrial circuits obtained from LSI Logic Corporation were used for the purpose of this benchmarking. The netlists were extracted from the layout of real designs generated in the design synthesis environment of LSI Logic using their ASIC cell libraries and deep submicron devices. These circuits range in size from 200 to 6000 cell units in the LSI technology. These netlists were then used along with input stimuli to serve as data for HSPICE, PowerMill and SYMPHONY. The results were obtained on a Sun Sparc10 platform with a 512 Mbyte main memory and 865 Mbyte virtual memory.

The circuits used in the experiments are listed in Table 2-3. Column 1 contains the circuit names. Column 2, Column 3 and Column 4 contain the number of cells, the number of MOS and number of capacitors in the design respectively. Column 5 contains a brief description of the circuit functionality.

Table 2-4 shows the power measurement results for these circuits. These results compare the SYMPHONY and PowerMill results with HSPICE, since HSPICE is widely accepted as the most accurate circuit simulator available. Column 2, Column 3, and Column 4 correspond to the power dissipation results reported by HSPICE, PowerMill, and SYMPHONY, respectively for the circuits in Column 1. Columns 5 and Column 6 show the percentage error in the PowerMill and SYMPHONY measurement respectively as compared to HSPICE. It can be seen that in all cases, SYMPHONY is more accurate than PowerMill. Also, note that while PowerMill both under and over-estimates the HSPICE power dissipation, SYMPHONY always provides a conservative power estimate. While this property can be guaranteed theoretically, it is very useful when applying power simulation to guide power optimization.

Table 2-5 compares the runtimes of all three simulators. Column 2, Column 3 and Column 4 contain the runtimes for HSPICE, PowerMill, and SYMPHONY respectively for the circuits in Column 1. Column 5 and Column 6 present the runtime speed-up achieved by PowerMill and SYMPHONY respectively over HSPICE. Note that the speed-ups for both PowerMill and SYMPHONY increase with the size of the circuit, and that in all cases SYMPHONY is faster than PowerMill. PowerMill constructs look-up tables for the device characteristics of each MOS in the circuit and caches them on disk. These tables have to be constructed only once when a design is simulated for the first time, or when a change is made in a previously simulated design. The current implementation of SYMPHONY does not have this optimization. However, this can be implemented in SYMPHONY and does not represent any algorithmic limitation. To give an idea of

the potential savings possible by this optimization, the runtimes for PowerMill excluding the table generation time (as would be the case for a previously simulated design) are reported in parentheses in Column 3.

Circuit	Size (cell units)	MOS devices	Capacitors	Description
mux2b16	208	214	134	16 bit 2-to-1 mux
cla16	993	1200	500	16 bit carry look ahead adder
mult8	1276	2691	1008	8 bit wallace tree multiplier
mult16	5320	9778	3148	16 bit wallace tree multiplier
multp16	6344	11314	3922	16 bit pipelined multiplier

Table 2-3 : Circuits for power simulation benchmarking

Circuit	Power Dissipation (mW)			% Error (w.r.t. HSPICE)	
	HSPICE	PowerMill	SYMPHONY	PowerMill	SYMPHONY
mux2b16	0.685	0.682	0.687	-0.44	0.29
cla16	1.611	1.6761	1.619	4.03	0.49
mult8	11.536	12.932	12.154	12.10	5.36
mult16	51.842	57.231	54.068	10.40	4.29
multp16	40.275	42.697	42.376	6.01	5.09

Table 2-4 : Power simulation accuracy comparison: HSPICE, PowerMill, SYMPHONY

Circuit	Runtime (sec)			Speed-up (over HSPICE)	
	HSPICE	PowerMill	SYMPHONY	PowerMill	SYMPHONY
mux2b16	31.38	30.80 (3.60)	4.57	1.02	6.87
cla16	265.73	228.80 (11.80)	29.63	1.16	8.97
mult8	2509.73	448.90 (37.70)	102.12	5.59	24.58
mult16	42887.88	624.10 (138.20)	599.18	68.72	71.58
multp16	37814.03	683.60 (126.70)	602.78	55.31	62.73
Total	83508.79	2016.20 (318.00)	1338.28	41.42	62.40

Table 2-5 : Power simulation runtime comparison: HSPICE, PowerMill, SYMPHONY

### **2.3.4. Conclusions**

We have presented an approach to power estimation using SYMPHONY. The proposed method exploits the stepwise equivalent conductance approximation to efficiently compute power dissipation while speeding up the transient simulation process. The results demonstrate that SYMPHONY can be used as a fast, accurate engine in any power estimation framework.

## **2.4. Power Estimation**

### **2.4.1. Introduction**

Average power dissipation is a measure of the power a circuit would dissipate when operated for an arbitrarily long period of time. This measure is also called the power rating of the circuit and is used in selecting the appropriate packaging and heat sinks when manufacturing a circuit. Thus, estimating the power rating essentially involves computing the average power dissipation of the circuit over a sampling time as the sampling time approaches infinity. This is usually approximated by simulating the circuit for a long enough time and monitoring the average power dissipation. Then, given a fast and accurate power simulator like SYMPHONY, the main challenge lies in determining how long is long enough for the resulting power estimate to be reliable.

The proposed approach for vector-independent transistor-level power estimation is similar to the work of [Bur93][Hui90] which addresses the problem of vector-independent gate-level power estimation, and combines the accuracy of simulation-based approaches with the weak vector dependence of probabilistic approaches. The theory of [Bur93] is extended here to handle transistor-level analog waveforms. Based on this theory, the power estimation problem is reduced to a mean estimation problem and is solved using a Monte Carlo approach. The resulting approach is statistical in nature; it consists of applying randomly generated input patterns to the circuit and monitoring the resulting power dissipation via a simulator. This is continued until a value of power

is obtained with a desired accuracy, at a specified confidence level.

Additionally, a divide-and-conquer strategy is proposed to reduce the size of the simulation problem during Monte Carlo. While Monte Carlo methods are in general dimension independent, i.e. the number of samples required is independent of the problem size, we gain performance improvements due to two reasons: there is a runtime improvement due to partitioning because the complexity of transistor-level circuit simulation is super-linear. In addition, partitioning allows us to exploit the multi-rate behavior and stiffness of a circuit from a power perspective, thereby improving the validity of the stopping criteria for the Monte Carlo estimation in cases where a single power dissipation model does not fit well. In this context, partitioning can be viewed as a stratified Monte Carlo sampling approach [Ham64].

In the following, details of the reduction of the transistor-level power estimation problem to a mean estimation problem are presented in Section 2.4.2. The set-up of the resulting Monte Carlo problem and its stopping criteria is described in Section 2.4.3. A divide-and-conquer estimation strategy to speed-up the estimation is presented in Section 2.4.4, and a statistical modeling technique for propagating information between each partitions in the divide-and-conquer approach proposed in Section 2.4.5. Some experimental results are presented in Section 2.4.6 and summary of the contributions of this work in Section 2.4.7.

### **2.4.2. A Measure of Switching Activity for Analog Waveforms**

In general, the instantaneous power dissipation  $p(t)$  in a MOS device can be represented as a polynomial function  $f(t)$  of the analog voltage waveform at the drain-source nodes.

$$p(t) = f(V_{ds}(t), V_{gs}(t)) \quad (\text{EQ 2.16})$$

We denote the total average power dissipated in the circuit during time interval  $(-T/2, T/2]$

as  $P_T = \frac{1}{T} \int_{-T/2}^{T/2} p(t) dt$ . The average power dissipation  $P_{avg}$  is then given by

$$P_{avg}(t) = \lim_{T \rightarrow \infty} P_T = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} p(t) dt = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.17})$$

In the following, the power estimation problem is reduced to a mean estimation problem. For this, a companion stochastic process to  $f(t)$  is constructed and it is proved that this process is strict-sense stationary (SSS) and mean-ergodic. The transformation to mean estimation will follow as a result.

The probability of an event  $A$  is denoted by  $Prob\{A\}$  and, if  $x$  is a random variable, its mean is denoted by  $E[x]$  and its distribution function by  $F_x(a) \equiv \mathcal{P}\{x \leq a\}$ . It is assumed that the values of the analog waveforms are bounded by some arbitrary constant  $K_v$ . This is not a restrictive constraint for any real circuit, and in any case, the power estimation problem is not well-defined in presence of unbounded node voltages.

A stochastic process  $f(t)$  is said to be SSS if its statistical properties are invariant to a shift of the time origin [Pap91]. This essentially means that the mean  $E[f(t)]$  of such a process is a constant and independent of time (we will denote it by  $E[f]$ ). By definition, a constant-mean stochastic process is said to be mean-ergodic [Pap91] if its time average tends to its constant-mean as  $T \rightarrow \infty$ .

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt =_1 E[f] \quad (\text{EQ 2.18})$$

where  $=_1$  is used denote convergence everywhere with probability 1.

Let  $\tau \in (-\infty, \infty)$  be a random variable with the probability function  $F_\tau(t) = 1/2$  for any

finite  $t$ ,  $F_{\tau}(-\infty) = 0$ , and  $F_{\tau}(\infty) = 1$ . If  $F_{\tau T}(t)$  is the uniform distribution over  $[-T/2, T/2]$ , then  $\lim_{T \rightarrow \infty} F_{\tau T} = F_{\tau}$ . Thus, one might say that  $\tau$  is uniformly distributed over the whole real line  $\mathcal{R}$ . We now use  $\tau$  to build from  $f(t)$ , a stochastic process  $f(t)$ , defined as follows:

**Definition 2.2 (companion process):** Given a polynomial function  $f(t)$  of an analog signal, and a random variable  $\tau$ , uniformly distributed over  $\mathcal{R}$ , define a stochastic process  $f(t)$  called the companion process of  $f(t)$  given by:  $f(t) \equiv f(t + \tau)$ .

**Proposition 2.1 :** Let  $f(t)$  be a polynomial function of an analog signal. If  $f(t)$  is the companion process of  $f(t)$ , then the following “convergence everywhere” results are true:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.19})$$

**Proof :** To prove EQ. 2.19, it is necessary to show that for any give finite  $\tau_1 \in \mathcal{R}$ , the difference

$$\Delta_a \equiv \frac{1}{T} \int_{-T/2}^{T/2} f(t + \tau_1) dt - \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.20})$$

tends to zero as  $T \rightarrow \infty$ . This can be written as:

$$\Delta_a = \frac{1}{T} \int_{-T/2 + \tau_1}^{T/2 + \tau_1} f(t) dt - \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt = \frac{1}{T} \int_{T/2}^{T/2 + \tau_1} f(t) dt - \frac{1}{T} \int_{-T/2}^{-T/2 + \tau_1} f(t) dt \quad (\text{EQ 2.21})$$

Since  $f(t) \leq K_v$ , then  $|\Delta_a| \leq K_v \frac{|\tau_1|}{T}$  must go to 0 as  $T \rightarrow \infty$ . Since this is true for any  $\tau_1 \in \mathcal{R}$ , then the convergence is everywhere, in the sense that every value of  $\tau$  will lead to convergence.

**Proposition 2.2 :** The companion process  $f(t)$  of a polynomial function  $f(t)$  of an analog signal is SSS and mean-ergodic, with:



$$E[f] = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.22})$$

**Proof :** At  $t = 0$ , we have  $E[f(0)] = E[f(\tau)]$ . An interesting property of  $\tau$  is that if  $a$  is a constant than  $a+\tau$  has the same distribution as  $\tau$ . Indeed, if  $F_{a+\tau}(t)$  is the distribution function of  $a+\tau$ , then  $F_{a+\tau}(t) = \mathcal{P}\{a + \tau \leq t\} = \mathcal{P}\{\tau \leq t - a\} = 1/2 = F_{\tau}(t)$ . Therefore, since  $t+\tau$  and  $\tau$  are identically distributed, we have  $E[f(t+\tau)] = E[f(\tau)]$ , which means that  $f(t)$  is a constant-mean process with  $E[f(0)] = E[f(t)] = E[f(\tau)]$  for any time  $t$ .

To prove mean-ergodicity, consider the random variable:

$$\eta_T \equiv \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.23})$$

from EQ. 2.19:

$$\lim_{T \rightarrow \infty} \eta_T = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.24})$$

where this convergence is everywhere. Therefore:

$$\lim_{T \rightarrow \infty} E[\eta_T] = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.25})$$

By linearity of the expected value operator, this can be rewritten as:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} E[f(t)] dt = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (\text{EQ 2.26})$$

But  $E[f(t)]$  is a constant. Therefore, the left-hand side of EQ. 2.26 is simply  $E[f]$ , and mean-ergodicity follows, with  $E[f]$  given by EQ. 2.22.

Thus, using EQ. 2.17:

$$P_{avg} = E[f] \quad (\text{EQ 2.27})$$

and the problem of estimating  $P_{avg}$  is reduced to the task of computing  $E[f]$ .

### 2.4.3. A Monte Carlo Approach to Transistor Level Power Estimation

The mean estimation problem corresponding to the power estimation problem can be efficiently solved by a Monte Carlo based approach involving monitoring simulation results over a length of time. In order to estimate the expected value of the mean  $E[f](= P_{avg})$ , we observe  $N$  samples of the power dissipation process and use their average  $\mu_N$  as a point estimate of  $E[f]$ :

$$\mu_N = \frac{1}{N} \sum f(t, \tau) \quad (\text{EQ 2.28})$$

An interval estimate of  $\mu_N$  is required in order to guarantee an error bound on this estimate  $\mu_N$  with a certain confidence level. This requires determining the distribution  $\mu_N$ . In general, this is a difficult problem involving multiple convolutions. To simplify it, it is assumed that  $\mu_N$  is normal. This is true if  $f$  is normal.

**Theorem 2.3 (Central Limit Theorem) :** Let  $X_1, X_2, \dots, X_n$  be a random sample of size  $n$  from a population whose distribution has finite mean and variance  $\mu$  and  $\sigma^2$  respectively, and let  $X$  be the sample mean. The random variable  $Z = \sqrt{N}(X - \mu)/\sigma$  has as its limiting distribution as  $n \rightarrow \infty$ , the standard normal distribution.

Since  $P_T$  is the sum of power dissipations at the  $m$  devices in the circuit, under the above

theorem, a sufficient condition for normality of  $P_T$  is that  $m$  be large, and power dissipation in each device be independent. This is true under fairly general conditions irrespective of the individual distributions making up  $P_T$  [Hal64]. In the context of power dissipation, this assumption is reasonable as described below:

In CMOS circuits, there are two components that contribute to power dissipation [Cha92a]: static dissipation (due to leakage current) and dynamic dissipation (due to switching transient current and charging and discharging of load capacitance). Ideally, CMOS circuits dissipate no static power since in the steady state there is no direct path from  $V_{dd}$  to ground. In practice, since the MOS transistor is not a perfect switch, there are always leakage currents and substrate injection currents, which give rise to a static component of CMOS power dissipation. Since the substrate current reaches its maximum for gate voltages near  $0.4V_{dd}$  and since the gate voltages only reside in this range during the switching transients, the actual power contribution of the substrate injection current is a function of the switching of gate-source voltage  $V_{GS}$  of the MOS device and is quite small [Gra94]. Another source of static power dissipation is sub-threshold currents of the transistors. Again, this contribution is dependent on  $V_{GS}$  and is very small for current technologies. Thus the static power dissipation for each MOS device in a static CMOS circuit is a strong function of the corresponding  $V_{GS}$ . The dynamic power dissipation of a circuit is proportional to the switching frequency of its nodes. The power dissipation during one transition from low-high-low depends on the  $V_{ds}$  and  $V_{gs}$  waveforms during the transition.

Thus, while power dissipation in each device in a circuit is a complex, non-linear function of many parameters, it is primarily controlled by its terminal node voltage waveforms. Although node voltages may be locally correlated, in a large circuits under random inputs, there is very little correlation among arbitrary node voltage pairs. Thus the Central Limit theorem can be applied in the power estimation context. [Bur93] discusses a similar proposition for logic level power

estimation. Experimental justification for this assumption and the validity of the stopping criterion when this assumption is violated are presented in Section 2.4.6.

Accordingly, suppose  $P_T$  is normally distributed with a mean  $\mu$  and variance  $\sigma^2$ . Let  $\mu_N$  be the observed mean and  $s_N$  be the observed standard deviation from  $N$  simulations of the circuit, each of length  $T$ . The parameters  $\mu$  and  $\sigma$  of the distribution of  $P_T$  are unknown, and are estimated by  $\mu_N$  and  $s_N$ . Since  $\mu_N$  is the mean of  $N$  stochastically independent observations from a normally distributed population with parameters  $(\mu, \sigma^2)$ ,  $\sqrt{N}(\mu_N - \mu)/\sigma$  is normally distributed with parameters  $(0, 1)$ . Since  $\sigma$  is unknown, if we use the estimate  $s_N$  instead of  $\sigma$ , then the variable  $\sqrt{N}(\mu_N - \mu)/s_N$  has a  $t$ -distribution with  $N-1$  degrees of freedom.

Thus, the hypothesis  $\mu = \mu_N$  can be tested by the  $t$ -test of significance. The critical region at the  $\alpha$  level of significance is now given by

$$\frac{|\mu - \mu_N| \sqrt{N}}{s_N} < t_{\alpha/2} \quad \Rightarrow \quad \frac{|\mu - \mu_N|}{\mu_N} < \frac{t_{\alpha/2} s_N}{\mu_N \sqrt{N}} \quad (\text{EQ 2.29})$$

Thus for a specified percentage error  $\epsilon$  in power estimate, and a given confidence level  $(1-\alpha)$ , we must simulate the circuit until EQ. 2.30 is satisfied.

$$\frac{t_{\alpha/2} s_N}{\mu_N \sqrt{N}} < \epsilon \quad (\text{EQ 2.30})$$

The samples in a Monte Carlo method, as a rule, are independent. We set-up the power estimation problem as a Monte Carlo estimation problem by sampling the power dissipation of a circuit using a circuit simulator. Random numbers are used at the primary inputs. These can be biased to reflect any additional information provided by the user. Independent samples are taken by allowing a settling time between two samples as in [Bur93].

#### 2.4.4. Partitioning for Estimation

In the context of power estimation via simulation, it is important to note that while we want the accuracy that can only be provided by circuit simulation, the final aim is to obtain a power estimate, a scalar quantity, out of the given system. Thus, it is not necessary to preserve the waveform vectors at each internal node of the circuit as long as it is possible to extract statistically significant information about the switching behavior. Motivated by this observation, a partitioning for estimation approach is proposed to speed-up the estimation process. This divide-and-conquer strategy is used to partition the estimation problem itself and is independent of any circuit partitioning used to speed-up the simulation.

A big advantage of our divide-and-conquer approach is that we can model power dissipation in each subcircuit separately. This allows us to exploit the multi-rate behavior and stiffness of a circuit from the power perspective. Depending upon the node switching activity, different parts of the circuit may need different number of node vectors to yield a statistically significant power estimate. In general, a very active node provides more information in the sample of a given length than does a quiet node. If we sampled the entire circuit together, we would be constrained to use the maximum length input vector set required among all nodes. By partitioning the design, we can use vector sets of appropriate length for each partition, thus gaining an over-all speed-up.

Another benefit of partitioning is the improved accuracy in estimating power dissipation in circuits where the normal distribution assumption of the previous section is not a good fit. In [Bur93], it was observed that many circuits have a *double normal* distribution (a special case of *bimodal* distribution where each of the two distributions is normal). This can be caused by circuits which have different parts with widely different power dissipation behavior or different functional modes with different power consumption. Our approach can easily handle cases where different

parts of the circuit have different behavior, since power dissipation in a normal distribution can be fitted to each part separately.

We perform static circuit partitioning based on channel connected components to obtain a fine-grain partitioning of MOS circuits. (In the case of static CMOS circuits, this would correspond to partitioning the circuit in simple logic gates). Since the granularity of this partitioning is very fine, we cluster together many such components to obtain a few partitions covering the entire circuit. A Monte Carlo based approach is used as before to perform statistical estimation of each cluster.

In theory, the number of simulations required to guarantee a confidence level is only weakly dependent on the circuit size [Bur93]. However, in practice, the simulation time will strongly depend on the granularity and quality of partitioning. Since the statistical estimate for each subcircuit is obtained by multiple runs of simulation, signal correlations and all physical affects inside the subcircuit are accounted for in the estimate. However, signal correlations between partitions are only weakly accounted for (via input biasing).

#### **2.4.5. Information Propagation Between Partitions**

The fanouts of a subcircuit can be fanins to other subcircuits in the design. Thus, the inputs to each subcircuit are not independent random variables. This poses two problems: scheduling of partitions for Monte Carlo estimation, and information propagation among partitions.

A cycle-free schedule is generated using a signal flow graph for the circuit with each partition represented by a node in the graph. Selective trace algorithm [New79][Kle82] is used to schedule the partitions for Monte Carlo estimation. Note that local feedback is not a problem as all tightly coupled nodes are clustered in the same component. Global feedback can cause the signal flow graph to be cyclic. We solve this problem by clustering the components forming a global cycle

into one big partition. This is not a major constraint since the number of partitions does not have to be very large.

We also need to account for the fact that inputs to a partition may be fanouts of other partitions in the design. One solution would be to simply propagate the output waveforms of one partition to all its fanout partitions. However, this would constraint that all partitions use the same number of vectors, thus prohibiting us from exploiting the multi-rate behavior and stiffness of the circuit. To solve this problem we construct a statistical model to each output and use this model to generate samples of this signals as required by the partitions it is fanning out to. We fit a normal distribution model to the analog signal waveforms at the outputs. The mean is obtained by computing the area under the waveform and the variance is obtained using this mean and the average waveform value during each clock cycle of the simulation. This model is consistent with our initial conjecture that device (and consequently nodal) waveforms are distributed normally. These parameters of normal distribution are then used to bias the input vector generation of the fanouts of the subcircuit.

Note that this approach does not account for spatiotemporal correlations between signals crossing subcircuit boundaries. This creates a trade-off between speed and accuracy. A very fine granularity partitioning speeds-up the simulation process but can be inaccurate since the error cause by neglecting the spatiotemporal correlations can dominate over the accuracy advantage gained by partitioning above; while in absence of partitioning, the runtimes can be very high. Spatiotemporal correlations can be approximately accounted by using pairwise correlations between signals to bias the input patterns generations for such signals.

#### **2.4.6. Results**

The algorithms outlined above were implemented in a power estimation program called

<b>Circuit</b>	<b># MOS</b>	<b>Description</b>
count32	174	32-bit counter
mux2b16	214	1 bit 2-to-1 mux
ripple4	442	16 bit comparator
cla16	1200	16 bit carry look ahead adder
mult8	2691	8 bit wallace tree multiplier
mult16	9778	16 bit wallace tree multiplier
multp16	11314	16 bit pipelined multiplier
m16	6323	processor block

Table 2-6 : Circuits for power estimation benchmarking

SYMPHONY-MC. There is no constraint on the choice of the internal circuit simulation engine for the Monte Carlo samplings. In our implementation the SYMPHONY circuit simulator was used since it provides good speed-ups at demonstrated high accuracy [Buc96b][Lin93b]. For the purpose of this benchmarking, we used industrial circuits obtained from LSI Logic Corporation. The netlists were extracted from the layout of real designs generated in the design synthesis environment of LSI Logic using their ASIC cell libraries and deep submicron devices. The results were obtained on a DEC 5000 platform with a 24 Mbyte main memory.

The circuits used in the experiments are listed in Table 2-6. Column 1 contains the circuit names. Column 2 contains the number of MOS in the design and Column 3 contains a brief description of the circuit functionality.

We compare the power estimation results from SYMPHONY-MC against two metrics: direct circuit simulation results using very long random input vector sets and vector sets with maximum switching at the primary inputs. The first comparison is intended to demonstrate the accuracy of the proposed Monte Carlo approach while the second comparison demonstrates the inadequacy of direct simulation in the absence of a formal vector set selection process



Circuit	Steady State Power Dissipation			Monte Carlo Estimation			% error
	# vectors	runtime	power ( $\mu$ W)	# vectors	runtime (sec)	power ( $\mu$ W)	
count32	10000	19.20 hr	4.25	1857	12461.15	4.14	2.59
mux2b16	10000	5.86 hr	190.11	103	261.22	189.77	0.18
ripple4	10000	33.17 hr	1587.28	468	2783.13	1665.31	4.91
cla16	10000	36.49 hr	795.46	80	1196.10	821.82	3.31
mult8	10000	26.06 hr	3951.45	48	1763.77	4118.40	4.22
mult16	2500*	46.46 hr	17888.04	38	6168.53	18404.70	2.81
multp16	5000*	77.44 hr	13173.72	43	8028.01	13805.30	4.79
m16	-*	-	-	297	44.84 hr	751.04	-

Table 2-7 : Power estimation accuracy and runtime comparison

Table 2-7 contains the results of the statistical power estimator on the benchmark circuits. Column 2, Column 3 and Column 4 present the runtime and power dissipation data for direct simulation of benchmark circuits of Column 1 with 10000 vectors. This was used to represent the *real* power dissipation of the circuit since it is an average of power dissipation over an extremely long simulation sample. Column 5 reports the number of vectors needed for SYMPHONY-MC to converge to a power estimate. Column 6 reports the runtime for this estimate and Column 7 contains the power dissipation estimate. For these results, a 5% error tolerance with a 90% confidence interval was used to determine the stopping criteria. Column 8 contains the actual error in SYMPHONY-MC estimate with the above stopping criteria when compared with the 10000 vector simulation. All errors were found to be within the required bound of 5%. The results clearly indicate that the stopping criteria predicts the error well, and that only a small set of vectors is required to obtain power estimates with good accuracy. This can result in large runtime savings while meeting the error specifications of the user, and without compromising the confidence in the estimate.

One of the key assumptions in the derivation of the stopping criteria of the Monte Carlo

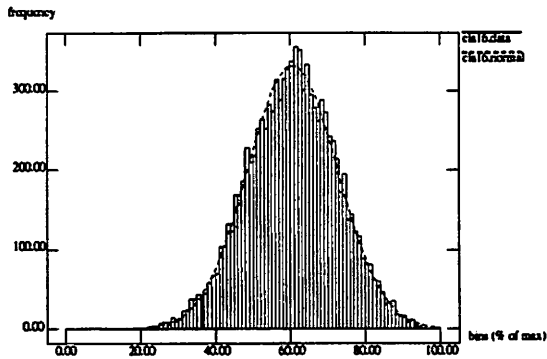


Figure 2-5 : Power dissipation distribution of the 16-bit carry look ahead adder

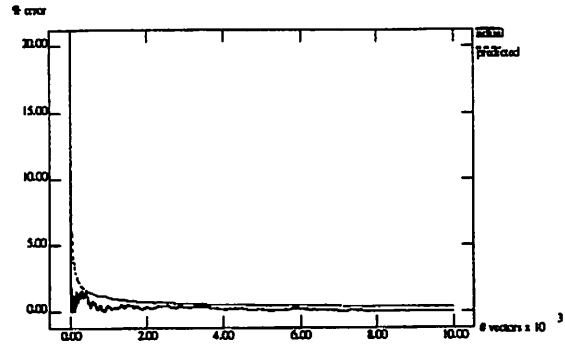


Figure 2-6 : Predicted and actual error for the 16-bit carry look ahead adder

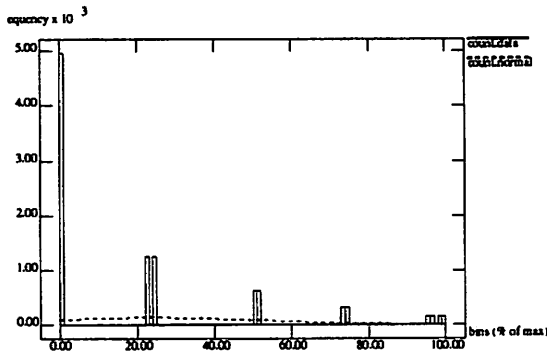


Figure 2-7 : Power dissipation distribution of the 32-bit counter

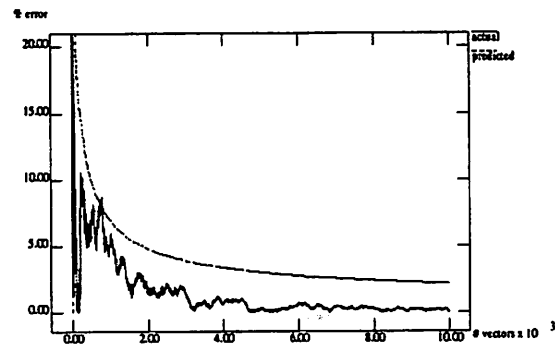


Figure 2-8 : Predicted and actual error for the 32-bit counter

estimator was that  $P_T$ , the power dissipation in a clock cycle, is normally distributed. In Section 2.4.3 we indicated that while this cannot be proved theoretically, under most general conditions this is a good approximation. Fig. 2-5 to Fig. 2-8 examine this assertion in more detail on two cases among our test circuits which are the best and the poorest fit for the normal distribution assumption used to derive the stopping criteria.

Fig. 2-5 compares the distribution of power dissipation data from a 10000 vector random simulation of a 16-bit carry look ahead adder circuit with a normal distribution (with the mean and

standard distribution derived to fit the data). It can be seen that the normal distribution is indeed a good approximation to the sampled data. To further test our proposition on this example case, we applied the  $\chi^2$  Goodness-of-Fit test [Hal64][Pea00] to the simulation data. It was found that the null hypothesis that  $P_T$  is normally distributed satisfied the  $\chi^2$  test with a type I error  $\alpha$  of 0.05. For this circuit the predicted error bound and the actual error are plotted in Fig. 2-6. Clearly, the error bound is a conservative, tight bound on the actual error.

In Fig. 2-7, we compare the distribution of power dissipation data from a 10000 vector random simulation of a 32-bit counter with a normal distribution (with the mean and standard distribution derived to fit the data). This circuit has five distinct modes depending upon which bit of the counter is high and clearly, the normal distribution is a very poor approximation to the actual power dissipation for this circuit. Even in this case, we found (Fig. 2-8) that the predicted error bound was a conservative bound on the actual error (although not as tight as in the best fit case).

In general, it is possible that  $P_T$  is not normally distributed. [Bur93] describes some such examples encountered in logic level power estimation, and their effect on overall accuracy. We expect that at the transistor level this effect would be even less pronounced since the number of variables (nodal power dissipation) is higher than at the logic level, thus causing the total power dissipation to be *better behaved* from the perspective of applying the Central Limit theorem as described in Section 2.4.3.

Table 2-8 compares the power estimates from the Monte Carlo approach and the 10000 vector direct simulation to a direct simulation using a small vector set with the worst case switching at the primary inputs. Column 2 reports the power dissipation results for circuits in Column 1 when each primary input is switches at every clock edge. Column 3 and Column 4 reproduce the power dissipation data from the Column 4 and Column 7 respectively of Table 2-7 for comparison. The results demonstrate that if the inputs are not biased properly and/or the input vector set is not

Circuit	power ( $\mu$ W)		
	Maximum switching at PI	Steady State Dissipation	Monte Carlo Estimation
count32	9.89	4.25	4.14
mux2b16	687.25	190.11	189.77
ripple4	2702.15	1587.28	1665.31
cla16	1619.02	795.46	821.82
mult8	12154.30	3951.45	4118.40
mult16	54067.00	17888.04	18404.70
multp16	42325.30	13173.72	13805.30

Table 2-8 : Comparison between direct simulation with worst case primary input switching, and the statistical approach

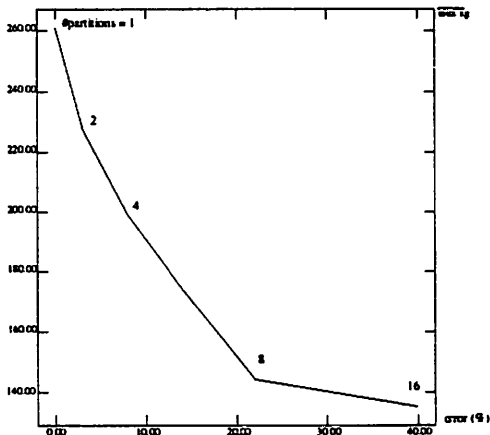


Figure 2-9 : Runtime vs. err or trade-off as a function of number of partitions: 16-bit 2-to-1 multiplexor

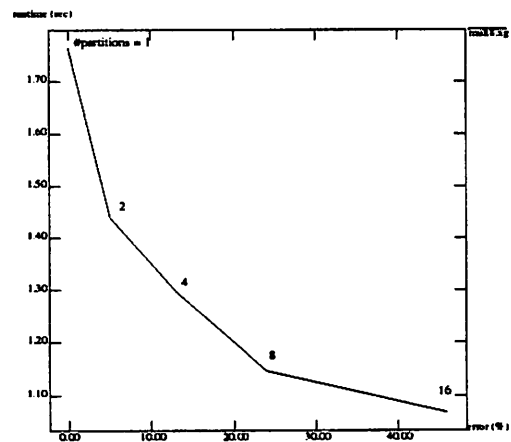


Figure 2-10 : Runtime vs. err or trade-off as a function of number of partitions: 8-bit wallace multiplier

statistically large enough, a simulation based result can be off the actual power dissipation by as much as 300%

Fig. 2-9 and Fig. 2-10 show the effect of varying the granularity of the partitioning for estimation on the accuracy and runtime of the program, with the runtime on the y-axis and the error

relative to the single partition estimate on the  $x$ -axis. The results demonstrate that partitioning does speed-up the estimation process. As mentioned in Section 2.4.4, since the statistical model used for information propagation between partitions does not account for the spatiotemporal correlations, accuracy considerations may limit the number of partitions employed in this approach.

### **2.4.7. Conclusions**

We have applied a statistical approach to power estimation at the transistor level. While transistor level power estimation can be very accurate, for practical usage, it suffers from two disadvantages: large runtimes, and pattern dependency. We proposed a two-point approach to overcome these problems: we use a Monte Carlo based method for estimation without requiring external input stimuli and a partitioning for estimation approach to speed-up the estimation process.

The main contributions of this work are:

- Extension of the concept of transitional density [Naj94a] (used at the gate level power estimation [Bur93]) to analog waveforms. It was shown that a similar approach can be adopted for power estimation at the transistor level. Specifically, it was proved that for each device voltage variable, a companion stochastic process can be constructed which converges to the device power dissipation everywhere and is strict-sense stationary and mean-ergodic.
- A formal stopping criterion to guarantee a desired error bound with a specified confidence level, under the normal distribution assumption. The validity of this approach was demonstrated with experimental results and the implications of its violation analyzed.
- Partitioning for estimation to speed-up the power estimation problem. It was shown that this approach can be used to exploit the multi-rate behavior and stiffness of a circuit from a power perspective to reduce the number of vectors required for the estimation. It was also shown that

this can also improve the accuracy of the power estimate in cases of *stiff* circuits (where a single normal distribution may not be an appropriate model for the power dissipation of the entire circuit).

- A statistical approach for propagating signals and associated switching probabilities between partitions. A statistical model was fitted to the output waveforms of a subcircuit and this model was used to generate input patterns for all its fanout subcircuits.

## **2.5. Summary**

Power estimation algorithms for average power estimation and instantaneous power computation were presented for transistor-level circuits. The proposed approach consists of two parts: a fast circuit simulation engine at the core of the estimation problem, and a statistical technique to obtain accurate, vector-independent power estimates using the fast simulation engine.

The process of circuit simulation involves solving nonlinear, time-varying system of circuit equations. The stepwise equivalent conductance model approximates the conductance of a nonlinear device by a constant equivalent conductance during each time-step of the transient simulation to speed-up the simulation of digital circuits. This was combined with a SPICE-like engine for analog subcircuit simulation in SYMPHONY, a fast mixed signal analog/digital circuit simulator. A new PWL model for digital bipolar devices and a dynamic partitioning strategy for BiMOS circuits were presented. Experimental results demonstrate that SYMPHONY can yield 2x-250x speed-up over SPICE3e; and is up to two orders of magnitude faster than HSPICE and more accurate and faster than PowerMill when applied to power simulation.

A stochastic model for power dissipation at the transistor level was used to reduce the power estimation problem to a mean estimation problem. A Monte Carlo approach was used for mean estimation to obtain vector-independent power estimates within prescribed error bounds with

a user-specified confidence level. A divide-and-conquer approach was used to further speed up estimation. A statistical model was used to propagate signal information between partitions of a circuit. Experimental results showed that the Monte Carlo power estimator converged to a power estimate in very reasonable runtimes and met the prescribed error bounds in all cases.

---

# 3 Logic Synthesis for Low Power Design

---

## 3.1. Introduction

Power dissipation of a system is influenced by decisions made at every stage of the design process starting from the behavioral, to logic and transistor levels of abstraction. In order to design low power, high performance integrated circuits, we need to set and meet power budgets at each level of design. This raises a need for power characterization and optimization techniques at each stage of the design flow. The objectives that these techniques need to serve, change as we move through different levels. The scope of power savings is maximum at the very highest levels of abstraction. At this level, decisions on the algorithm and architecture of the design can dramatically influence the power dissipation of the chip [Keu94]. There have been several publications on algorithms to automate the high level synthesis process for power optimization [Cha95]. However, it is very difficult to accurately predict the impact of different design choices using formal methods. Since the size of the specification and the number of variable parameters is small at this stage, this problem is typically solved manually, relying primarily on designer experience.

At the logic synthesis level on the other hand, the problem size and the number of variable parameters (e.g. logic associated with nodes in the circuit, load capacitance on each node etc.) is too large to lend itself to manual optimization. Thus, while the optimization problem is constrained by the high level decisions made earlier in the design flow (thus limiting the possible power savings), there is a great need for CAD algorithms for low power logic synthesis.



From Section 1.3, power dissipation of a logic gate depends on the load capacitance connected to the gate output and the frequency at which the gate output switches. While node capacitances can be estimated fairly accurately at the logic level, switching activity is a strong function of the input vectors. As a result, a big challenge in designing algorithms for low power logic synthesis is to model power dissipation accurately enough so that all the power gains achieved at the logic level do translate to power gains at the silicon level. It is thus very important to ensure that the complexity of the algorithm is consistent with the accuracy limits of the models employed, and that the power optimization techniques for logic synthesis do not trade-off traditional optimization criteria like delay and area (for which more mature logic level models exist, making the savings at logic level more reliable) for small savings in power. With this in mind, the main objective of this work is to make significant reductions in power dissipation during logic synthesis without compromising delay or area.

Logic synthesis is the process of transforming a set of Boolean functions, obtained from the register transfer level structure, in to a network of gates in a particular technology, while optimizing the network for delay, power, area etc. In the traditional flow targeted at the static CMOS based standard cell technology, the implementation is usually in the form of multiple level logic circuit, represented by a multi-level Boolean network. The traditional logic synthesis process is usually divided into technology independent and technology dependent phases ([Bar87][Bra87]) as shown in Fig. 3-1. The objective of the technology independent phase is to simplify the multi-level Boolean network as much as possible without making any assumptions about the underlying technology. A network so optimized is then mapped to a gate level circuit implementation using a specific pre-characterized cell library in the technology dependent phase. In this work, we address the problem of power optimization at both technology independent and dependent level.

At both technology independent and dependent level, logic synthesis uses the fact that

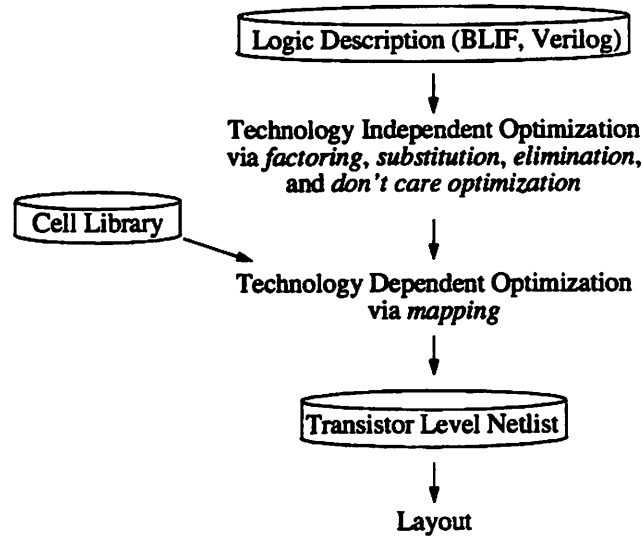


Figure 3-1 : The traditional static CMOS synthesis flow

nodes internal to a network generally do not have a uniquely specified function for satisfying correctness of an implementation. A subset of the Boolean space known as the Don't Care (DC) set [Sav90b] can be generated at each node which gives the range of functionality possible at the node. These include input combinations for which the designer does not care about the output, which cannot affect the output, or those that cannot occur. The functionality of nodes can then be manipulated within the DC set to minimize area, delay or power.

The proposed algorithms exploit some of the statistical properties of minterm probability distribution in the Boolean space. The algorithm at the technology independent phase employs guided node simplification, while optimization at the technology dependent phase uses rewiring. This takes advantage of the fact that more optimization flexibility is available at the technology independent phase to search the solution space, while at the technology dependent phase, the design is more stable and can be characterized more accurately, thus making smaller, local changes more appropriate.

In the following, an efficient technique is presented for exploring the Boolean space to

identify minterms most suitable for influencing switching activity. The notion of power sensitive minterms is introduced to capture the variance in minterm probability distribution. The overlap of power sensitive minterms, which cover a large fraction of the probability space but a very small fraction of the Boolean space, with the DC set is used to bias technology independent area/delay optimization towards reducing switching activity. Experimental results indicate that power can be reduced by as much as 46%, with an average reduction of 16%, without any area penalty.

At the technology dependent level, a power optimization algorithm is proposed which formulates the problem of hot spot reduction as a variant of the engineering change (EC) problem. A technique is presented for determining the sensitivity of circuit power dissipation to functional changes considering both local and global effects. This sensitivity is combined with a measure of synthesis flexibility to identify hot regions in the circuit which have a lot of flexibility in making functional changes and for whom a small functional change can greatly affect the overall power dissipation. An incompletely specified target function is constructed for the hot region such that any implementation satisfying it is expected to reduce power. A rewiring algorithm is used to solve the resulting EC problem without affecting circuit area, gate capacitance or delay under the unit delay model. Experimental results on a set of MCNC benchmark circuits show that the proposed approach can give up to 13% reduction in power dissipation with an average reduction of 4%.

The rest of the chapter is organized as follows: After introducing the necessary terminology in Section 3.2, algorithms for the problem of technology independent and technology dependent logic synthesis for low power are described in Section 3.3 and Section 3.4 respectively. Section 3.5 concludes with a summary of the work presented in this chapter on power optimization techniques within the framework of traditional logic synthesis.

## 3.2. Preliminaries

In the following we introduce some basic definitions and concepts essential for describing the work in this chapter. For a more detailed and rigorous definition of the terminology, the interested reader is referred to [Bra84].

**Definition 3.1 (Boolean functions):** A completely specified Boolean function  $f$  with  $n$  inputs and  $l$  outputs is a mapping  $f: B^n \rightarrow B^l$ , where  $B = \{0, 1\}$ . In particular, if  $l = 1$ , the onset and offset of  $f$  are given by:  $onset = \{m \in B^n \mid f(m) = 1\}$  and  $offset = \{m \in B^n \mid f(m) = 0\}$

An incompletely specified Boolean function  $\mathcal{F}$  with  $n$  inputs and  $l$  outputs is a mapping  $\mathcal{F}: B^n \rightarrow Y^l$ , where  $Y = \{0, 1, X\}$ . The symbol  $X$  implies that the function can be either 0 or 1. In particular, if  $l = 1$ , the onset, offset and the don't care (DC) set of  $\mathcal{F}$  are given by:  $onset = \{m \in B^n \mid \mathcal{F}(m) = 1\}$ ,  $offset = \{m \in B^n \mid \mathcal{F}(m) = 0\}$ , and  $DCset = \{m \in B^n \mid \mathcal{F}(m) = X\}$ . An incompletely specified function  $\mathcal{F}$  is usually described as  $(f, d, r)$ , where  $f$ ,  $d$ , and  $r$  are the onset, DC set and the offset of  $\mathcal{F}$  respectively. A completely specified function  $f$  can be similarly described by  $(f, \emptyset, f')$ . In future, when there is no ambiguity, we will denote a completely specified function by its onset.

A *vertex* or a *minterm* is an  $n$ -tuple in  $B^n$ . A cover of a function  $(f, d, r)$ , is any completely specified function  $f$  such that  $f' \subseteq f$  and  $\bar{f}' \subseteq r$  (e.g.  $x_1$  is a cover of  $(x_1x_2, x_1\bar{x}_2 + \bar{x}_1x_2, \bar{x}_1\bar{x}_2)$ ). A literal is a variable in its true or complemented form (e.g.  $x_1$  or  $\bar{x}_1$ ). A cube is a conjunction of some set of literals (e.g.  $x_1\bar{x}_2$ ).

**Definition 3.2 (Boolean Network):** A Boolean network  $N$  is a directed acyclic graph such that for each node  $i$  in  $N$  there is associated Boolean function  $f_i$ , and a Boolean variable  $y_i$ , where  $y_i = f_i$ , there is a directed edge  $(i, j)$  from  $y_i$  to  $y_j$  if  $f_j$  depends explicitly on  $y_i$  or  $\bar{y}_i$ . A node  $y_i$  is a fanin of node  $y_j$  if there is a directed edge  $(i, j)$  and a fanout if there is a directed edge  $(j, i)$ . A node  $y_i$  is in

the transitive fanin (TFI) of a node  $y_j$  if there is a directed path from  $y_i$  to  $y_j$ , and in the transitive fanout (TFO) is there is a directed path from  $y_j$  to  $y_i$ . Primary inputs  $x = \{x_1, \dots, x_n\}$  are inputs of Boolean network and primary outputs  $z = \{z_1, \dots, z_m\}$  are its outputs. Intermediate nodes of the Boolean network have at least one fanin and one fanout. The global function  $f_i^g$  at  $y_i$  is the function at the node  $i$  expressed in terms of primary inputs.

The don't cares at node  $y_i$  are the don't cares in the function specification of  $f_i$ . Since the actual circuit implementation cannot be incompletely specified, the task of multi-level logic optimization is to select an optimal cover for each node in the network. The DC of  $f_i$  provide the flexibility in this selection process called node simplification. In practice, node simplification is performed by using a two-level minimizer (such as ESPRESSO [Bra84][Rud87]) to optimize each node in the network with the DC derived from the environment of the node. These DC arise from various sources: external don't cares (XDC - input combination for which the designer does not care about the circuit output), satisfiability don't cares (SDC - input combinations at the node inputs which can never occur) and observability don't cares (ODC - input combinations for which the node output does not affect the primary outputs). Using the maximum possible DC (MSPF - maximum set of permissible functions [Mur89]) to simplify a node is not computationally viable because MSPF are very expensive to compute and the choice of cover at a node using its MSPF will affect the ODC of other nodes in the network, which means that the DC for all nodes have to be recomputed after simplifying any node [Sat90]. In practice, a subset of DC called compatible DC (CDC) is used for node simplification which allows all nodes to be optimized simultaneously without ODC having to be recomputed. This is based on the notion of compatible set of permissible functions (CSPF) introduced by [Mur89] for a network of NOR gates and generalized for arbitrary networks by [Sav90b]. Multi-level logic synthesis then consists of using this DC flexibility to optimize the circuit for delay, power, area etc.

In this work, the well accepted switching-activity dominated model of [Cha92a] is used to model power dissipation and guide logic synthesis for power. Namely

$$P(x_i) = \frac{1}{2} \cdot C_i \cdot \frac{V_{dd}^2}{T} \cdot E(x_i) \quad (\text{EQ 3.1})$$

where  $P(x_i)$  denotes the average power dissipated by node  $x_i$ ,  $C_i$  is the load capacitance connected to the node  $x_i$ ,  $V_{dd}$  is the supply voltage,  $T$  is the clock period, and  $E(x_i)$  is the average number of transitions per clock cycle for node  $x_i$ . In the following we introduce some basic terminology and definitions needed to relate the average number of gate transitions per clock cycle to the probability of the gate output being high. This is used to transform the problem of changing the gate power dissipation to changing the function implemented at the gate.

**Definition 3.3 (signal probability [Par75]):** The *signal probability*  $p(x)$  at node  $x$  is defined as the average fraction of clock cycles in which the steady state value of  $x$  is a logic high.

Let  $B^n$  be the Boolean space described by the Boolean variable set  $x = \{x_1, \dots, x_n\}$  where  $x_i \in B^1 \forall i = 0, 1, \dots, n$ .

For any minterm  $m \in B^n$ , let *phase*:  $B^1 \times B^n \rightarrow B^1$  be defined as

$$\begin{aligned} \text{phase}(x_i, m) &= 1 && \text{if } x_i \cdot m = m \\ &= 0 && \text{otherwise} \end{aligned} \quad (\text{EQ 3.2})$$

that is,  $\text{phase}(x_i, m)$  returns the phase of variable  $x_i$  in the minterm  $m$ . Then the *minterm signal probability* (or *minterm probability*) is defined as:

$$p(m) = \prod_{i=0}^n (p(x_i) \cdot \text{phase}(x_i, m) + (1 - p(x_i)) \cdot (1 - \text{phase}(x_i, m))) \quad (\text{EQ 3.3})$$

where  $p(x_i) \in [0, 1]$  is the signal probability at input node  $x_i$ . The *function signal probability* (or *function probability*)  $p(f)$  for a completely specified function  $f: B^n \rightarrow B^1$  with onset  $f_{on}$ ,  $f_{on} \subseteq B^n$ , is defined as:

$$p(f) = \sum_{m \in f_{on}} p(m) \quad (\text{EQ 3.4})$$

**Definition 3.4 (transition probability [Naj94b]):** The *transition probability*  $p_t(x)$  at node  $x$  is defined as the average fraction of clock cycles in which the steady state value of  $x$  is different from its initial value.

Note that the transition probability is independent of the internal delays in the circuit and hence does not account for glitching power (power dissipation due to spurious transitions which do not effect the steady state output). This is the same as assuming a zero-delay model for the circuit. Then, under the assumption of a zero-delay model, the power dissipation formula of (EQ 3.1) reduces, to

$$P(x_i) = \frac{1}{2} \cdot C_i \cdot \frac{V_{dd}^2}{T} \cdot p_t(x_i) \quad (\text{EQ 3.5})$$

This is a valid assumption in estimating power dissipation if the functional switching power is the predominant effect in determining the power consumption. In this work power estimates obtained under this assumption are used to guide the power optimization. This is reasonable since the focus of this work is on functional power dissipation and the changes in the circuit made during the optimization are uncorrelated to glitching power, i.e. any change to reduce the functional switching power has random effect on the glitching power.

Computing the transition probability for nodes in a circuit is itself not a trivial task since

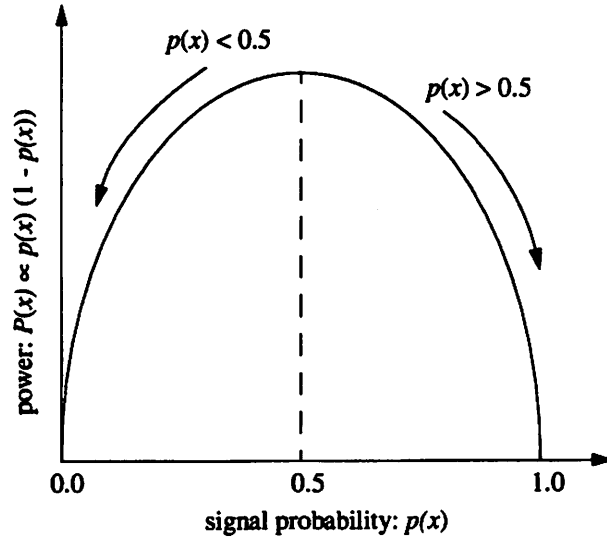


Figure 3-2: Power dissipation at node vs. node signal probability

signals in a circuit may be correlated. These correlations can be due to dependencies between different signal lines due to reconvergent fanouts (spatial correlations), as well as correlations induced by the sequence of inputs applied to the circuit (temporal correlations). It is computationally very expensive to account for these correlations, and hence the circuit inputs and internal nodes are usually assumed to be (spatiotemporally) independent. Under this assumption, the transition probability can be easily obtained from the signal probability by

$$p_t(x) = 2 \cdot p(x) \cdot (1 - p(x)) \quad (\text{EQ 3.6})$$

and (EQ 3.6) reduces to

$$P(x_i) = C_i \cdot \frac{V_{dd}^2}{T} \cdot p(x_i) \cdot (1 - p(x_i)) \quad (\text{EQ 3.7})$$

Fig. 3-2 plots the power dissipation at a node  $x$  as a function the signal probability of  $x$  when the load capacitance at  $x$  is constant. In this case, functional power minimization is equivalent to optimizing signal probability  $p(x)$  to be close to zero or one. The following addresses the problem



of making functional changes in a circuit using the DC to change the signal probability at a node in the desired direction at the technology independent and dependent stage of logic synthesis.

### **3.3. Technology Independent Logic Synthesis for Low Power**

#### **3.3.1. Introduction**

Manipulating the DC flexibility is at the core of technology independent logic optimization. Section 3.2 describes the different classes of DC that can be generated at each node to construct the incompletely specified function providing the range of functionality possible during a valid optimization step. In practice CDC are used for node simplification allowing the DC sets to be constructed a single time for the entire network in the beginning of the optimization phase. This construction is order-dependent and traditional technology independent multi-level logic optimization consists of traversing the multi-level logic network from primary outputs to primary inputs and applying two-level logic minimization to simplify each complex node in the network. In this work, we do not concern ourselves with the DC generation or the core two-level logic minimization algorithm (e.g. ESPRESSO [Rud87]). Instead, we perform power optimization by controlling the DC flexibility provided to the two-level minimizer to guide the minimization in the desired direction. Controlling the DC flexibility involves identifying the parts of the DC set beneficial to power optimization.

In the following, Section 3.3.2 summarizes the previous work in this area. The concept of the probability distribution within the Boolean space is described in Section 3.3.3 and based on this, the notion of Power Sensitive Minterms is defined in Section 3.3.4 to identify the parts of DC beneficial to power optimization. Section 3.3.5 outlines an algorithm for the use of these minterms in power optimization. Section 3.3.6 is a presentation of the results of applying this theory to standard benchmarks, and Section 3.3.7 concludes with a summary of the contributions of this work.

### 3.3.2. Previous Work

Low-power synthesis algorithms which address the issue of distributed input switching probabilities already exist [Ima94][Ima95]. These approaches either try to reduce the functional support from high-activity inputs or guide sub-expression extraction through simple high-level power approximations. However, a formalization of minterm probability distributions within the Boolean space is not described in any of these works.

### 3.3.3. Partitioning the Boolean Space

To define the concept of a probability distribution throughout the Boolean space, consider first a simple example. Take the Boolean space described by two variables  $\{x, y\}$  where  $p(x) = 0.20$  and  $p(y) = 0.45$ . There are four minterms which describe this space:  $\{\bar{x}\bar{y}, \bar{x}y, x\bar{y}, xy\}$ , which have respective probabilities:  $\{0.44, 0.36, 0.11, 0.09\}$ . For this example there are no two minterms with the same probability of occurrence, and 80% of the total probability is contained by the two minterms within the space described by  $f(x,y) = \bar{x}$ . Note also that if the space is partitioned according to:  $f(x,y) = x$  and  $f(x,y) = \bar{x}$ , the minterm probabilities in each partition only differ by at most 0.08, allowing them to be well approximated by their respective averages.

This approximate equality in minterm probability within a partition implies that functional probability is well approximated by a simple proportionality relationship to the number of onset minterms contained in each partition. Furthermore, these partitions can be ranked in terms of their likelihood to influence switching activity of a function if used during synthesis. It is thus desirable to partition the space into sets of minterms of similar probability. These partitions will be referred to as *classes*.

The definition of suitable classes follows from an analysis of the relationship between functionality and probability. Consider a circuit with a set of inputs  $I_p$ , where all inputs are

independent and have an onset probability  $p$ . Clearly, there are  ${}^j C_{|I_p|}$  minterms of probability  $p^j \cdot (1-p)^{|I_p|-j}$  in the Boolean space defined by the input variables. The distinct minterm probabilities correspond to the number of different ways of choosing  $j$  inputs to be on, and  $(|I_p| - j)$  inputs off for each  $\{j: 0 \leq j \leq |I_p|\}$ . For example, in a circuit with four inputs  $I_p = \{w, x, y, z\}$ , each with probability  $p$  of being high, there are  ${}^4 C_2 = 6$  minterms with probability  $p^2(1-p)^2$ , namely:  $\{\bar{w}\bar{x}yz, \bar{w}x\bar{y}z, \bar{w}xy\bar{z}, w\bar{x}\bar{y}z, w\bar{x}y\bar{z}, wx\bar{y}\bar{z}\}$ .

These sets of minterms form a set of classes  $\phi_{I_p}^0$ , the zero superscript indicating that approximating the minterm probability in each class with the class average is exact. By construction, this is a set of  $(|I_p| + 1)$  distinct classes. Now consider a circuit with a set of inputs  $I$  with onset probabilities  $\mathcal{P}$  where  $|\mathcal{P}| \leq |I|$ , and all inputs are independent. A set of exact-average classes which partition the  $|I|$  variable Boolean space is then given by the product of the exact-average classes for each  $I_p, p \in \mathcal{P}$ . i.e.

$$\phi_I^0 = \prod_{p \in \mathcal{P}} \phi_{I_p}^0 \quad (\text{EQ 3.8})$$

$|\phi_I^0|$  from EQ. 3.8 is the minimum cardinality set of exact average classes under the assumption of input independence made in this analysis. The number of exact average classes is thus  $O(\prod_{p \in \mathcal{P}} (|I_p| + 1))$ , which is generally impractically large even if  $|\mathcal{P}| \ll |I|$ . (For example, even if  $|\mathcal{P}| = 5$  and  $|I| = 30$ ,  $|\phi_I^0|$  could be as large as:  $(30/5 + 1)^5 \simeq 17$  thousand!). Furthermore, as the classes are formed from the *choose* operation, they bear similarities to XOR functions and are in some sense very disjoint in the Boolean space. In general, smooth functions which are subsets of the DC set are more suited for synthesis optimization. Consequently, if the intention is to use the classes to modify use of the DC set, it is necessary to deal with smoother functions. A class in  $\phi_I^0$  function can be smoothed by collapsing classes within a  $\phi_{I_p}^0$ . As was shown in the first example, this collapse can be made without significantly affecting error of approximation by the average

class minterm probability. A set of Boolean classes for the entire space with minimal minterm-probability variance can be built from considering error/class count trade-off curves for each  $I_p$ .

**Definition 3.5 (class approximation error):** Let  $B^n$  be the Boolean space described by the Boolean variable set  $x = \{x_1, \dots, x_n\}$ ,  $x_i \in B^1 \forall i = 0, 1, \dots, n$ , and class  $C$  a set of minterms of  $B^n$ ,  $C \subseteq B^n$ . The class approximation error  $\varepsilon(C)$  is defined as the maximum error in approximating the minterm probabilities  $p(m)$ ,  $m \in C$  by the average minterm probability  $p(C)/|C|$  and is given by:

$$\varepsilon(C) = \sum_{m \in C} \left| \left( p(m) - \frac{p(C)}{|C|} \right) \right|. \quad (\text{EQ 3.9})$$

For a specific  $p \in \mathcal{P}$ , let  $\varphi_{I_p} = \{C_1, C_2, \dots\}$  be a set of classes chosen from the union of classes in  $\varphi_{I_p}^0 = \{C_1^0, C_2^0, \dots\}$ , where the elements of  $\varphi_{I_p}^0$  are arranged in the increasing order of their average minterm probability, i.e.  $p(C_1^0)/|C_1^0| < p(C_2^0)/|C_2^0| < \dots$ . The class approximation error will be minimized if the subset of  $\varphi_{I_p}^0$  corresponding to any  $C_i \in \varphi_{I_p}$  is contiguous with respect to indices of  $\varphi_{I_p}^0$ . e.g.  $C_1 = C_1^0 \cup C_2^0$ ,  $C_2 = C_3^0 \cup C_4^0$  will have smaller error than  $C_1 = C_1^0 \cup C_3^0$ ,  $C_2 = C_2^0 \cup C_4^0$ . Generation of  $\varphi_{I_p}$  for minimal error then becomes equivalent to the optimal selection of a set of contiguous, mutually exclusive subsets of  $\varphi_{I_p}^0$ . A heuristic technique to do this is the selection of the subsets of  $\varphi_{I_p}^0$  in order to maximize the uniformity of the total probability contained in each of the final classes.

A plot of the total error (which is the sum of the class approximation error for each class,  $\sum_{C_i \in \varphi_{I_p}} \varepsilon(C_i)$ ) in  $\varphi_{I_p}$  against the total number of classes for this grouping strategy is shown in Fig. 3-3. For each curve,  $|I_p| = 10$  and  $p \in \{0.1, 0.2, 0.3, 0.4\}$ . Note that in each case, the number of classes can be significantly reduced with minimal impact on error. For small allowable error (<10%), this effect increases with decrease in  $p$ .

Now consider the set of classes described by the Boolean product of classes chosen for each

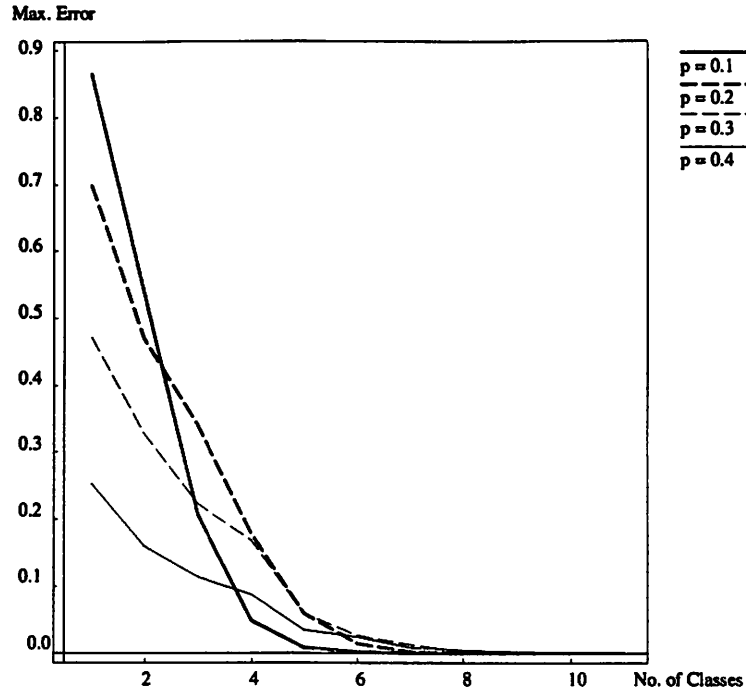


Figure 3-3 : Error vs. Number of Classes:  $|I_p| = 10$

input subset  $I_p$ . As each set of classes  $\phi_{I_p}$  covers the entire space with mutually exclusive sets, the overall set of classes formed from the product maintains this necessary property. Let  $\epsilon_p$  be the total error for the set of classes  $\phi_{I_p}$ . An upper bound on the error for the set of classes on  $I$  defined by the product is given by:

$$\epsilon = 1 - \prod_{p \in \mathcal{P}} (1 - \epsilon_p) \quad (\text{EQ 3.10})$$

The tightness of this bound is illustrated in Fig. 3-4. The data on this graph is generated from a series of statistical tests for input probabilities  $p \in \{0.1, 0.2, 0.3, 0.4\}$  with  $|I_p|$  chosen randomly from  $[2, 10]$  and  $|\phi_{I_p}|$  from  $[0, |I_p|]$ . It is clear that the error/class count trade-off curves generated for each  $I_p$  can be used to define an error/class count sensitivity for the entire class product  $\phi_I$ .

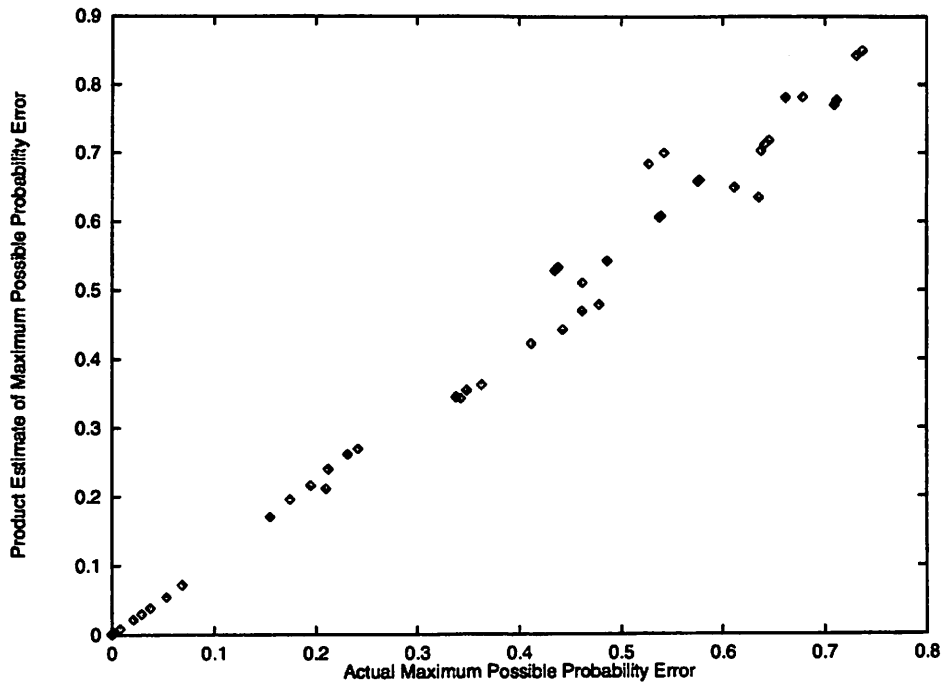


Figure 3-4 : Actual Global Error vs. Product of Errors Estimate from (EQ 3.10)

The practical implementation of this work first approximates the entire set of input probabilities (every input has an onset or offset probability in the range of 0 to 0.5) with a set  $\mathcal{P}$ ,  $|\mathcal{P}| = 5$ , chosen for minimum RMS error. The error curves are then generated for each  $p \in \mathcal{P}$ . The reduction in the number of classes achieved this way with less than a 10% error tolerance can be several orders of magnitude for large circuits.

In general, the number of classes remaining after this operation is still too many for practical use in synthesis. For technology independent synthesis, these classes are collapsed into two minterm classes to construct the set of power-sensitive minterms as defined in Section 3.3.4. For technology dependent synthesis, five minterm classes are used to guide the rewiring algorithm (Section 3.4.4). In this case, the grouping into the final five classes is based upon the similarity of average minterm probabilities in the original classes, and an attempt to equally partition the total

probability.

### 3.3.4. Power Sensitive Minterms

The flexibility provided by a don't care set in power optimization is proportional to the functional probability of the don't care set function. This is different from area and delay optimization, where the flexibility is proportional to the size or cardinality of the don't care set function. From the power optimization point of view, depending upon the minterm probability distribution, only a part of the don't care set may be very useful. The concept of power sensitive minterms identifies the part of the Boolean space which contains most of the minterm probability and is thus useful for power optimization. It turns out that power sensitive minterms constitute a very small fraction of the Boolean space for most of the real life minterm probability distributions, and is thus a very useful tool in separating the minterms important for power optimization from the don't care set without significantly effecting the size of the remaining don't care set (and thus the flexibility for area and delay optimization).

An example of the probability distribution of minterms within the Boolean space is shown in Fig. 3-5 for an ISCAS benchmark circuit with 41 input variables. The curves show the cumulative coverage of the Boolean space (y-axis) against the cumulative consumption of total probability (x-axis) in accumulating from the minterms with the largest probability to the minterms with the smallest. These curves are generated for three different input probability distributions - all inputs set to a unique onset probability 0.5, a gaussian distribution centered at 0.5 with  $\sigma = 0.1$ , and a uniform distribution between [0, 1]. Clearly, probability in Boolean space localizes as input probabilities become more evenly distributed on [0, 1]. This localization captures the fact that most of the minterm probability is concentrated in a small fraction of the Boolean space, and motivates the following definition of *power sensitive minterm* class.

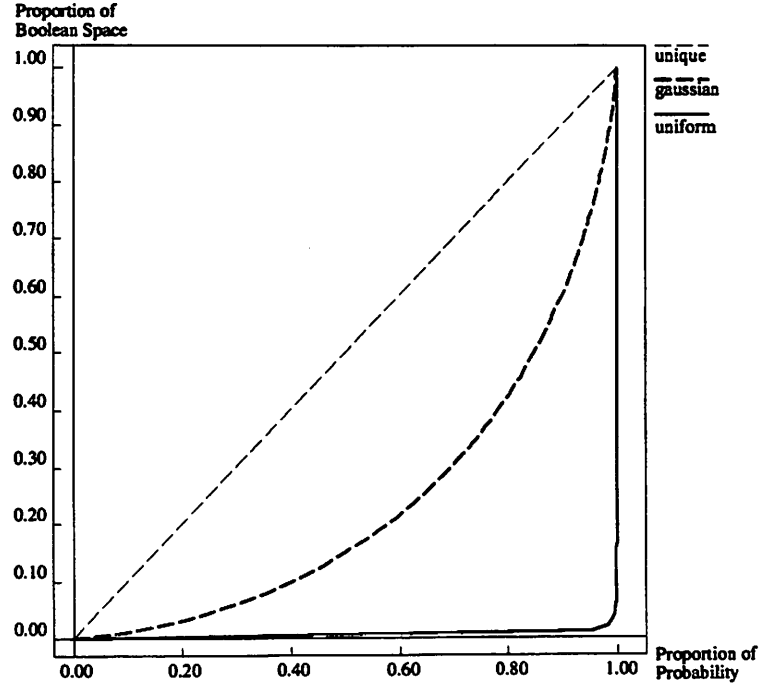


Figure 3-5 : Proportion of Boolean Space vs. Proportion of Total Probability

**Definition 3.6 (Power Sensitive Minterms):** Let  $B^n$  be the Boolean space described by the boolean variable set  $x = \{x_1, \dots, x_n\}$ ,  $x_i \in B^1 \forall i = 0, 1, \dots, n$ , with associated probability set  $\mathcal{P}$ . The power sensitive minterm class  $C_{PS}$  is the set defined by the relation  $C_{PS}: (\mathcal{R}^1 \times B^1)^n \rightarrow B^n$ ,

$$C_{PS} = \left\{ m \mid p(m) \geq \frac{1}{2^n}, m \in B^n \right\} \quad (\text{EQ 3.11})$$

The notion of *power sensitive minterms* attempts to capture the variance in the minterm probability distribution in the Boolean space and as such can be treated as a set of minterms with probability greater than some arbitrary threshold. This definition is motivated by the fact that for the particular case of all inputs to a network having a unique probability  $p$ , some interesting properties can be demonstrated for  $C_{PS}$ .



**Proposition 3.1 :** Let  $B^{|I|}$  be the Boolean space described by the set of inputs  $I$ , each with the same input onset probability  $p \neq 0.5$ . Then  $C_{PS} \leq |B^{|I|}|/2$ .

**Proof :** Without loss of generality, assume  $p > 0.5$ . For  $j \in [0, |I|]$ , there are  $|I|C_j$  minterms of probability:  $p^j \cdot (1-p)^{|I|-j}$ . Let  $S_j$  be the set of minterms with probability  $p^j \cdot (1-p)^{|I|-j}$  and  $S_{\{0, \dots, k\}} = \bigcup_{j=0}^k S_j$ . It follows from the symmetry of the function  $|I|C_j$  as  $j$  moves from 0 to  $|I|$  that  $|S_{\{0, \dots, \lfloor |I|/2 \rfloor\}}| \geq |B^{|I|}|/2$ . To complete the proof, we need only show that  $|S_{\{0, \dots, \lfloor |I|/2 \rfloor\}}| \subset \overline{C_{PS}}$ , i.e.  $\forall m \in S_{\{0, \dots, \lfloor |I|/2 \rfloor\}}, p(m) < 1/|B^{|I|}|$ .

Clearly,  $p^{j-1} \cdot (1-p)^{|I|-(j-1)} < p^j \cdot (1-p)^{|I|-j}$ , so it is sufficient to show that  $p^j \cdot (1-p)^{|I|-j} < 1/|B^{|I|}|$  for  $j = \lfloor |I|/2 \rfloor$ .

We may replace every instance of  $p$  with  $p = 0.5 + \epsilon$ ,  $\epsilon > 0$ . If  $|I|$  is even, this case is  $p^{|I|/2} \cdot (1-p)^{|I|/2} = ((0.5 + \epsilon)(0.5 - \epsilon))^{|I|/2} = (0.25 - \epsilon^2)^{|I|/2} < 1/|B^{|I|}|$ . If  $|I|$  is odd, this case is  $p^{|I|/2} \cdot (1-p)^{|I|/2} \cdot (1-p) < 1/|B^{|I|}|$ . ■

We conjecture that for any Boolean space  $B^n$  and associated input probability set  $\mathcal{P}$  it can be shown that greater than 50% of the probability is contained in  $C_{PS}$  and  $C_{PS} \leq \frac{|B^n|}{2}$ . In practice, the Boolean space is first split into a set of similar probability minterm classes using the techniques outlined in Section 3.3.3, and then  $C_{PS}$  is approximated as the union of all classes with *average* minterm probability  $\geq \frac{1}{|B^n|}$ . The point at which the classes are collapsed to form  $C_{PS}$  is equivalent to the point on the curve of Fig. 3-5 where  $\frac{dy}{dx} = 1$ .

### 3.3.5. Guided Logic Synthesis for Low Power

Two-level logic minimization is the core workhorse step in multi-level logic minimization. This step is applied to each complex node in the network (travelling from primary outputs to primary inputs) to optimize each node using the DC flexibility at each node. Section 3.2 described how the

switching activity at a node can be changed by making functional changes to the cover of the function implemented at the node. Now consider an incompletely specified function  $(f_n, DC_n)$  at a node  $n$  in a multi-level logic network, where  $f_n$  is the onset of the function,  $DC_n$  is the don't care set, and  $\overline{f_n \cup DC_n}$  is the offset. From EQ. 3.4, function probability is a monotonically increasing function of the function onset, i.e.  $f_1 \subseteq f_2 \Rightarrow p(f_1) \leq p(f_2)$ . Accordingly, since all possible covers  $f$  of  $f_n$  satisfy  $f_n - DC_n \subseteq f \subseteq f_n + DC_n$ , we have  $p(f_n - DC_n) \leq p(f) \leq p(f_n + DC_n)$ . Thus,  $[p(f_n - DC_n), p(f_n + DC_n)]$  give the bounds on the possible probabilities that can be synthesized at node  $n$ . From Fig. 3-2, the lowest power implementation of the function at node  $n$  can be contained by implementing  $f_n - DC_n$  or  $f_n + DC_n$  depending upon which of  $p(f_n - DC_n) \cdot (1 - p(f_n - DC_n))$  or  $p(f_n + DC_n) \cdot (1 - p(f_n + DC_n))$  is lower. The main drawback of this strategy is that while it yields the lowest power implementation at the node, all DC flexibility is used up and there is no scope of using it for area or delay optimization. In the following we describe an algorithm to which uses only a part of the DC flexibility for power optimization. The proposed algorithm used the notion of power sensitive minterms to identify a small part of the DC which can give most of the power savings possible from the strategy above, without significantly affecting the flexibility available for area or delay optimization. The two-level minimization is then performed by a two phase process which first addresses reduction in switching activity using this subset of the DC set, and the reduction in area.

The concept of separating the optimization phase for reduction in activity and area (therefore, capacitance) is illustrated in Fig. 3-6. Fig. 3-6(a) indicates two classes which partition the Boolean space, with  $C_{PS}$  representing the power sensitive minterms which cover most of the probability space and a small fraction of the Boolean space with a few high probability minterms, and  $\overline{C_{PS}}$ , encompassing most of the Boolean space and a small fraction of the probability space with many small probability minterms. Fig. 3-6(b) shows an incompletely specified node function with onset  $f_n$ , and a don't care set denoted by  $D_n$ . Without loss of generality, assume that the

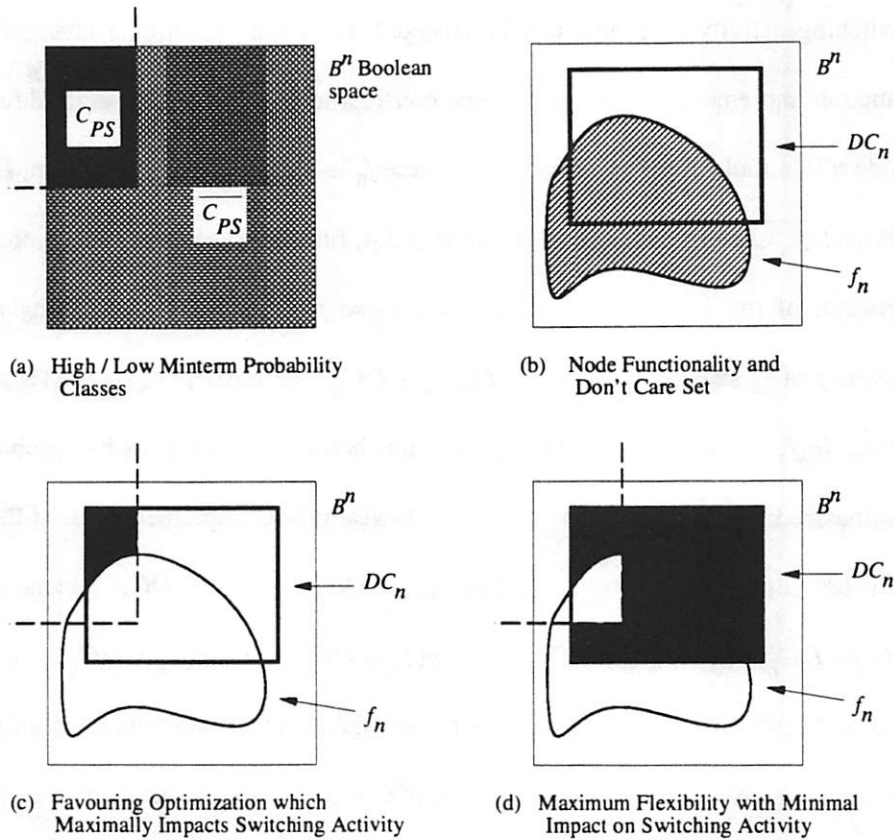


Figure 3-6 : Power / Area Optimization Phases,  $p(f_n) > 0.5$

function probability  $p(f)$  of the current cover  $f$  of  $f_n$  is greater than 0.5. To reduce switching activity, we want to increase the onset probability to push it towards 1. Thus from Fig. 3-2, it is desirable during the optimization of  $f_n$  to absorb elements of  $C_{PS}$  with a functional expansion into the set described by:  $DC_n \cdot \bar{f} \cdot C_{PS}$ . This region is represented by the black shaded area of Fig. 3-6(c). Although an expansion will benefit switching activity, this is usually a very small subset of the DC set and unlikely to provide sufficient area optimality. Optimization for area requires the flexibility of a large subset of the DC set. However, it is important to avoid functional contraction within the set  $C_{PS}$  as even a small excursion in that direction could sharply increase switching activity. (i.e.  $p(f_n)$  would move closer to 0.5.) The set providing maximum flexibility without allowing small

```

for  $i = 1$  to  $N$ 
  /* Low Area Flexibility, High Expected Activity Change */
  if  $p(f_{n_i}) > 0.5$ 
     $f'_{n_i} = \mathcal{A}(f_{n_i}, DC_{n_i} \cdot \overline{f_{n_i}} \cdot C_{PS})$ 
  else
     $f'_{n_i} = \mathcal{A}(f_{n_i}, DC_{n_i} \cdot f_{n_i} \cdot C_{PS})$ 
  if ( $P(f'_{n_i}) < P(f_{n_i})$ )
     $f_{n_i} = f'_{n_i}$ 
  /* High Area Flexibility, Low Expected Activity Change */
  if  $p(f_{n_i}) > 0.5$ 
     $f'_{n_i} = \mathcal{A}(f_{n_i}, DC_{n_i} \cdot (\overline{f_{n_i}} + \overline{C_{PS}}))$ 
  else
     $f'_{n_i} = \mathcal{A}(f_{n_i}, DC_{n_i} \cdot (f_{n_i} + \overline{C_{PS}}))$ 
  if ( $\text{Area}(f'_{n_i}) < \text{Area}(f_{n_i})$ )
     $f_{n_i} = f'_{n_i}$ 

```

Figure 3-7 : Pseudo-code for Favoring Power Reduction during Area Optimization

functional changes to have a strongly detrimental influence upon switching activity is:  $DC_n \cdot (\overline{f} + \overline{C_{PS}})$ . This region is represented by the black shaded area of Fig. 3-6(d). A similar strategy can be applied in the case when  $p(f) < 0.5$ , to guide the optimization in to contracting to further reduce  $p(f)$  and push it towards 0.

A pseudo-code form of the ideas presented above is provided in Fig. 3-7. In that presentation,  $N$  is the number of nodes in the network, and  $\{n_i; 1 < i < N\}$  are the set of nodes indexed in reverse topological order from the primary outputs to the primary inputs. Assume that the class  $C_{PS}$  and the DC for the network have already been produced. For node  $n_i$  with function  $f_{n_i}$  and local DC set  $DC_{n_i}$ , let  $\mathcal{A}(f_{n_i}, DC_{n_i})$  be the functional manipulation performed on  $f_{n_i}$

during area optimization within the DC set.  $\text{Area}()$  is the area estimate (usually based on the literal count of the factored form representation of the argument function) used in the traditional two-level minimizer, and  $P(f)$  is the function power dissipation of a function  $f$  as defined in Section 3.2.

Note that the final area optimization step deals with the function  $f_{n_i}$  after any activity optimization. The restriction of the DC set in that step therefore prevents the area optimization from undoing any critical expansion/contraction already achieved within the set  $C_{PS}$ .

### 3.3.6. Results

The algorithm outlined in the previous section was implemented inside the SIS logic synthesis package to guide the node minimization phase during multi-level logic optimization. The resulting programs for power-sensitive node minimization -  $\text{power\_simplify}()$  and  $\text{power\_full\_simplify}()$  are counterparts of the area optimization routines  $\text{simplify}()$  and  $\text{full\_simplify}()$  of SIS. For benchmarking purposes, we replaced the occurrences of  $\text{simplify}()$  and  $\text{full\_simplify}()$  in *script.rugged* with our  $\text{power\_simplify}()$  and  $\text{power\_full\_simplify}()$  command to obtain *script.power*. A subset of circuits from the MCNC benchmark set was used to obtain the experimental results. All circuits were mapped using *msu.genlib*. Power estimation and switching activity computation was performed using the symbolic simulation method of [Gho92] using a zero-delay model. All input probabilities were chosen from a uniform distribution in the range [0, 1]. All experiments were run on a DEC-station ALPHA with a 160Mb memory.

The results comparing the area and power reduction obtained by optimization via *script.rugged* and *script.power* are presented in Table 3-1. The names of the benchmark circuits tested is given in Column 1, column 2 contains the number of literals in the factored form. In Column 3 and 4 respectively we present the percentage of total probability contained in the set  $C_{PS}$  and the proportion of the Boolean space it encompasses. Column 5 shows the area of the circuit

Circuit	#lit	$p(C_{PS})$	$\frac{ C_{PS} }{ B^n }$	Area			Power			% change	
				init	rug	our	init	rug	our	Area	Power
cm138a	39	82%	9%	480	472	456	67.6	49.2	43.0	-3.4	-12.6
pm1	65	94%	7%	1000	792	760	198.4	90.3	86.9	-4.0	-3.8
sct	194	93%	5%	2128	1336	1456	540.1	248.6	244.5	9.0	-1.6
f51m	207	87%	9%	2272	1840	1752	474.6	356.3	336.0	-4.8	-5.7
lal	252	94%	4%	3856	1616	1320	630.0	312.1	258.5	-18.3	-17.2
9symml	328	87%	9%	3552	3464	2040	912.7	820.4	437.4	-41.1	-46.7
tnt2	420	94%	4%	4624	3240	3296	1085.4	458.1	548.9	1.7	19.8
alu2	635	92%	15%	7264	5960	3952	1249.7	931.6	640.6	-33.7	-31.2
vda	1870	94%	6%	19672	10400	10160	3097.1	955.0	901.2	-2.3	-5.6
Total		91%	7%	60960	31528	27600	8643.1	4538.4	3813.8	-12.5	-16.0

Table 3-1 : Comparison between *script.rugged* and *script.power*

prior to optimization. Columns 6 and 7 present the results after network optimization using *script.rugged* and *script.power* respectively. Similarly, Column 8 shows the power dissipation of the unoptimized circuit and Columns 9 and 10 present the power dissipation results after optimization using *script.rugged* and *script.power* respectively. Columns 11 and 12 show the percentage change in area and power results by using *script.power* instead of *script.rugged*.

The results demonstrate that in most cases *script.power* yields a circuit with lower power dissipation than *script.rugged*. There is no significant trade-off in terms of circuit area. In fact, on the whole, *script.power* performs better in area optimization as well. Note that *script.power* does not always give a better result than *script.rugged*. This is to be expected since our approach attempts to favorably bias (from the power perspective) the network optimization process at the node minimization level, but cannot guarantee that this will always translate in a lower power network. At the same time, since the area optimization flexibility is not strongly affected by our algorithm,

we do expect that in most cases our algorithm will yield a lower power network without any area penalty. This assertion is validated by the experimental results (a total of 16% reduction in power and a 12.5% reduction in area over the set of benchmark circuits, averaging 8.0% reduction in power and 7.5% reduction in area for each individual case).

The positive and negative aspects of this approach both arise from a the same overall property of the results - this algorithm reduces both area and power. The computational cost of this gain is a doubling of the runtime of *simplify()* and *full\_simplify()*, two key steps in area optimization.

Although this trade-off may be regarded as acceptable, it is clear that the use of power-sensitive DC sets does not achieve a controllable de-correlation of reduction in activity and reduction in area. This follows from the attempt to reduce switching activity at the output of a complex node through high-level area optimization. In general, the output activity of complex node does not dominate the power consumption of its internal structure. It would therefore be necessary to bias the extraction of internal structure (e.g. [Ima95]) to reduce the area-dominant effect of power reduction.

Ongoing work in this area is examining the application of activity-sensitive classes to extraction of internal node structure as well as guided functional alteration through engineering change. In the latter case, power reduction should be achieved with zero change in area as only existing gates are used in the optimization.

### **3.3.7. Conclusions**

This work addresses the problem of technology independent power optimization. The Boolean space spanned by the primary input vectors of a combinational function may contain a large variance in minterm probabilities. By partitioning the DC set into regions strongly and weakly

influential upon switching activity we exploited this variance to bias area/delay optimization towards reduced power dissipation.

The main contributions of this work are:

- The notion of power sensitive minterms to capture the large variance in minterm probabilities. Power sensitive minterms were defined such that it can be proved under some restrictions that they cover more than half of the probability space while containing less than half of the minterms in Boolean space. In practice, the variance is much more marked and over a set of MCNC benchmark it was found that the power sensitive minterms covered 91% of the probability space while containing just 7% of the minterm space.
- An efficient technique to identify the power sensitive minterms and construct the power sensitive minterm class given the probability distribution of the primary inputs spanning the Boolean space. The exact construction is exponential in nature. An approximate technique was outlined and error bounds derived to control the approximation error.
- An algorithm to apply the concept of power sensitive minterms to technology independent logic synthesis for low power. The proposed approach uses the intersection of power sensitive minterms and the DC set to control the node simplification process during the optimization of multi-level logic networks in the desired direction. The algorithm is aimed at reducing power without any significant trade-off in other optimization criteria like area or delay. Experimental results show that this approach can yield a lower power implementation of the circuit compared to the standard SIS area optimizer without paying any area penalty.



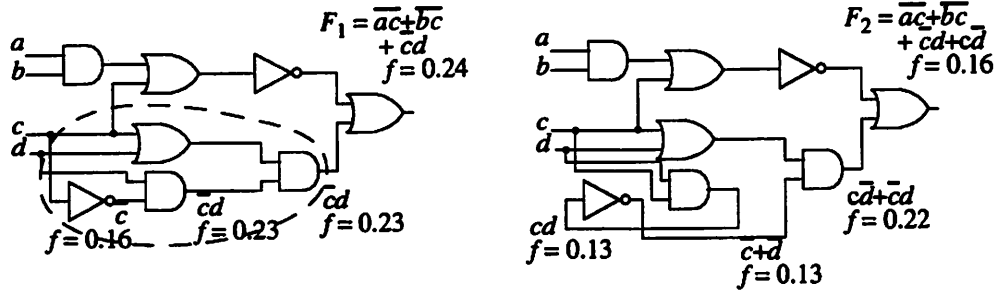


Figure 3-8 : Reducing switching activity via rewiring: an example.  $F = \overline{ac} + \overline{bc} + \overline{cd}$ ,  $DC = \overline{cd}$ ,  $p(a) = 0.4$ ,  $p(b) = 0.6$ ,  $p(c) = 0.2$ ,  $p(d) = 0.8$ . transition probability  $f = p(\text{high})(1-p(\text{high}))$ , where  $p(\text{high})$  for a variable is the probability of the variable being high in any clock period [She92]

### 3.4. Technology Dependent Logic Synthesis for Low Power

#### 3.4.1. Introduction

Technology dependent logic synthesis optimizes a circuit with a given target technology in mind. Optimizations at this stage consist of algorithms for technology mapping, in which a technology independent Boolean network with arbitrary complex logic nodes is mapped to a network whose nodes are synthesizable gates, as well as algorithms for refining a technology mapped circuit. Power savings are possible at both these stages. In this work, we address the problem of reducing power dissipation of a technology mapped circuit. Since a elements of a mapped circuit have a direct correspondence to the underlying transistor level implementation, power savings obtained by the techniques presented here are very reliable.

We formulate the problem of power optimization of a technology mapped circuit as a variant of the engineering change (EC) problem. EC is a class of synthesis algorithms which aim at implementing a new function in a circuit by making minimal changes to the existing circuit. Rewiring is a subset of this set of algorithms which preserves the original gates of the network. Fig.

3-8 shows an example where rewiring a region inside the circuit reduces the switching activity. Note that rewiring an internal region affects the switching activity of the transitive fanout (TFO) of the region as well. In this example, rewiring takes advantage of the external DC to yield a function with lower switching activity.

We will show that rewiring can be used to reduce power of hot regions by providing it an incompletely specified target function for the region such that any rewiring compatible with the target function is expected to dissipate less power. This rewiring does not change the circuit area or gate capacitance and does not increase the circuit delay under the unit delay model. The selection of hot regions, EC formulation and the construction of this incompletely specified function for EC are the major contributions of this work.

A very useful by-product of the EC formulation of the problem is that this work can also be applied to the problem of hot spot reduction in integrated circuits. Hot-spots are regions of circuit which have particularly high power reduction. Apart from contributing to reliability problems due to high power dissipation, hot-spots create performance and/or functionality hazards as well. This is because the high power dissipation in hot-spots causes a significant amount of heat to be generated locally. This changes the operating temperature of the circuit elements in the hot spot region with respect to the rest of the circuit. The change in operating temperature changes the performance characteristics of this region, causing signal delays to change on some paths, slowing down the circuit or even causing a failure. There is no easy engineering solution to this problem as the circuit cannot necessarily be made to compute correctly by just running it slowly, and it is not feasible to put heat sinks in the middle of the circuit specially for the hot-spots. Since such a problem is detected only after running detailed simulations at the end of the design process, it is very expensive to make major modifications in the design to fix it. In this context, an EC solution which takes as an input a technology mapped circuit, identifies hot regions and rewires them to

reduce their power dissipation is perfectly suited to this problem. In fact, the proposed solution to the technology dependent power optimization can be viewed as successively applying hot spot reduction to a circuit until no further improvements are possible.

An overview of the proposed algorithm is presented in Section 3.4.2, after a quick definition of CMOS power consumption in. The notion of sensitivity and flexibility in the context of logic level power optimization is described in Section 3.4.4. Section 3.4.5 outlines the engineering change solution based on these notions. Section 3.4.6 presents the results of applying this theory to standard benchmarks, and Section 3.4.7 concludes with a brief summary of this work.

### **3.4.2. Previous Work**

The power dissipation of a CMOS logic circuit depends on the gate capacitances and node switching activity. Low-power synthesis algorithms at the technology independent level ([Bah95][Ima95][Lin93a][She92]) use techniques like reducing the functional support from high-activity inputs, or guiding sub-expression extraction through simple high-level power approximations. However, in these works the effect on capacitance during network restructuring is difficult to predict. At the technology dependent stage of synthesis these effects are more predictable. At this stage, there have been works which try to minimize the total switching activity using a procedure similar to Huffman's algorithms ([Tsu93a]) as well as methods which use area-delay trade-off curves ([Pra93][Tiw93]).

Rewiring, which is employed after technology mapping, allows us to make small functional changes in the circuit which do not change the circuit structure and thus the capacitance. This gives us more control over power dissipation and yields results which can be expected to translate in power gains even at the transistor and layout level. [Roh96] and [Bah96] optimize a mapped circuit for power by evaluating candidate circuits generated by a set of structural transformations using

ATPG methods and learning-based redundancy addition/removal respectively. Both of these methods can change area and delay properties of the circuit.

### **3.4.3. Algorithm Overview**

In this work, we are interested in reducing the power dissipation of a circuit by making very small functional changes. An engineering change-based formulation is used to achieve this (in Section 3.4.5.1 we discuss the rationale behind this in detail). Specifically, we use rewiring to make functional changes so that gate capacitances are unchanged and the power minimization problem is reduced to minimizing switching activities. Since switching activity at a node depends on the node functional probability, we are now interested in making beneficial changes in the functional probability. This is implemented as follows:

1. We partition the Boolean space into minterm classes such that each class has minterms of similar probabilities, which can each be approximated by a single value per class. This allows us to relate probability of a function in each class to its onset size in each class (Section 3.3.3).
2. We use a global sensitivity based formulation to relate the expected change in overall power dissipation to a change in the onset size of an internal node (Section 3.4.4.1).
3. We make some observations about the distribution of the number of possible logic functions in a Boolean space as a function of their onset sizes. This is used as a measure of the flexibility available at a node in making functional changes and to predict the expected onset size when an arbitrary functional change is made at an internal node. (Section 3.4.4.2).
4. (2) and (3) above are used to select nodes with a lot flexibility in making functional changes and for whom a small functional change can greatly affect the overall power dissipation.

5. For each selected node, (1) and (3) are used to select the classes of the ODC minterms where the predicted onset size when making a functional change results in a beneficial probability. This is used to construct the incompletely specified target function for engineering change (Section 3.4.5.3).
6. Rewiring is used to select the minimum wire implementation which satisfies the target function. This guarantees that the circuit area and gate capacitance do not change and the critical path does not increase under the unit delay model (Section 3.4.5.4).

#### **3.4.4. Sensitivity and Flexibility in Synthesis for Low-Power**

A change in functionality at a node in a network may influence both the power dissipation local to that node as well as at nodes throughout the TFO. It was established in our work of [Len96b] that there exists a simple and highly accurate numerical technique for computing the expected change in functionality throughout the TFO of a node when functional manipulation is performed within the bounds of the ODC. This was related to power under the assumption that all inputs had a probability of 0.5 of switching during any clock period. In general, however, this assumption which allows functional *size* (i.e. minterm count) to be directly related to switching probability is invalid. The extension of this work in [Len96b] provided a simple mechanism for generalizing the theory under the assumption that the Boolean space could be sectioned into sets of like-probability minterms. In [Len96b] we outlined an efficient mechanism for determining such sets. The work presented here will unify these ideas to establish the concept of power-sensitivity for nodes within a network with arbitrary input probabilities incorporating both local and global considerations.

To establish the concept of functional sensitivity, it is important to briefly outline the relevant work of [Len96b] and [Len96a] (Section 3.3.3, Section 3.4.4.1). We will first outline the technique for establishing similar minterm-probability classes. We then describe the technique for

estimating the change in functionality throughout the TFO of a node when it is resynthesized and we will relate this to the estimated change in activity.

In Section 3.4.4.2 we present a new technique to measure the synthesis flexibility at a node by making some observations about the expected size of functions under an arbitrary functional change.

### 3.4.4.1. Transitive Fanout Sensitivity

Consider a node  $n$  with intermediate node inputs  $\{n_1, n_2, n_3, \dots\}$  where the functionality of node  $n_1$  is to change from  $f_{n_1}$  to  $f'_{n_1}$ . Let  $A_{n_1}$  be the set of minterms added to  $f_{n_1}$ ,  $R_{n_1}$  be the set of minterms removed. A minterm is added to the functionality at node  $n$  if it is added to (removed from) the functionality at node  $n_1$  and it is contained within the positive (negative) sensitivity of node  $n$  to  $n_1$ ,  $S_n^{pos}(n_1)$  ( $S_n^{neg}(n_1)$ ). The *expected* change in function at node  $n$  is therefore given by the probability of overlap of the sensitivity and added/removed minterm sets at  $n_1$ . As the change in onset at node  $n_1$  can only occur within the ODC at that node, and the added (removed) minterms must lie within  $\overline{f_{n_1}}$  ( $f_{n_1}$ ), the following formulation may be derived:

$$E(|A_n|) = p(S_n^{pos}(n_1) | (\overline{f_{n_1}} \cdot ODC_{n_1})) \cdot |A_{n_1}| + p(S_n^{neg}(n_1) | (f_{n_1} \cdot ODC_{n_1})) \cdot |R_{n_1}| \quad (\text{EQ 3.12})$$

Similarly, for the expected number of minterms removed:

$$E(|R_n|) = p(S_n^{neg}(n_1) | (\overline{f_{n_1}} \cdot ODC_{n_1})) \cdot |A_{n_1}| + p(S_n^{pos}(n_1) | (f_{n_1} \cdot ODC_{n_1})) \cdot |R_{n_1}| \quad (\text{EQ 3.13})$$

where  $p(A|B) = |A \cap B|/|B|$ .

This formulation can be propagated throughout the TFO to estimate the expected size of

the change in functionality at every node influenced by the change at  $n_1$ . (The technique for handling reconvergent fanout is outlined in [Len96a], and omitted here.) Although this is only a prediction of *average* change in functionality without an estimate of standard deviation, extensive practical experimentation has shown that the variance of actual size of functional change versus average estimate is extremely small. This follows from the fact that the vast majority of possible functions which may arise during a synthesis step cover near half the total number of minterms available within the ODC flexibility. This is discussed in greater detail in Section 3.4.4.

When all minterms (input variable assignments) have the same probability of occurrence, the relationship between the change in switching activity and expected size of the change in functionality is trivial. However, this is not the case when minterm probabilities are distributed. Although the prediction of change in power assuming that all minterms are equally likely would correctly average out when sensitivity performance is examined over a large number of circuits, in general the standard deviation would become much too large to guarantee the usefulness of the method for any specific instance. To make the technique more viable, it has to be able to be tuned to the specific input probability distribution. This is achieved by splitting the Boolean space into a set of classes,  $\varphi$ , of like-probability minterms. The technique outlined above is then performed inside each class, resulting in the following sums:

$$E(p(A_n)) = \sum_{C_i \in \varphi} E(|A_{n_i}|) \cdot \frac{P(C_i)}{|C_i|} \quad (\text{EQ 3.14})$$

$$E(p(R_n)) = \sum_{C_i \in \varphi} E(|R_{n_i}|) \cdot \frac{P(C_i)}{|C_i|} \quad (\text{EQ 3.15})$$

As the computation of expected change in power given the local expected change is a completely numerical procedure once the  $p(S_n^{neg}(n_1) | (\overline{f_{n_1}} \cdot ODC_{n_1} \cdot C_i))$  etc. terms have been

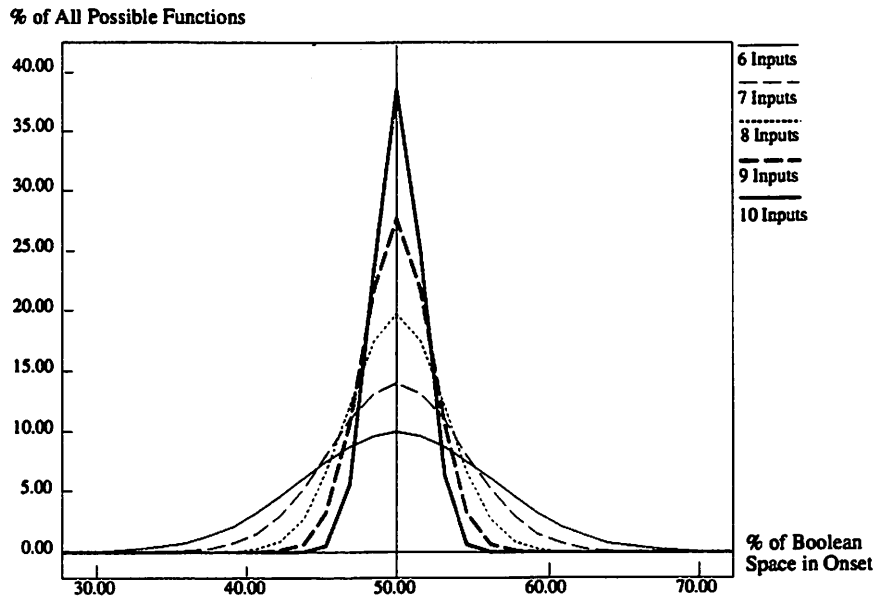


Figure 3-9 : Function size probability profile

computed for each node and class in a single pass over the network, a reasonable number of classes can be handled without the computational penalty dominating the synthesis routine.

### 3.4.4.2. Flexibility

We have generalized the TFO sensitivity algorithm for circuits with arbitrary input switching probabilities through the assumption of being able to partition the Boolean space into several classes containing similar-probability minterms. Further, we have now shown how those classes can be efficiently computed. All that remains in the computation of a global power sensitivity is a prediction of the size of the expected functional change during resynthesis.

To establish this, we assume that any function within the bounds of the provided flexibility is equally likely. This allows a functional size probability profile to be trivially established as there exists  ${}^N C_k$  ways of forming a  $k$ -minterm size function in an  $N$ -minterm Boolean space. Example profiles are shown in Fig. 3-9 for several input variable counts. (All profiles are binned into 64 x-



axis data points for comparative purposes.)

As the number of variables in the functional support increases, the *centralizing* effect becomes more dramatic. Even for input counts {4, 6, 8, 10} the normalized standard deviation of the function count profile is {0.125, 0.062, 0.031, 0.016} respectively. This would decrease exponentially for real-life functions with more inputs. It is this property which allows an average prediction technique for establishing sensitivity to work incredibly well for estimating the global effect of specific synthesis cases.

This property also lets use define the functional flexibility for an arbitrary synthesis step. The *flexibility* is defined as the *expected functional change*. Due to the above centralization property, any function resulting from an arbitrary functional change is *expected* to have half of the don't care flexibility in its onset. For example, an ODC containing  $N$  minterm,  $m$  of which are originally in the function on-set, it is given by:  $\left| \frac{N}{2} - m \right|$ . The application of this to synthesis is detailed in Section 3.4.5.

### **3.4.5. Power Optimization Through Engineering Change**

#### **3.4.5.1. Engineering Change Based Formulation**

Given a logic network that has been already been synthesized (possibly for low power), we want to reduce the power dissipation by making incremental changes. In [Len96b] a technique was explored whereby regional synthesis was guided to make small changes in node functionality throughout a multi-level network such that the total power was reduced. However, the additional circuitry that might be required to implement those functional changes could possibly offset the power reduction achieved. This adverse effect can be reduced by using an engineering change approach which aims at modifying the circuit in a *minimal* way to realize the new specification at the node.

The problem of minimal modification of the circuit to reduce power differs a little from the engineering change problem in that the target function is not a hard constraint. In general, we just want to achieve an arbitrary expansion/contraction of the onset within the ODC set such that the power dissipation is decreased. This behavior can be captured by an incompletely specified target function, i.e. a target function such that any onset change which meets the target function specification is beneficial. Computing a function which includes all possible such changes is exponentially complex. However, the techniques described in Section 3.4.4 may be used to compute a target function for which any arbitrary change which meets the function specification is *expected* to be beneficial.

In the following, we outline an algorithm to compute this target function and a rewiring based approach to solve the engineering change problem. The choice of a rewiring approach is particularly appropriate in the context of power optimization as rewiring a region of the circuit does not affect existing gate capacitance.

#### **3.4.5.2. Rewiring based Power Optimization**

The proposed rewiring-based algorithm consists of two phases: identifying the redesign region and applying rewiring to reduce power dissipation. The first phase involves determining the circuit nodes which have the highest flexibility and power sensitivity. All nodes are ranked based on the sensitivity of network power to expected change in functionality and the nodes contributing the largest beneficial changes are selected for optimization. In a mapped circuit, optimizing just one node does not yield significant power gains. In order to provide a sufficiently large input for the optimizer to manipulate, we identify a region for rewiring with the flexible node at its root.

### 3.4.5.3. Choosing Rewiring Region and the Target Function based on Flexibility/ Sensitivity Considerations

For each node, we estimate the *suitability* of the node for power optimization by computing the expected change in its power dissipation under an arbitrary optimization step as follows:

We first partition the boolean space in to  $k$  classes using the techniques presented in Section 3.3.3, such that all minterm probabilities in the  $i$ th class can be approximated by one average probability value, say  $p_i$ . For a node implementing a functionality  $f$ , let the node cover before optimization be  $f'$  and after optimization be  $f''$ . Let the essential minterms of  $f$  be represented by the function  $f_{essential}$  and the DC by  $f_{DC}$ . The probability of  $f''$ ,  $p(f'')$  is then  $p(f'') = \sum_{i=0}^k p(f_i'')$ , where  $f_i''$  is the projection of  $f''$  over the  $i$ th class. Since minterms in each class are represented by a single probability,  $p(f'')$  is given by  $p(f'') = \sum_{i=0}^k p_i \cdot |f_i''|$ .

Based on the current onset probability, we then decide if it is beneficial to expand or contract the onset. Since from (EQ 3.7)  $P(f) \propto p(f) \cdot (1 - p(f))$ , if the current probability of the node cover  $f'$ ,  $p(f') > 0.5$ , it is desirable to expand the onset so that the node power dissipation decreases, and to contract if  $p(f') < 0.5$ .

In each class  $i$ , the final cover  $f_i''$  must include  $f_{essential_i}$ , and may include some subset of  $f_{DC_i}$ . Thus, for each class we have two possibilities: keep the original  $f_i'$  within the class, or optimize  $f_i'$  using  $f_{DC_i}$ . We compute the expected value of the onset size of  $f_i''$ ,  $E(|f_i''|)$ , under the conditions of allowing or not allowing the DC flexibility to influence functional specification within the class. The configuration most compatible with the objective of decreasing local and TFO activity is then chosen. For example, in the case where DC flexibility is permitted,  $E(|f_i''|)$  can be computed as  $E(|f_i''|) = |f_{essential_i}| + E(|f_{DC_i}''|)$ .  $E(|f_{DC_i}''|)$  is the expected value of the onset size of a function selected from a set of minterms with a cardinality of  $|f_{DC_i}|$  via some optimization

step. From the discussion in Section 3.4.4.2,  $E(|f_{DC_i}''|) = 0.5 \cdot |f_{DC_i}|$  giving  $E(|f_i''|) = |f_{essential_i}| + 0.5 \cdot |f_{DC_i}|$ . This expected change in functionality is combined with the TFO sensitivity work of Section 3.4.4.1 to predict a global power change.

This expected global power dissipation change estimate is computed for all possible combinations of allowing/not allowing the use of DC in each class and the best combination of DC classes is then chosen for each node. The minterm classes  $i$  for which  $f_{DC_i}$  is used in the best flexibility combination are referred to as *useful DC classes*. The nodes with the highest potential for reducing power are then chosen for optimization and the useful DC classes for each are used to define the incompletely specified target function to be implemented at the node.

#### **3.4.5.4. The Rewiring for Low Power Algorithm**

We propose an algorithm based on rewiring to redesign the hot, flexible network region identified by the techniques of the previous section. The redesign algorithm presented here modifies existing circuitry by reconnecting gates in the region with all the gate types and gate counts unchanged. As a result, the power optimization process does not change the total gate capacitance of the circuit, which means that any reduction in switching activity is made without a capacitance trade-off.

The proposed rewiring algorithm is an adaptation of [Kuk94] which formulates the redesign problem as a Boolean-constraint problem and gives an algorithm based on BDDs to generate all possible assignments of gate connections which satisfy the specified target functionality.

The rewiring algorithm assigns a Boolean connection variable for each ordered pair of gate outputs and inputs in the region. The value of the variable is 1 if there exists a connection between this pair in the redesigned circuit and 0 otherwise. It then builds a characteristic function for each gate to capture all possible functionalities that can be implemented at that gate using all possible

combinations of the connection variables.

Formally, lets  $c_{v_i, v_k}$  ( $i = 1, \dots, k-1$ ) be the Boolean variable for the connection from the output of gate  $i$  to an input of a 2-input AND gate  $k$ . Let  $v_i$  be the Boolean variable corresponding to the output of the gate  $i$ . Then this characteristic function  $\chi_{2\text{-AND}}$  of the AND gate  $k$  is

$$\chi_{2\text{-AND}} = \text{LTE2}(c_{v_1, v_k}, \dots, c_{v_{k-1}, v_k}) \cdot \left( v_k \equiv \prod_{i=1}^k (c_{v_i, v_k} \Rightarrow v_i) \right) \quad (\text{EQ 3.16})$$

where  $\text{LTE2}()$  is the Boolean function which evaluates to one iff  $\leq 2$  of its arguments are one. The  $\text{LTE2}$  thus selects two of the all possible connection variables feeding into the inputs of gate  $k$ , and  $\chi_{2\text{-AND}}$  captures all possible connection assignments which implement the AND function at  $v_k$ .

The complete set of possible functions which can be implemented by the region can then be computed given by ANDing the characteristic functions of all gates. That is, the characteristic function of the circuit after reconnection  $\chi$ , is given by

$$\chi(\mathbf{v}, \mathbf{c}) = \prod_{g \in G} \chi_g(\mathbf{v}, \mathbf{c}) \quad (\text{EQ 3.17})$$

where  $G$  is the set of gates in the original network and  $\chi_g$  is the characteristic function for gate  $g$ .  $\mathbf{v}$  is the set of all circuit variables and  $\mathbf{c}$  is the set of all connection variables.

After smoothing out all the internal variables in  $\mathbf{v}$  which are associated with the intermediate nodes in the region, we are left with  $\chi(\mathbf{i}, \mathbf{o}, \mathbf{c})$ , a function of the primary input ( $\mathbf{i}$ ) and output variables ( $\mathbf{o}$ ) and all the connection variables. We then compare this with the characteristic function  $\chi_s(\mathbf{i}, \mathbf{o})$ , of the target specifications. Note that in our case, this is target function an incompletely specified function with the don't cares flexibility provided by the useful DC classes computed in 3.4.5.3.. The condition on the connection variables  $\mathbf{c}$  then is that the input-output

behavior of the reconnected circuit implies the input-output behavior of the specification. i.e.,

$$\chi_{redesign}(c) = \zeta_{i,o}(\chi(i, o, c) \Rightarrow \chi_s(i, o)) \quad (\text{EQ 3.18})$$

The consensus operator above extracts all 0-1 assignments to  $c$  such that  $\chi(i, o, c) \Rightarrow \chi_s(i, o)$  is a tautology. Note that each minterm of the characteristic function represents a 0-1 assignment of the connection variables which will satisfy the target functionality. For more details of the algorithm, refer to [Kuk94].

The target function for the redesign region is computed using  $f_{essential}$  and the useful DC classes as determined by the algorithm in the previous section. The union of these useful DC classes gives a subset of the node DC set which is expected to be beneficial for power optimization and this incompletely specified function is used to direct the rewiring algorithm. The output of the rewiring algorithm is a set of minterms of connection variables, each of which satisfy the incompletely specified target function. While each of these represents a different wiring scheme with a different power dissipation, based on the reasoning of Section 3.4.4.2, the target function is constructed such that the power dissipation is expected to reduce when we arbitrarily pick a single wiring assignment. Without any loss of generality, we pick the assignment which implies the minimum numbers of connection wires. This minimizes the power dissipation due to wiring capacitances, and under a unit delay model guarantees that the critical path length for the region does not increase.

### 3.4.6. Results

The algorithms above have been implemented inside the SIS logic synthesis package. A subset of circuits from the MCNC and ISCAS\_89 benchmark set were used to obtain the experimental results. All circuits were mapped using *msu.genlib*. Without any loss of functional generality, the rewiring algorithm uses a reduced form of this library due to the limitations of the current rewiring implementation. Power estimation and switching activity computation was performed using the

symbolic simulation method of [Gho92] using a zero-delay model. All input probabilities were chosen from a uniform distribution over [0,1]

The results from the rewiring algorithm are presented in Table 3-2. The results obtained by first mapping the circuit, then optimizing it for area using script.rugged and then applying our rewiring algorithm to it. The runtimes in Column 2 are on a DEC Alpha machine. Column 3 contains the power dissipations of the mapped, area optimized circuit input to our algorithm and Column 4 has the power dissipation of the rewired circuit resulting from our algorithm. Column 5 contains the ratio of these two. Overall, a 4% reduction in power was achieved, with reductions of up to 13% in some cases. Note that rewiring can never increase the gate count so in essence there is no trade-off in this power reduction. In fact, there was in general a reduction in literal count due to the fact that during the rewiring procedure, not all gates pins are necessarily re-used. We expect the results to further improve as we extend our benchmarking to large circuits, since these circuits would have more flexibility for redesign.

### **3.4.7. Conclusions**

We have addressed the problem of power optimization at the technology-dependent level. The main contributions of this work are:

- An engineering change (EC) based formulation of the problem of resynthesis for low power which allows the adaptation of EC algorithms to power minimization. This formulation is made possible by two critical observations about the numerical properties of minterms and functions in the Boolean space. We use these to construct a target function for EC, such that any implementation satisfying the target function is expected to reduce power.
- A unified framework combining the theory of [Len95] and [Len96b] to allow global power sensitivities to be defined for networks with arbitrary input probability distributions. To achieve

Circuit	run time (sec)	power ( $\mu$ W)		power ratio
		before EC	after EC	
traffic_cl	0.1	26.1	25.7	0.98
b1	0.1	16.6	16.1	0.97
mux_cl	0.6	96.0	93.6	0.98
cm82	0.6	83.6	79.2	0.95
cm151	3.2	94.7	88.6	0.94
parity	4.2	225.5	197.4	0.88
cm42	0.3	59.4	57.3	0.96
cm138	0.3	49.4	47.3	0.96
c17	0.1	25.8	25.8	1.00
tcon	0.7	112.2	110.9	0.99
decod	0.7	67.2	65.1	0.97
cmb	1.2	174.3	167.9	0.96
cm163	1.3	157.2	149.6	0.95
pcl	2.8	168.5	160.1	0.95
mux	2.0	94.7	190.4	0.98
cm162	0.7	104.0	101.7	0.98
cm150	1.9	176.4	171.3	0.97
cm85	1.1	95.3	88.2	0.93
z4ml	1.2	105.4	100.9	0.96
cu	1.2	131.6	126.1	0.96
pcler8	0.7	229.5	218.8	0.95
cc	2.3	161.6	151.4	0.94
unreg	36.1	328.4	314.3	0.96
count	10.9	414.3	412.1	0.99
my_adder	38.2	649.0	622.5	0.96
comp	87.0	437.8	398.7	0.94
cht	33.6	218.1	190.3	0.87
c8	10.5	488.3	467.4	0.96
lal	4.9	310.4	295.7	0.95
b9	66.5	364.9	350.2	0.96
cordic	2.4	195.1	185.6	0.95
frg1	12.0	457.5	444.1	0.97
ttt2	14.2	362.3	348.9	0.96
term1	36.7	548.9	527.8	0.96
Total				0.96

Table 3-2 : Power reduction by rewiring on some area-optimized MCNC/ISCAS-89 benchmarks



this, we partitioned the Boolean space into similar-minterm-probability classes within which the existing functional sensitivity theory applies. Our global sensitivity based approach takes into account the effect of TFO activity change on the circuit power dissipation when making local functional changes.

- New theory for estimating the expected change in onset size of a logic function during synthesis given a specific flexibility. This was based on our observations about the small variance in the distribution of the number of possible logic functions in the Boolean space as a function of their onset size.
- A technique using these theories to select hot nodes with a lot of flexibility in making functional changes and for whom a small functional change can greatly affect the overall power dissipation; and a technique to construct the target function for rewiring.
- A rewiring approach to EC which achieves the above target function while guaranteeing that the circuit area and gate capacitance do not change and the critical path does not increase under the unit delay model. Experimental results show an average of power reduction of 4% on a set of MCNC benchmark circuits, with reductions of up to 13% in some cases.

### **3.5. Summary**

Power optimization algorithms were presented at the technology independent and dependent phases of logic synthesis. These algorithms exploit the fact that the Boolean space spanned by the primary input vectors of a combinational function may contain a large variance in minterm probabilities. An efficient technique was outlined for exploring the Boolean space to identify minterms highly appropriate for influencing switching activity. The notion of power sensitive minterms was introduced to capture the variance in minterm probability distribution. The overlap of power sensitive minterms, which cover a large fraction of the probability space but a very small fraction

of the Boolean space, with the DC set was used to bias technology independent area/delay optimization towards reducing switching activity. Experimental results show that power can be reduced by as much as 46% and by 16% on the average without any area penalty.

A technology dependent power optimization technique was proposed which formulates the problem of hot spot reduction as a variant of the engineering change (EC) problem. A technique was presented for determining the sensitivity of circuit power dissipation to functional changes considering both local and global effects. This sensitivity was combined with a measure of synthesis flexibility to identify hot regions in the circuit which have a lot of flexibility in making functional changes and for whom a small functional change can greatly affect the overall power dissipation. An incompletely specified target function was constructed for the hot region such that any implementation satisfying it is expected to reduce power. A rewiring algorithm was used to solve the resulting EC problem without affecting circuit area, gate capacitance or delay under the unit delay model. Experimental results on a set of MCNC benchmark circuits show that the proposed approach can give up to 13% reduction in power dissipation with an average reduction of 4%.

---

# 4 Logic Synthesis for Pass Transistor Circuits

---

## 4.1. Introduction

Static CMOS has long been the design style of choice for most IC designers. However, switching capacitances in a static CMOS circuit can be fairly large. With the shrinking feature sizes and increasing transistor counts on chips, the push for higher speed and lower power makes it necessary to look for alternative design styles which can offer better performance characteristics to static CMOS. These include pass-transistor-based logic families, domino-like dynamic logic styles etc. [Rab96]

Among these, pass transistor logic (PTL) circuits offer great promise. Compared to domino circuits, they are less susceptible to crosstalk problems, which is a major issue in deep sub-micron technology. Several case studies have shown that PTL can implement most functions with fewer transistors than static CMOS [Cha92a][Yan90]. This reduces the overall capacitance, resulting in faster switching times and lower power. It was reported in [Yan90] that a complementary PTL multiplier was twice as fast as conventional CMOS due to lower input capacitance and higher logic functionality. At a supply voltage of 4V, PTL designs typically consume 30% less power than static CMOS designs [Cha92a]. To illustrate this point, consider a function  $F = \bar{A} + B\bar{C}$ . Fig. 4-1(a) shows one implementation of this function in PTL and Fig. 4-1(b) shows the corresponding static CMOS implementation. Clearly, the PTL design style can yield a circuit which can be much more compact

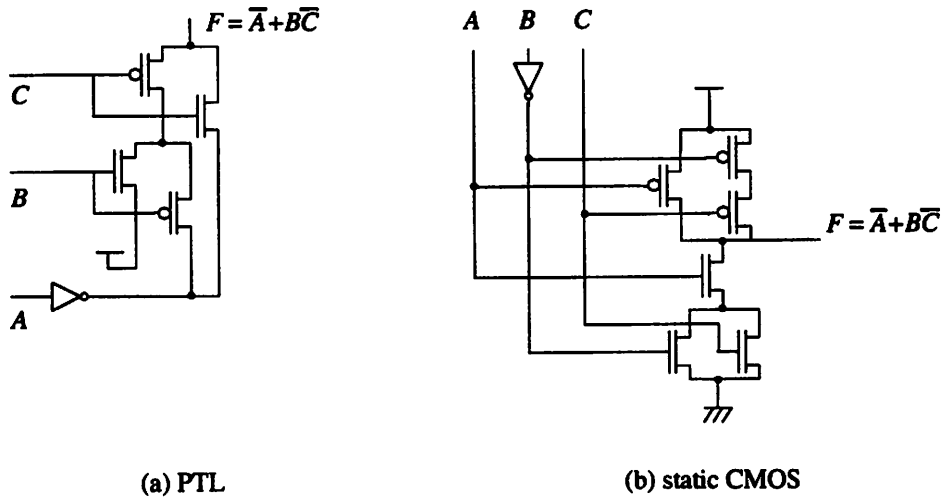


Figure 4-1 : Comparing pass transistor and static CMOS implementations of an example function  $F = \bar{A} + \bar{B}\bar{C}$ .

than static CMOS. It was reported in [Yan96] that the PTL yielded a 32% improvement in area, 29% improvement in delay and a 47% improvement in power over a static CMOS OR/NAND-based implementation of this function.

The circuit in Fig. 4-1(b) can in fact, also be interpreted as a PTL circuit. The only difference between PTL and static CMOS is that in static CMOS, unlike PTL, all paths from  $V_{dd}$  to the output are connected via pMOS (the pull-up network) and paths from output to ground are connected via nMOS (pull-down network). Thus, static CMOS can be viewed as restricted case of PTL. These restrictions make the task of synthesizing safe, large static CMOS circuits easier, but reduce the potential of circuit optimization. Thus, given a methodology to synthesize safe, large circuits, PTL can be more attractive than static CMOS.

The lack of such a methodology is why the use of pass transistors in industry circuits has been very limited. While there have been several attempts in this area ([Ala91][Jae96][Nev94][Rad85][Sak90][Sal95][Sas95][Sha95][Yan96]), limitations of some of

which are discussed later in the chapter, there are no algorithms which can be used to design safe, large PTL circuits. Thus, while designers can manually design very efficient small PTL circuits as in Fig. 4-1, a satisfactory solution to automatic synthesis of circuits realizing the expected benefits of PTL does not exist.

This work addresses this void with a decomposed BDD-based approach which exploits some of the strengths of PTL logic and is scalable in that it can be used to obtain compact, multi-stage transistor-level circuits for large, arbitrary designs.

A comprehensive synthesis flow is outlined for PTL design starting from an unoptimized logic level netlist, all the way up to generating a spice netlist. For this, a suitable logic level abstraction based on decomposed BDDs is proposed which allows performing logic level optimizations similar to the traditional multi-level network based synthesis flow for static CMOS. This representation takes advantage of the correspondence between PTL circuits and BDDs without suffering from the drawbacks imposed by properties of monolithic BDDs. A straightforward mapping exists from this logic level abstraction to a transistor-level PTL netlist which preserves all the interconnection information. This makes possible optimizations with a direct impact on area, delay and power of the final circuit implementation. A set of heuristical algorithms to synthesize PTL circuits optimized for area, delay and power, which are key to the proposed synthesis flow, are presented. Experimental results on ISCAS benchmark circuits show that the proposed technique yields PTL circuits with substantial improvements over conventional static CMOS designs. To the best of our knowledge this is the first time PTL circuits have been synthesized for the entire ISCAS benchmark sets.

The rest of the chapter is organized as follows: Section 4.2, argues why a BDD-based approach is suitable for PTL circuit synthesis and reviews the shortcomings of monolithic BDD-based approaches. Section 4.3 motivates decomposed BDDs as a suitable logic level abstraction for

PTL synthesis. Section 4.4 compares the proposed decomposed BDD-based synthesis flow and the traditional approach for static CMOS. Section 4.5 presents decomposition techniques to obtain PTL circuits optimized for area, delay and power. Section 4.6 presents the experimental results. Section 4.7 outlines issues for future research, Section 4.8 reviews the contributions of this work and Section 4.9 concludes with a summary of this work.

## **4.2. Pass Transistor Logic Networks and BDDs**

One of the main strengths of static CMOS designs is that they are guaranteed to not have a steady-state sneak path connecting a node to both power supply and ground at the same time under some input combination. From Section 4.1, PTL admits more general circuit structures than static CMOS. However, it suffers from the drawback that there is no guarantee on the absence of sneak paths in the circuit. Hence, special care needs to be taken to ensure that the circuit is sneak path-free. For example, the PTL circuit in Fig. 4-2 requires only three transistors to implement the example function from Fig. 4-1. However, this circuit has a sneak path as shown, forcing the output to be connected to both ground and power supply at the same time when  $A=1$ ,  $B=0$ ,  $C=0$ . There is therefore the need of a methodology to synthesize PTL circuits which ensure the absence of such sneak paths, or guarantee that if they exist in the logic they cannot be exercised (e.g. via the use of input don't cares).

The basic unit in PTL is a MOS transistor which is used as a switch. When the control signal at the MOS gate is enabled, the input (drain/source) is connected to the output (source/drain). The output is in a high impedance state when the control signal is disabled. This switching characteristic of the MOS makes it very easy to implement a multiplexer in PTL as a wired OR of transistors.

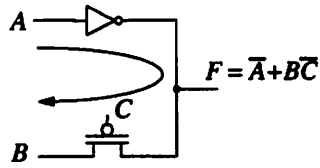


Figure 4-2: A PTL circuit with a sneak path

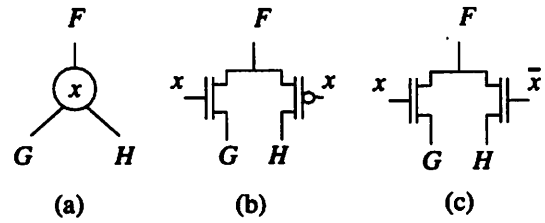


Figure 4-3: Implementing a BDD node in PTL

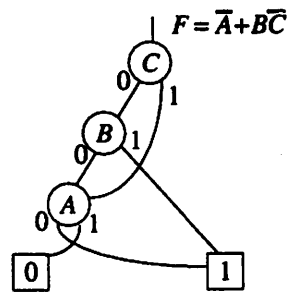
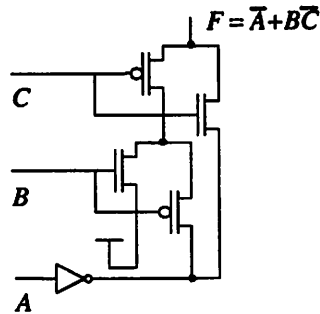


Figure 4-4: Comparing pass transistor implementations of the example function of Fig. 4-1 with its BDD

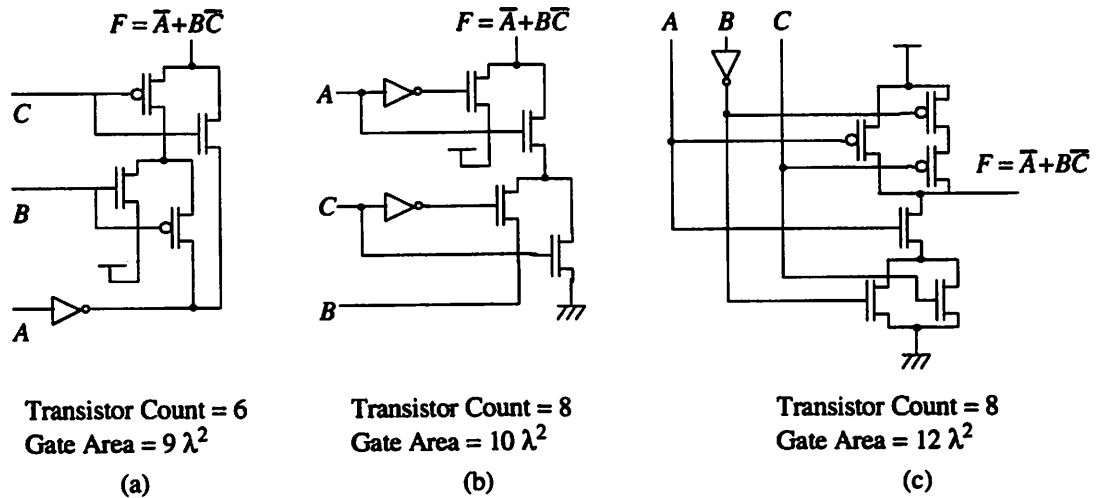


Figure 4-5: Alternative BDD-based implementation of the example function from Fig. 4-1

A 2-input multiplexer implements the same functionality as a BDD node, with the BDD node variable corresponding to the control signal of the multiplexer and the outgoing and incoming branches of the BDD node corresponding to the inputs and output of the multiplexer respectively. Fig. 4-3 shows two different ways of implementing a BDD node using two MOS transistors.

Thus, the BDD representation of the target function can be very easily mapped to a multiplexer network, which in turn can be implemented compactly using pass transistors. This provides a way to construct efficient PTL circuits [Sak90]. In fact, the PTL implementation in Fig. 4-1(a) corresponds to the BDD of  $F$ , as shown in Fig. 4-4.

The main advantage of such a BDD-based approach is that it always gives correct, sneak-path-free circuits, since at a time, only one path connecting the ground/power supply to the output is active. Using the two different implementations of a BDD node from Fig. 4-3 yields the two circuits shown in Fig. 4-5(a) and Fig. 4-5(b), both smaller than the static CMOS implementation in Fig. 4-5(c). Note that the nMOS-only implementation in Fig. 4-5(b) uses more transistors than the implementation in Fig. 4-5(a) because it needs *signal* and *signal* for each BDD node. However, it is quite competitive in terms of gate area. This is due to the fact that to obtain a similar current drive, pMOS has to be twice as big as nMOS in terms of the gate size (a minimum size pMOS has dimensions  $3\lambda \times \lambda$  while a minimum size nMOS is  $1.5\lambda \times \lambda$ ). This results in higher active gate area per transistor in the case of static CMOS and a pMOS/nMOS PTL. Also, in a pMOS/nMOS PTL, a pMOS can be in a path propagating a "1" and an nMOS can be in a path propagating "0", resulting in output levels of  $V_t$  and  $V_{dd}-V_t$  for "0" and "1" respectively. In comparison, in the nMOS-only case, the voltage is  $V_{dd}-V_t$  for output "1", and 0V for output "0" since nMOS are good conductors of "0".

This has three advantages:



- Each nMOS is at a better operating point when propagating “0” and has a higher drive, resulting in a faster circuit.
- The output has a better noise margin, which can be particularly important if it is driving MOS gates (of buffers or subsequent stages).
- Apart from the savings in active gate area, the smaller size of nMOS also means a lower gate capacitance. This results in a lower switching capacitance for the circuit making it faster and also reducing its power dissipation.

In fact, for large circuits, it was found empirically that the overhead of generating  $\overline{\text{signal}}$  was quite small (in most cases, particularly in case of large circuits, *signal* was required in the circuit anyway as *A* is in Fig. 4-5(a)), and the gate area savings and performance gains more than offset this. For this reason, this work uses the nMOS-only implementation of Fig. 4-3(c) in synthesizing transistor-level circuits.

While a BDD-based PTL network can be quite compact, a naive BDD-based methodology for implementing PTL circuits suffers from the drawback that for many functions of practical interest, the size of a BDD representing the function can be exponential in the number of inputs. Also, a circuit generated from a monolithic BDD can have long chains of transistors corresponding to long paths from the root to the 0/1 terminals for the BDD. This is equivalent to implementing a single-stage static CMOS circuit and can make the circuit very slow.

A technique for generating PTL circuits in which buffers are inserted in the monolithic BDD to solve the speed problem is given in [Yan96]. However, this approach still suffers from the BDD size problem. A multi-level pass transistor logic is introduced in [Sas95], which tries to maximize the logic shared between different parts of the circuit by looking at the structure of a monolithic BDD. Using a monolithic BDD as the starting point and modifying its structure has two

disadvantages: first, the approach will not be viable for large circuits with exponentially sized BDDs (e.g. a multiplier circuit). Secondly, even when a monolithic BDD can be built, the resulting circuit is highly sub-optimal in area because the optimizations are based on the topology of the BDD and not the logic implemented from it, thereby restricting the sharing to sub-graphs found in the original monolithic BDD.

### **4.3. PTL Networks and Decomposed BDDs**

We propose a synthesis approach which does not construct monolithic BDDs for the circuit at all. The common problem of the previous works outlined in Section 4.2 is that they try to improve a monolithic BDD-based solution. The proposed approach is truly multi-stage in that it always works with a multi-level representation of the PTL circuit which is similar to the traditional multi-level network for static CMOS. For such a flow, decomposed BDDs are proposed as a suitable logic level abstraction of the circuit which exploits the correspondence between PTL circuits and BDDs without suffering from the drawbacks imposed by properties of monolithic BDDs (e.g., canonicity, which may be desirable when using BDDs as a logic level data representation but is unnecessary for circuit generation).

The growth in BDD size can be controlled by introducing new, intermediate variables during the construction of the BDD itself. These intermediate variables are called decomposition points and the resulting set of BDDs (BDDs of the decomposition points, and the BDD of the target function in terms of the primary inputs and decomposition points) is called a decomposed BDD [Jai96]. An example of a decomposed BDD is shown in Fig. 4-6<sup>1</sup>. Note that the output of a

---

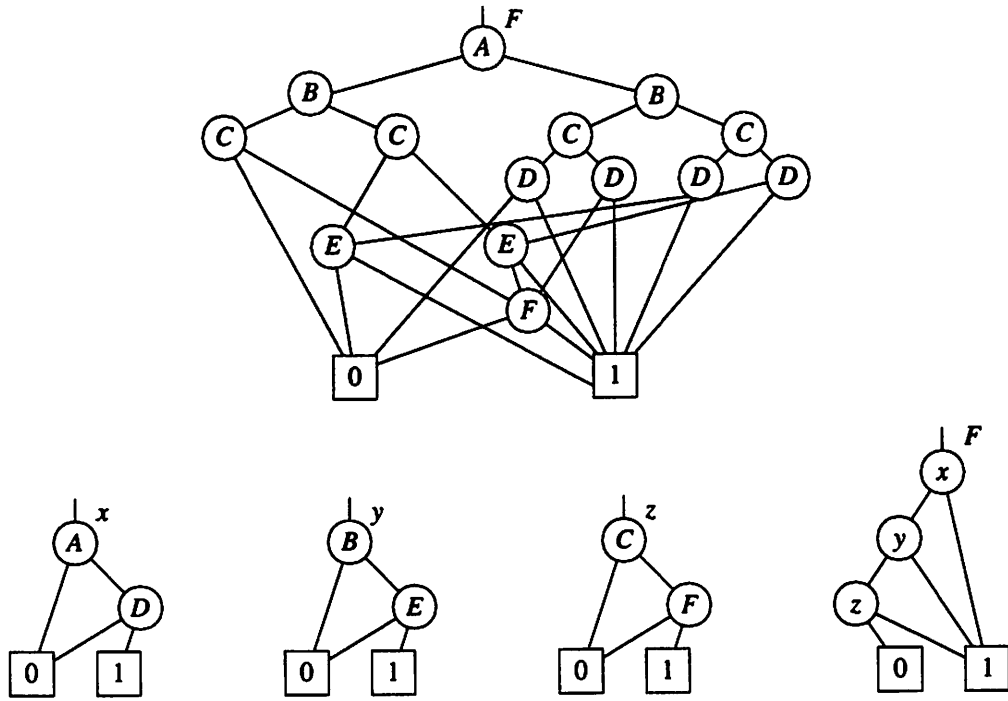
1. Although a more efficient ordering for this monolithic BDD exists [Bry86], for the given ordering this case serves to illustrate the potential BDD size reduction due to decomposition.

decomposition point BDD can be a node variable for the BDDs of subsequently introduced decomposition points or the target function. From Section 4.2, this corresponds to the output of a decomposition point driving MOS gates in the circuits of subsequent decomposition points or the target function. The resulting circuit is then a multi-stage circuit with cells in any given stage being driven by the primary inputs and the outputs of preceding stages.

The intuition behind the savings in BDD size due to decomposition is as follows: in general, when constructing the graph of a function  $F = G_1 \langle \text{op} \rangle G_2$ , the size of  $F$ ,  $|F|$ , is  $O(|G_1||G_2|)$ , where  $|G_1|$  and  $|G_2|$  are the sizes of the input graphs. By introducing decomposition points for  $G_1$  and  $G_2$ , the size of the decomposed BDD is reduced to  $O(|G_1|+|G_2|)$ . Thus, decomposition can be very useful when there is a memory explosion due to a difficult BDD manipulation during BDD construction. The trade-off here is that while monolithic ROBDDs are canonical for a given ordering, a decomposed BDD is not, since a BDD for a given function can be decomposed in many ways. This however does not pose a problem in PTL synthesis case, since the aim is to generate PTL circuits and not manipulate BDD as a data structure.

Note that this approach is orthogonal to the approach of [Yan96], in that decomposed BDDs can be used to obtain a compact BDD representation of the circuit. Each individual BDD can then be optimized by the techniques presented in this work and then mapped to a transistor-level circuit with appropriate buffering using [Yan96]. Similarly, optimization algorithms for area, delay and power presented here can be applied to BDDs generated using [Sas95] as well. Section 4.5.2, provides some more arguments on why, from a delay perspective for large circuits, a decomposed BDD approach is better than a monolithic BDD-based approach combined with buffer insertion.

The idea of introducing intermediate variables to control the size of BDDs has previously been used in [Jai96][McG95] for unrelated problems. In these approaches decomposition was used



$$F = AD + BE + CF \quad x = AD, y = BE, z = CF$$

Figure 4-6: Comparing monolithic and decomposed ROBDDs

in a different context. In [Jai96] decomposition was used to reduce the intermediate memory requirements during BDD construction and in [McG95] it was used for cycle-based simulation. In this work, decomposition is applied to construct a compact, decomposed BDD representation of the target logic function which can be directly mapped to a PTL network. The objective then is to develop decomposition techniques such that the PTL network corresponding to the resulting BDD is optimized for the desired objectives (e.g. area, speed, power).

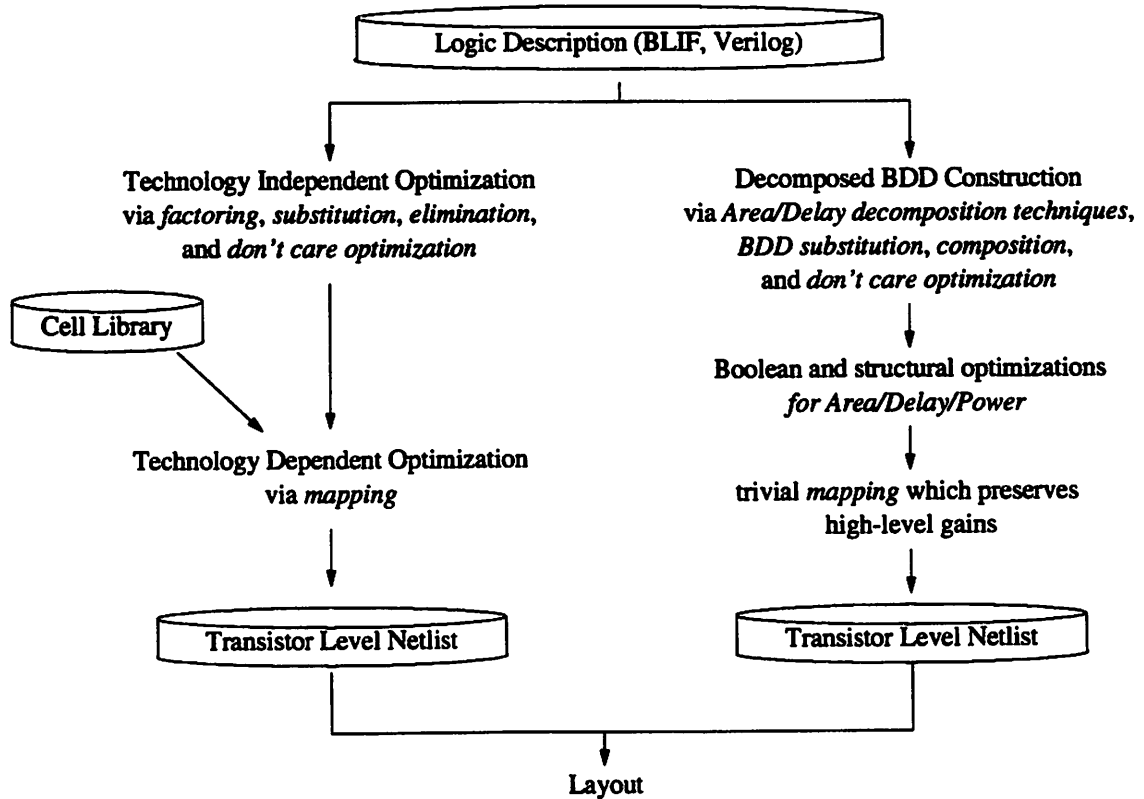


Figure 4-7: The traditional static CMOS synthesis flow vs. the proposed decomposed BDD synthesis flow

#### 4.4. A Synthesis Flow for PTL Design

Apart from proposing a decomposed BDD-based representation for PTL synthesis, a major contribution of this work is a comprehensive synthesis flow for PTL design.

Fig. 4-7 shows the key steps of the traditional multi-level network based synthesis flow for static CMOS. We propose an analogous synthesis flow where a decomposed BDD is used to represent a circuit similar to the multi-level network in the traditional flow and each decomposition point BDD is manipulated similar to a complex node in the multi-level network.

A big advantage of the BDD-based PTL network design is that the one-to-one mapping

between the BDD and the PTL network makes the technology mapping problem very straightforward. As a result, we can perform circuit level optimizations by manipulating the BDD. The fact that mapping preserves the circuit structure allows us to make high-level changes which can have significant impact on area, power and performance, but for which gains made at the high level hold at the circuit level as well. This addresses a big problem with the existing multi-level network based synthesis flow where technology independent optimizations are becoming increasingly irrelevant with respect to the final performance of the transistor-level design because the technology mapping does not preserve the structure. This is particularly important in the context of deep sub-micron designs, where logic level optimizations need to be driven by physical issues which depend on the circuit structure and topology.

The *factoring* operation of the conventional flow aims at extracting common sub-expressions out of a function description. This is similar to selecting good decomposition points in the proposed flow. *Substitution* is similar to using a decomposition point as a BDD variable in the construction of the BDDs of subsequent decomposition points and the target function. *Elimination* is similar to composition operation on decomposition point BDDs, where a decomposition point BDD is composed into the BDDs of the rest of the circuit and the BDD node variable corresponding to the decomposition point eliminated if there is an overall saving in BDD nodes. Design optimization using don't cares can be employed in the proposed flow in a fashion very similar to the conventional flow. This is described in more detail in Section 4.7.1.

Apart from above operations which are analogous to optimization steps in the conventional synthesis flow, the decomposed BDD-based approach allows optimization of circuits in several ways which have no equivalent in the conventional multi-level network based synthesis flow. These are outlined in Section 4.5.

between the BDD and the PTL network makes the technology mapping problem very straightforward. As a result, we can perform circuit level optimizations by manipulating the BDD. The fact that mapping preserves the circuit structure allows us to make high-level changes which can have significant impact on area, power and performance, but for which gains made at the high level hold at the circuit level as well. This addresses a big problem with the existing multi-level network based synthesis flow where technology independent optimizations are becoming increasingly irrelevant with respect to the final performance of the transistor-level design because the technology mapping does not preserve the structure. This is particularly important in the context of deep sub-micron designs, where logic level optimizations need to be driven by physical issues which depend on the circuit structure and topology.

The *factoring* operation of the conventional flow aims at extracting common sub-expressions out of a function description. This is similar to selecting good decomposition points in the proposed flow. *Substitution* is similar to using a decomposition point as a BDD variable in the construction of the BDDs of subsequent decomposition points and the target function. *Elimination* is similar to composition operation on decomposition point BDDs, where a decomposition point BDD is composed into the BDDs of the rest of the circuit and the BDD node variable corresponding to the decomposition point eliminated if there is an overall saving in BDD nodes. Design optimization using don't cares can be employed in the proposed flow in a fashion very similar to the conventional flow. This is described in more detail in Section 4.7.1.

Apart from above operations which are analogous to optimization steps in the conventional synthesis flow, the decomposed BDD-based approach allows optimization of circuits in several ways which have no equivalent in the conventional multi-level network based synthesis flow. These are outlined in Section 4.5.

## 4.5. Decomposition Techniques for BDD-based PTL Networks

### 4.5.1. Area Minimization

Since each node of a BDD corresponds to a PTL multiplexer cell, minimizing the area of the final circuit implementation is the same as minimizing the size of the decomposed BDD representation.

A simple, greedy heuristic is employed to control the size of the decomposed BDD by monitoring the BDD size while it is constructed. This is similar to [Jai96]. When building the BDD depth-first from inputs to outputs, a decomposition point is introduced whenever the BDD size increases by a disproportionate amount. This attempts to avoid difficult BDD manipulations. A decomposition point is also introduced when an individual BDD grows beyond a threshold value. This ensures that none of the individual BDDs in the decomposed representation exceeds the threshold. This is particularly important in the PTL context since both resistance and capacitance increase linearly with the number of transistor in series. Thus, a very deep BDD can result in a slow circuit.

Due to the local, greedy nature of the proposed heuristic, it is possible that the introduction of a decomposition point prevents Boolean simplification in the target function BDD. To discover some of these simplifications the decomposition points are composed back into the target function

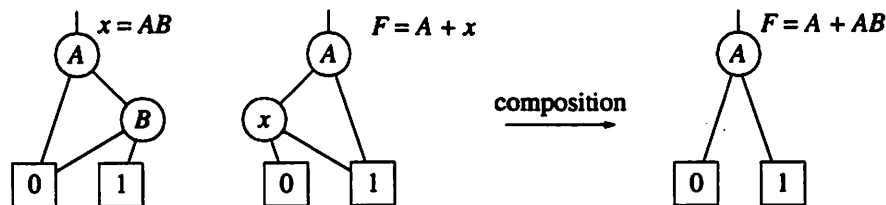


Figure 4-8 : BDD size reduction due to Boolean simplification via composition



BDD as long as the overall BDD size reduces. An example of BDD size reduction by composition due to Boolean simplification is shown in Fig. 4-8. Since the amount of reduction is dependent on the order of composition, several different orderings are experimented with to determine a good choice.

Complementary edges can be used to reduce the size of the BDDs even further. A complementary edge introduces an inverter in the circuit, saves at least one BDD node and in the best case reduces the BDD size by half [Ake78]. Hence the net transistor count can only decrease. Also, these inverters provide the added benefit of restoring the signal to the rail values, offsetting any signal degradation due to its passage through a long pass transistor chain. Additionally, the output of decomposition points are buffered if they are connected to MOS gates of a subsequent stage.

Further, when synthesizing PTL networks from a decomposed BDD, a global variable ordering for all BDDs is not required. This provides an additional flexibility for reducing the size of each BDD by reordering them independently.

#### **4.5.2. Performance**

In a monolithic BDD implementation, the critical path cannot be longer than the number of input variables  $n$  and can be as low as  $\log n$ . Decomposition introduces extra control variables whose critical paths can be in series with the critical path of the primary outputs' BDD. Note that the critical path length of the decomposition point BDD is bounded by the number of its variables, which can be more than  $n$  if the decomposition point is expressed in terms of other decomposition points. The critical path of the decomposed BDD is then bounded by  $\max$  {critical paths of decomposition point BDDs, length of the longest path in the primary output BDDs}. However, while the monolithic BDD has a smaller upper bound compared to decomposed BDDs,

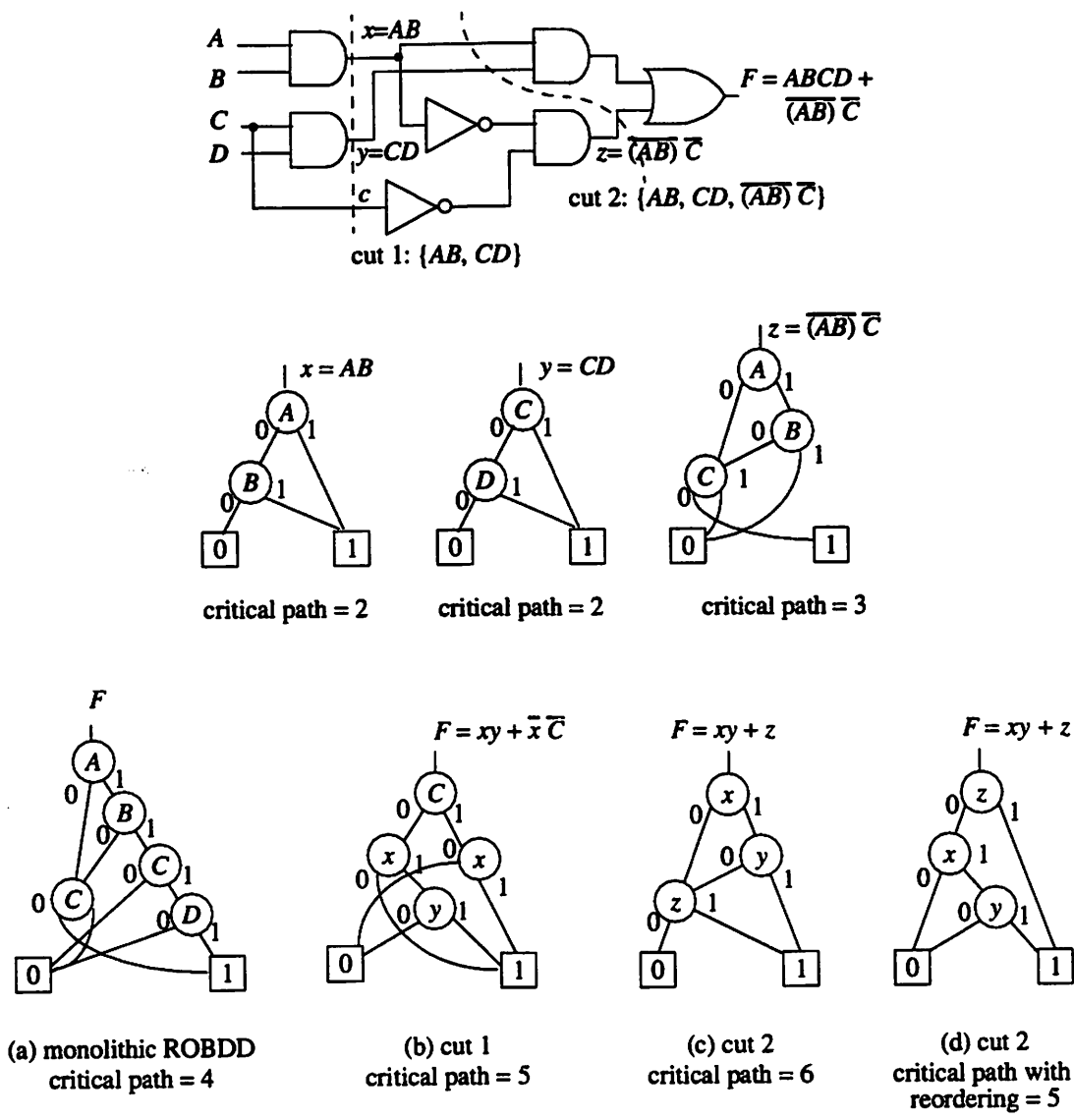


Figure 4-9: High performance heuristics (ordering:  $A, B, C, D, x, y, z$ )

decomposed BDDs have two advantages compared to monolithic BDDs: the multi-level nature of the resulting circuit allows logic signals to be computed in parallel (e.g.,  $x$  and  $y$ , both would be computed in two time units for the decomposition in Fig. 4-9). Secondly, when circuit level issues are considered, the quadratic dependence of delay on the transistor chain length more than offsets

any advantages of a shorter critical path of a monolithic BDD which consists of a single transistor chain as compared the critical path of a decomposed BDD which has several buffered stages. For today's static CMOS it is known that transistor chains longer than 3-4 transistors in series can be unacceptably slow [Rab96]. A monolithic BDD-based circuit would require buffer insertion as in [Yan96] for all but the smallest circuits. In comparison, a decomposed BDD-based circuit where outputs of decomposition points are buffered allows exploiting area gains (and the associated reduction in switching capacitance) while controlling the length of unbuffered chains. Selecting decomposition points with appropriate thresholds on the depth of decomposition point BDDs is thus a more powerful strategy than selecting buffer insertion points in a monolithic BDD.

Apart from directly controlling the depth of decomposition point BDDs, the choice of decomposition points can be targeted at minimizing the upper bound on the decomposition point BDD delay when speed is the main concern. If a cutset<sup>1</sup> of the circuit is selected as the set of decomposition points, then the critical path in the BDDs of the primary outputs is bounded by the cutset cardinality (because BDDs of the primary outputs can be constructed in terms of the cutset variables only). Using the minimum cardinality cutset of the circuit as the decomposition set is then a good heuristic to reduce the critical path length. An example to illustrate this heuristic is shown in Fig. 4-9. Fig. 4-9(b) and Fig. 4-9(c) compare the critical path length when two different cutsets are chosen as decomposition points. The critical path is lower when the mincut is selected as the decomposition set (Fig. 4-9(b)) but longer than the monolithic ROBDD (Fig. 4-9(a)).

BDD variable ordering has a great impact on the BDD size and consequently the circuit area. This ordering can also influence the circuit power and speed. Placing late arriving signals

---

1. set of nodes such that all paths from primary inputs to primary outputs pass through some node in the set

closer to the outputs can speed-up the circuit by minimizing the number of transistors that need to be charged after the late signal arrives. Signal flow in a BDD-based PTL network corresponds to traversing the BDD from leaf nodes up. Thus it is advantageous to place late arriving control variables close to the top of the BDD. Variables in a BDD can be swapped pairwise as long as the resulting variable order does not cause the BDD size to increase significantly. In the example of Fig. 4-9, placing the late arriving signal  $z$  at the top (Fig. 4-9(d)) reduces the critical path by 1 unit over Fig. 4-9(c). The best known dynamic reordering algorithms for BDD size ([Rud93][Pap91]) move each variable or a block of variables throughout the order to find an optimal position for the variable. A similar reordering can be performed for delay, where the optimal position is the one resulting in the smallest depth BDD instead of the smallest sized BDD. As in Section 4.5.1, each decomposition point BDD can be reordered independently to optimize for total delay.

### **4.5.3. Low Power**

Power dissipation in a circuit is a function of switching capacitance and switching activity. It is thus desirable that the capacitance connected to nodes with high switching activity is minimized. Since the gate capacitance of a transistor is substantially higher than the drain/source capacitances, this translates into ensuring that the high switching activity nodes are not connected to the gates of too many transistors. Note that neglecting drain-source capacitance switching is analogous to ignoring the internal node switching in a static CMOS gate.

In the case of PTL networks, only control variables are connected to the gate terminals of transistors. In our decomposed BDD-based approach, the control variables consist of primary inputs and decomposition points. Note that every node in the BDD is implemented as a multiplexer in the corresponding circuit and the node variable in the BDD is connected to the gates of two transistors of the multiplexer. Minimizing the occurrences of high switching activity node then

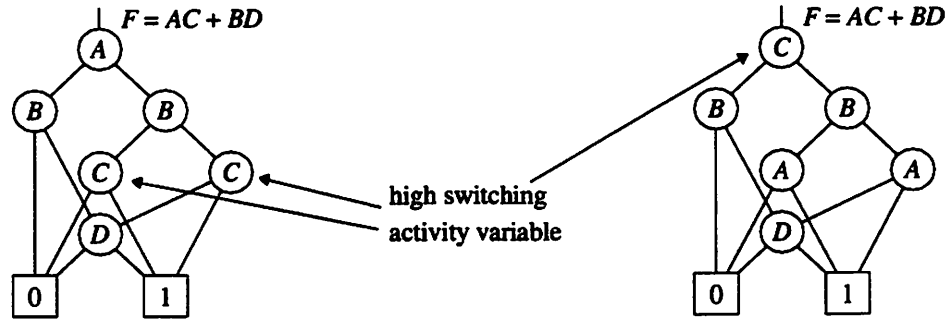


Figure 4-10 : Reducing occurrences of high switching activity node

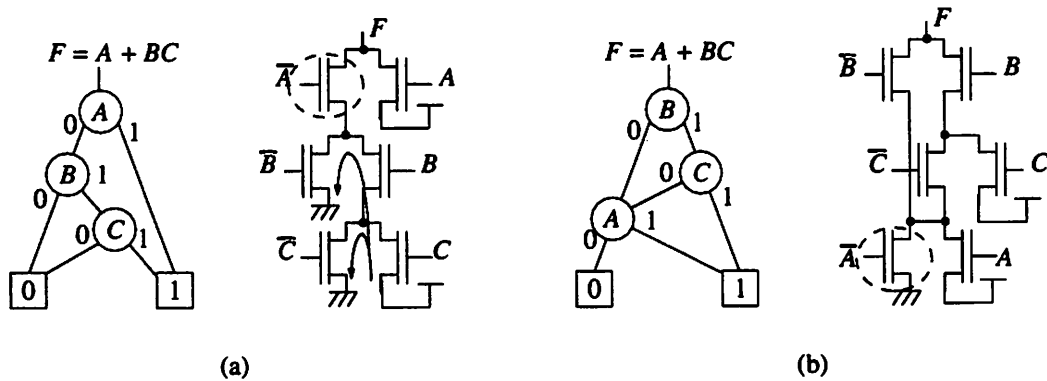


Figure 4-11 : Low Power heuristic to minimize glitching:  $F = A + BC, p(A=1) \approx 1 \Rightarrow p(F=1) \approx 1$

translates into minimizing the occurrences of the corresponding variable in the BDD. Re-ordering BDD variables can be used to achieve this. Fig. 4-10 illustrates a case where re-ordering reduces the occurrences of the high switching activity variable at the expense of more occurrences of a lower switching activity variable.

A node in the PTL network is charged high when there is a path connecting it to the power supply and discharged when there is a path connecting it to ground. Even when the output does not change, glitching (charging and discharging of internal nodes) can consume a significant amount of power. Glitching can be minimized by placing variables which have a low switching probability

close to the bottom of the BDD. This implies that the transistors controlled by these variables are close to the power supply and ground in the PTL network. Depending upon their state, this will cut-off the rest of the PTL network from the power supply or ground, resulting in a lower switching power dissipation. In the example in Fig. 4-11, since the probability of  $A$  being high is almost 1, the nMOS connected to  $A$  is almost always cut-off, and  $F$  is almost always 1. The ordering in Fig. 4-11(a) can however result in a significant power dissipation due to the internal nodes being charged through the nMOS connected to  $C$  and discharged through the nMOS connected to  $B$  and  $C$ . Compared to this, the ordering in Fig. 4-11(b) has no internal power dissipation as the nMOS connected to  $A$  cuts-off the rest of the circuit from ground.

Note that these heuristics are similar to re-ordering transistors for low power at the circuit level in static CMOS ([Hos96]). However, in our approach, technology mapping is straightforward and there is a one-to-one correspondence between BDDs and PTL circuits. This allows us to perform Boolean manipulations at a high level in which we can trade-off circuit area for power, rather than making restricted structural changes at the circuit level.

## 4.6. Results

The techniques described above have been tested on ISCAS benchmarks circuits, which include circuits which are *hard* for monolithic BDD-based approaches (e.g. C6288 - the multiplier circuit). This section presents results comparing our PTL synthesis algorithm with different static CMOS synthesis algorithms to demonstrate the area and delay gains achieved by the proposed approach, and HSPICE simulation results to verify the validity of the logic level results.

The PTL synthesis algorithm was implemented in the SIS framework. It is compared against four synthesis scripts for static CMOS: area and delay optimization scripts which do not use don't cares, and *script.rugged* and *script.delay* of SIS. Technology mapping was performed using

three different libraries: msu.genlib, 33-4.genlib, and 44-3.genlib. All experiments were carried out on a 400 MHz DEC Alpha with a SPECint\_92 rating of 341, DEC 21164 CPU, 4Mb cache and 2Gb total memory.

PTL circuits were synthesized with four different threshold parameters. This threshold parameter from Section 4.5.1 controls the depth of decomposition point BDDs. For a given logic circuit, the best of the four PTL transistor-level circuits was selected and data for this circuit is presented in all tables. Compared to this, static CMOS circuit in each table is the best of several test runs with different parameters. Moreover, not the same circuit is used in all tables. That is, the same PTL circuit data is compared with the area optimized static CMOS circuit in the area columns of the tables and against the delay optimized static CMOS circuit in the delay columns. The gains achieved by PTL are thus very conservative, since the area-optimal static CMOS circuit is far from delay-optimal and vice versa.

The PTL results against results from area and delay optimization scripts for static CMOS which do not use don't cares (mapped for area and delay respectively using msu.genlib) are compared in Table 4-1. Since this PTL implementation does not perform don't care optimization yet, this table gives the best picture of the efficiency of the PTL algorithm. Column 1 contains the names of the ISCAS benchmark circuits. Column 2 and 3 compare the active gate area (measured in  $\lambda^2$ ) of circuits synthesized by the PTL and minimum area static CMOS algorithm. Column 4 contains the relative gain in area achieved by our PTL algorithm over the static CMOS algorithm. Columns 5 and 6 compare the critical path length of the circuits generated by PTL and minimum delay static CMOS. Column 7 contains the relative gains of PTL over the static CMOS algorithm.

Table 4-2 presents results in the same format, this time comparing the PTL data of Table 4-1 with static CMOS results optimized using local optimizations and full don't cares. The static

CMOS results in Column 2 are optimized using *script.rugged* and the results in Column 5 are optimized using *script.delay*. Note that the current PTL implementation does not perform local optimizations or don't care optimizations. This is not an algorithmic limitation, and techniques for these optimizations in the decomposed BDD based PTL synthesis context are outlined in Section 4.7.1. In spite of this handicap, the current PTL implementation yields impressive gains over the *script.rugged* and *script.delay*.

Table 4-3 compares the runtimes of the PTL synthesis algorithm (column 6) with the area and delay optimization scripts without don't cares, and *script.rugged* and *script.delay* (column 2, column 3, column 4 and column 5 respectively). Note that the output of static CMOS algorithms is a mapped logic network while the output of the PTL algorithm is an HSPICE netlist. The results clearly indicate that PTL synthesis is substantially faster than all static CMOS techniques.

When using the critical path length as the metric to compare delays of two circuits, it is important to ensure that the amount of logic implemented in a cell is similar for each case, because a circuit with large individual cells can have a small critical path but be slow due to high cell propagation delay. Table 4-4 compares the cell count of the PTL circuit with the area and delay optimized static CMOS circuits of Table 4-1 in column 2, column 3 and column 4 respectively, and the average cell size in column 5, column 6, and column 7 respectively. The data indicates that the PTL circuit indeed uses fewer cells with more logic in each individual cell. However, the delay of each cell is not directly related to the cell size since each cell in the PTL circuit implements a BDD structure, while the static CMOS cells implement a series-parallel pull-up and pull-down tree structure. We analyze a full adder circuit and perform HSPICE analysis to examine the delay trade-off between a larger cell implementing a BDD structure vs. a smaller series-parallel logic cell. These results are presented in Fig. 4-12.

As an aside, static CMOS circuits were also synthesized using larger libraries like 33-



4.genlib (87 cells, average cell size:  $27.4 \lambda^2$ ) and 44-4.genlib (625 cells, average cell size  $43.4 \lambda^2$ ) to see if a greater choice of cells, including very large cells, improved the static CMOS results. However, it was found that there was no major change in the results, and in fact, the synthesis runtimes for static CMOS increased by factors of 5 -100 due to the library size.

Table 4-5 presents the logic synthesis and HSPICE analysis results for a full adder circuit implemented in PTL and static CMOS. Column 2, column 3, column 4, and column 5 provide logic level data (area, number of transistors, number of cells and average cell size respectively) for the two test cases. Column 6 and column 7 contain the slowest rise and fall times from an exhaustive HSPICE simulation. Fig. 4-12 (a) and (b) show the rise and fall time waveform plot. The results indicate that PTL has a smaller fall time and the same rise time as static CMOS. This is to be expected since an nMOS-only PTL circuit is good at conducting "0". The critical path thus seems to be a good indicator of the fall times. In general, a lower area implies a smaller switching capacitance, which does indeed correlate with a faster circuits. Thus, the 20-50+% gains achieved at logic level should translate to gains at the transistor level, albeit in slightly smaller numbers. (Note that the delay gains are not really *reduced* in mapping to the transistor-level. This analysis is aimed only at providing a good understanding of the logic-level critical path length as a metric of delay at the transistor-level).

Column 8 and column 9 present the average and rms power dissipation results. The static CMOS circuit has a lower average power dissipation but a higher rms power dissipation. Fig. 4-12(c) indicates that the static CMOS has a higher peak power dissipation as well. This can be explained by the fact that static CMOS has a higher switching capacitance, while PTL may have a higher leakage current. While the lower average power dissipation of static CMOS is good from the battery life perspective, a higher peak and rms power dissipation is undesirable from the electromigration and IR drop point of view.

Circuit	Area			Delay		
	static CMOS (Area Opt.)	PTL	gain	static CMOS (Delay Opt.)	PTL	gain
C17	54.0	58.5	-8 %	3	2	33 %
C432	1620.0	1468.5	9 %	18	23	-28 %
C499	3424.5	2920.5	15 %	12	9	25 %
C880	2673.0	2433.0	9 %	16	9	44 %
C1355	3424.5	2953.5	14 %	14	6	57 %
C1908	4851.0	3174.0	35 %	20	14	30 %
C2670	5787.0	4797.0	17 %	18	11	39 %
C3540	9279.0	7495.5	19 %	28	18	36 %
C5315	13266.0	12415.5	6 %	24	11	54 %
C6288	21321.0	16180.5	24 %	120	70	42 %
C7552	18310.5	19902.0	-9 %	21	14	33 %

Table 4-1 : Comparing the best area static CMOS vs. PTL and the best delay static CMOS vs. PTL (Area is measured in  $\lambda^2$  and delay is measured as the length of the critical (longest topological) path)

Circuit	Area			Delay		
	static CMOS ( <i>script.rugged</i> with full DC)	PTL (without DC)	gain	static CMOS ( <i>script.delay</i> with full DC)	PTL (without DC)	gain
C17	49.5	58.5	-16 %	3	2	33 %
C432	1233.0	1468.5	-19 %	18	23	-28 %
C499	3244.5	2920.5	10 %	11	9	18 %
C880	2596.5	2433.0	6 %	16	9	44 %
C1355	3244.5	2953.5	9 %	11	6	45 %
C1908	3226.5	3174.0	2 %	18	14	22 %
C2670	4491.0	4797.0	-9 %	-	11	-
C3540	8176.5	7495.5	8 %	26	18	31 %
C5315	9985.5	12415.5	-24 %	23	11	52 %
C6288	19885.5	16180.5	19 %	61	70	-15 %
C7552	-	19902.0	-	-	14	-

Table 4-2 : Comparing static CMOS optimized using local minimizations and full Don't Cares (using *script.rugged* for area and *script.delay* for delay) vs. PTL (which does not use local optimizations or any Don't Cares in this implementation). (Area is measured in  $\lambda^2$  and delay is measured as the length of the critical path). A "-" indicates that the program could not complete due to space out.

Circuit	static CMOS				PTL
	Area optimized	Delay optimized	<i>script.rugged</i> with full DC	<i>script.delay</i> with full DC	
C17	0.01	0.01	0.10	0.10	0.01
C432	0.6	0.6	113.2	91.4	0.2
C499	1.1	1.0	12.9	10.5	0.6
C880	0.9	0.9	4.3	11.0	0.4
C1355	1.4	1.4	13.3	24.4	0.6
C1908	1.7	1.7	15.9	47.7	1.1
C2670	3.0	2.6	100.4	-	1.8
C3540	4.3	3.5	30.0	371.4	2.8
C5315	7.2	5.9	22.9	664.9	5.1
C6288	5.5	6.1	65.0	287.2	10.4
C7552	10.1	8.6	-	-	14.3

Table 4-3 : Comparing static CMOS vs. PTL in runtime (measured in seconds) (Note that the output of static CMOS algorithms is a mapped logic network while the output of the PTL algorithm is an HSPICE netlist). A "-" indicates that the program could not complete due to space out.

Circuit	Cell Count			Average Cell Size		
	static CMOS		PTL	static CMOS		PTL
	Area opt.	Delay opt.		Area opt.	Delay opt.	
C17	6	6	3	6.0	6.0	13.0
C432	141	144	58	7.7	7.7	16.9
C499	238	243	133	9.6	9.8	14.6
C880	223	215	94	8.0	8.8	17.2
C1355	238	235	41	9.6	10.8	48.0
C1908	356	345	189	9.1	10.1	11.2
C2670	425	514	227	9.1	8.8	14.1
C3540	778	789	382	8.0	8.5	13.1
C5315	1030	1238	316	8.6	8.7	26.2
C6288	2326	2340	929	6.1	6.1	11.6
C7552	1596	1512	989	7.6	8.3	13.4

Table 4-4 : Comparing static CMOS of Table 4-1 vs. PTL for cell counts and average cell size (measured in  $\lambda^2$ )

Logic	Area ( $\lambda^2$ )	#MOS	#Cell	Avg Cell Size ( $\lambda^2$ )	Delay			Power	
					Critical Path	from HSPICE		Average	RMS
						$t_{rise}$	$t_{fall}$		
PTL	42	22	1	42.0	1	0.6 ns	0.2 ns	30 $\mu$ W	83 $\mu$ W
static CMOS	81	36	8	10.1	4	0.6 ns	0.5 ns	17 $\mu$ W	122 $\mu$ W

Table 4-5 : Experimental data for a full adder circuit

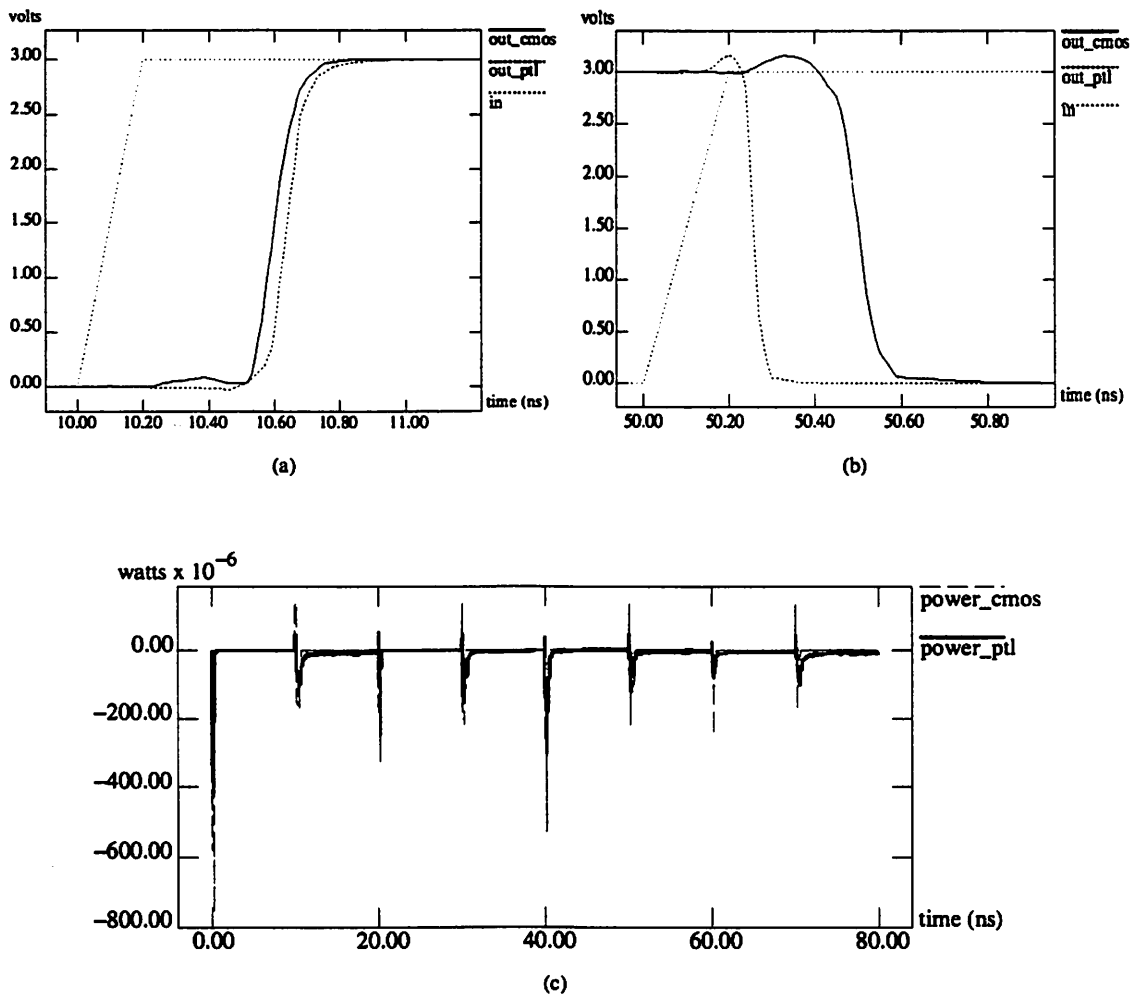


Figure 4-12: HSPICE results on timing and power dissipation of a full adder circuit implemented in static CMOS and PTL

## **4.7. Enhancing the Decomposed BDD-based Approach**

### **4.7.1. Don't Care Optimization**

The PTL synthesis tool benchmarked in Section 4.6 does not use don't cares for design optimization. Don't cares provide a significant amount of flexibility in minimizing a circuit as witnessed from the improvement of the static CMOS area and delay results between Table 4-1 and Table 4-2 in Section 4.6.

Extending the proposed approach to handle don't cares is relatively straightforward. Several heuristics to minimize BDD size using don't cares are presented in [Shi94]. Since the area of a decomposed BDD-based PTL circuit is proportional to the BDD size, the results of [Shi94] can be applied to PTL synthesis directly. The synthesis algorithm would then be modified as follows: after generating the decomposed BDD representation, the target function BDD and each decomposition point BDD are minimized while travelling from the primary outputs of the circuit to primary inputs. In the context of the multi-stage circuit represented by the decomposed BDDs, travelling from the outputs to the inputs amounts to first minimizing the target function BDD and then each decomposed BDD in the reverse order of decomposition point introduction. For each BDD, the compatible observability don't cares for the output function are computed in terms of the primary inputs. This is mapped to a local don't care set via image computation. The don't care set construction is the same as in the case of the multi-level network minimization and the reader is referred to [Sav90b] for more details. The heuristics of [Shi94] are then applied to minimize the BDD. Based on the results reported in [Shi94], this extension can be expected to yield significant reduction in the area of the PTL circuits.

### **4.7.2. Synthesis of Mixed static CMOS/PTL Circuits**

This work has proposed the use of PTL for large deep sub-micron designs. PTL can provide

substantial gains in area and delay over static CMOS, while the static CMOS has the advantage of a well-established design flow for synthesizing robust circuits. Static CMOS may be preferable over PTL in cases where a static CMOS implementation of a gate is particularly efficient, or where an nMOS conducting “1” is not allowed.

The PTL synthesis flow proposed in this work is very general in nature and allows synthesis of mixed static CMOS/PTL circuits which can leverage the strengths of static CMOS as well as PTL as appropriate. Each decomposition point BDD can be viewed as a complex node and can be implemented by static CMOS logic or PTL as desired.

Among other issues, currently ROBDDs are used as the underlying data structure for the decomposed BDDs. General BDDs ([Ash91]), which allow input variables to appear multiple times along any path in the BDD, may be more appropriate from the PTL network design point of view since in this case compactness is of more interest than canonicity.

## **4.8. Conclusions**

We have outlined a methodology for synthesizing large pass transistor networks. The main contributions of this work are the following:

- A decomposed BDD-based representation was proposed to take advantage of the correspondence between PTL circuits and BDDs without suffering from the drawbacks imposed by properties of monolithic BDDs.
- A comprehensive synthesis flow was outlined for PTL design. It was showed that the proposed approach allows logic level optimizations similar to the traditional multi-level network based synthesis flow for static CMOS, and also makes possible optimizations with a direct impact on

area, delay and power of the final circuit implementation which do not have any equivalent in the traditional approach. Using these techniques PTL circuits could be synthesized for the entire ISCAS benchmark set.

- A set of heuristical algorithms to synthesize PTL circuits optimized for area, delay and power which are key to the proposed synthesis flow, were presented. These algorithms are very intuitive and simple and have a great impact on the optimality of the resulting circuit.

Experimental results on ISCAS benchmark circuits show that the proposed technique yields PTL circuits with substantial improvements in area and delay over conventional static CMOS designs. We believe that with more research in this area PTL can become a viable alternative to static CMOS, and that this work is the first step in that direction.

#### **4.9. Summary**

Synthesis techniques for pass transistor logic design were presented. The merits of pass transistor logic as an alternative to static CMOS for deep sub-micron design were discussed and the need for CAD algorithms for PTL circuit design was motivated. Decomposed BDDs were proposed as a suitable logic level representation for the synthesis of PTL networks. Decomposed BDDs can represent large, arbitrary functions as a truly multi-stage circuit and can exploit the natural, efficient mapping of a BDD to PTL. A synthesis flow based on decomposed BDDs was outlined and compared with the traditional static CMOS standard cell based flow. Synthesis algorithms for area, delay and power minimization were presented. Experimental results on ISCAS benchmark circuits show an average reduction of 12% in area and 33% in delay compared to static CMOS designs. To the best of our knowledge this is the first time PTL circuits have been synthesized for the entire ISCAS benchmark set.

---

# 5 Conclusions

---

A set of CAD algorithms for high performance circuit design were presented in dissertation. The area of high performance circuit design is currently going through a paradigm shift in terms of the cost functions and design objectives. Until very recently, minimizing area and delay were the two main design objectives. With the growing device counts and operating frequencies, power is now a major design concern. Also, with the advent of deep submicron technologies, the delays are now associated mostly with the interconnect. In this context, the research presented in this dissertation addressed the problem of algorithms for these design objectives for the current and next generation of circuits.

Specifically, the problem of CAD for low power design was approached from two aspects: estimation and synthesis. Techniques for power estimation at the transistor level of circuit abstraction and optimization algorithms for minimizing power at the logic level without trading off delay or area were presented.

The problem of next generation logic synthesis was approached in the context of the need of alternatives to the traditional static CMOS logic and the increasing irrelevance of standard cell based design methodology in addressing the deep submicron design concerns. Library free synthesis algorithms for designing pass transistor logic (PTL) networks were presented as a solution to this problem.

The work in Chapter 2 addresses the transistor level power estimation problem by



proposing a fast circuit simulation technique to speed-up power estimation and a methodology to obtain statistically significant estimates without externally provided input stimuli.

SYMPHONY, a mixed signal simulator which exploits the special characteristics of BiMOS mixed signal circuits was presented. SYMPHONY contains a fast simulator for digital circuits and several new techniques to efficiently simulate BiMOS circuits. The typical switching behavior of bipolar devices in digital setting was exploited by using a simplified model to approximate the bipolar device characteristics. The problem of minimizing the worst case approximation error was formulated and a heuristic proposed to achieve this by using a PWL model with expanded Chebyshev points as the breakpoints. Dynamic circuit partitioning was combined with an event-driven approach to exploit the latency and multi-rate behavior. Experimental results showed that SYMPHONY can yield 2x-250x speed-up over SPICE3e depending upon the amount of analog circuitry present in the design. SYMPHONY was faster and more accurate than PowerMill when applied to power simulation.

A stochastic model was developed for power dissipation at the transistor level, and certain properties proved for this model. Power estimation was then reduced to a mean estimation problem using these properties. A Monte Carlo approach was used for mean estimation. A formal stopping criterion was derived to guarantee a desired error bound at a specified confidence level, under the assumption that power dissipation in a clock cycle is normally distributed. To further speed-up the estimation process, a divide-and-conquer approach was used to break down the problem in sizes that become practically feasible for transistor-level simulation. The main advantage of partitioning for estimation is that the multi-rate behavior and stiffness of a circuit from the power perspective can be exploited. A statistical model was used to propagate signal information between partitions of a circuit. Experimental results showed that the power estimator converged to a power estimate in very reasonable runtimes and met the prescribed error bounds in all cases.

The goal of logic synthesis is generation and optimization of a multi-level logic description which implements a specified function. In the work of Chapter 3, the objective is minimization of power at the technology independent and dependent stages of logic optimization without trading off delay or area.

The Boolean space spanned by the primary input vectors of a combinational function can contain a large variance in minterm probabilities. It was shown that the Boolean space can be partitioned into classes such that classes containing only 10% of the Boolean space cover as much as 90% of the total probability. This characteristic of the minterm probability distribution was exploited to reduce power dissipation. The notion of power sensitive minterms was introduced to capture the parts of the Boolean space which can significantly affect power dissipation of a function. An algorithm for technology independent power optimization was proposed which used power sensitive minterms along with the don't cares of a function to minimize its switching activity without compromising the flexibility for delay/area optimization. The results showed that an average power reduction of 16% was achieved over all test cases without any area penalty.

Power sensitive minterms can also be used to guide resynthesis techniques for low power. The objective of resynthesis for low power is to make changes in a synthesized multi-level network such that the power dissipation of the circuit is reduced. However, any additional circuitry required to implement this change can possibly offset the power reduction achieved by the onset change. An engineering change based formulation was presented which aims at minimum modification of a technology mapped circuit to realize the new specification. Rewiring was used to solve this problem such that power reduction is achieved without increasing the circuit area or delay (under a unit delay model). The results showed that power reductions of up to 13%, with an average reduction of 4%, were achieved over a set of benchmark circuits.

Pass transistor logic (PTL) can be a promising alternative to static CMOS for deep sub-

micron design and has the potential to improve area, delay, and reduce power consumption in interconnect dominated technologies. Chapter 4 motivated the need for CAD algorithms for PTL circuit design and proposed decomposed BDDs as a suitable logic level representation for synthesis of PTL networks. Decomposed BDDs can represent large, arbitrary functions as a multi-stage circuit and can exploit the natural, efficient mapping of a BDD to PTL.

A comprehensive synthesis flow based on decomposed BDDs was outlined for PTL design. It was shown that the proposed approach allows logic-level optimizations similar to the traditional multi-level network and standard cell based synthesis flow for static CMOS. Moreover, this approach also makes possible optimizations with a direct impact on area, delay and power of the final circuit implementation which do not have any equivalent in the traditional approach. A set of heuristical algorithms were presented to synthesize PTL circuits optimized for area, delay and power. Experimental results on ISCAS benchmark circuits showed that the proposed technique yields PTL circuits with substantial improvements over static CMOS designs. To the best of our knowledge this was the first time PTL circuits were synthesized for the entire ISCAS benchmark set.

## **5.1. Future Work**

The research work presented in this dissertation has focused on both estimation and synthesis aspects of a design methodology for deep submicron, low power circuits. The following discusses some specific extensions to this research as well as directions for future work in the area of synthesis for deep submicron technologies in general.

A major limitation of the logic synthesis algorithms for low power design presented here and by other researchers is that the power dissipation models input to the synthesis algorithms are not very accurate or monotonic (with respect to the power dissipation of the final circuit extracted

from the layout). As a result, the gains achieved at the logic level do not always translate to gains in the final silicon implementation. This is primarily due to the fact that these models do not account for spatiotemporal correlations or the glitching power dissipation (Section 3.2). A comprehensive study on the monotonicity of these models and the synthesis techniques which employ them would be very useful for a designer in making power optimization decisions.

The work presented in this dissertation attempted to minimize the impact of such a loss of gains as described above by proposing techniques based on the minterm probability distributions in the primary input space. While this can take care of spatial correlations, it still does not account for temporal correlations or glitching power. This is not a fundamental limitation of the algorithm, since this and most of the other published algorithms use probability computation as a black box routine, but is due to the fact that at present considering all spatiotemporal correlations is prohibitively expensive in terms of both runtimes and memory requirements. Further research alleviating these problems can make it practical to use these models in guiding the existing synthesis techniques.

The technology independent power optimization algorithm presented in Chapter 3 can itself be enhanced as follows - currently, the power sensitive minterms are computed by grouping together like probability minterms and selecting the highest average minterms probability groups. These minterms are then applied towards power optimization and the remaining part of the Boolean space is applied towards area/delay optimization. In general, this would be expected to provide substantial power savings without significantly affecting the area/delay optimization flexibility. The results presented in Chapter 3 show that substantial power savings can indeed be achieved by this approach. This however is not the best that this approach can yield. This is because the power sensitive minterms are few and randomly distributed throughout the Boolean space. As a result, the subset of the Boolean space formed by these minterms as well as the subset formed by taking these

away from the Boolean space are in a sense disconnected. Consequently, minimization algorithms a la ESPRESSO [Rud87], which rely on expanding and reducing the cube cover of a function in the Boolean space are obstructed by these discontinuities and do not perform as well as may be expected from just the proportion of the probability and minterm space covered by these subsets. Smoothing out the power sensitive minterm set so that it is a strongly connected set of minterms instead of the highest probability minterms only, can provide more flexibility to the minimization algorithms and realize the full potential of the savings possible from the power sensitive minterms based approach.

The technology dependent power optimization algorithm presented in Chapter 3 can be further improved by changing the cost function in the selection of the new wiring of the circuit. Currently, the rewiring algorithm generates a set of possible wiring solutions for the circuit, all of which satisfy the circuit functionality and are expected to have a lower power dissipation. Selecting the lowest power wiring among these without exhaustively computing the power dissipation for each solution is a very difficult problem, and currently the minimum wire solution is selected as a heuristic since it minimizes the number of internal signals which can switch and dissipate power. An efficient technique to select the lowest power wiring given the input probabilities could yield larger power savings over this.

The pass transistor logic synthesis work presented in Chapter 4 can fit in a synthesis framework for deep submicron technologies and also be used as a point tool. Some enhancements to this work like don't care optimization and mixed static CMOS-PTL design have already been outlined in detail in Section 4.7 and their implementation is straightforward.

The work on the deep submicron synthesis framework is currently underway [Nex97]. This framework aims at optimizing the circuit at the logic level with the interconnect delay, noise, crosstalk and other physical design considerations in mind. The current over-all architecture and

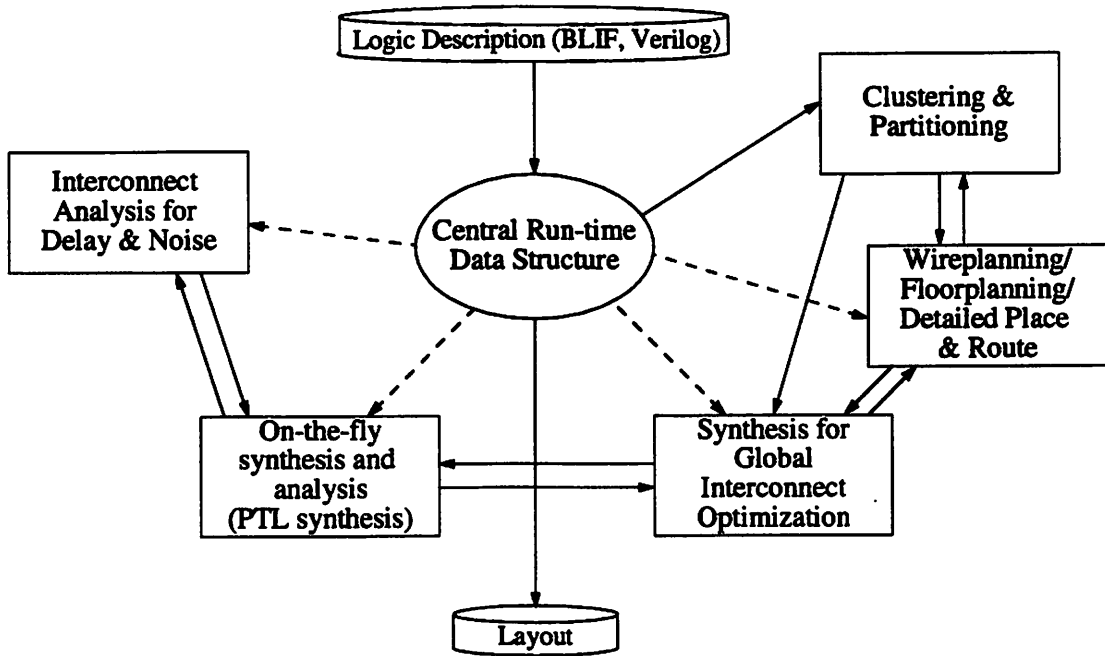


Figure 5-1 : A Framework for synthesis in deep submicron technologies [Nex97]

flow of this framework are presented in Fig. 5-1. The optimizations are performed by first partitioning and floorplanning the circuit so that physical locations can be associated with different parts of the circuit. The partitions are of a size such that the wires within a partition can be considered as short wires who do not contribute significantly to path delays and only long wires crossing partitions need be considered for delay optimization. The circuit is then optimized for interconnect delay by minimizing the number of long wires by various logic optimization techniques. Each partition is then synthesized on-the-fly using static CMOS, PTL, domino logic, or a combination of these logics. This is the context in which the PTL synthesis work fits in the deep submicron synthesis framework. This process of global interconnect optimization followed by on-the-fly synthesis may be iterated to improve the solution quality. After this on-the-fly synthesis stage, detailed place and route is performed to obtain the final layout of the circuit. The interested

reader is referred to [Nex97] for more details.

A big challenge in applying library-free on-the-fly-synthesis like the PTL synthesis technique presented in this work to a framework as described above is circuit characterization. Since the circuits are not implemented using pre-characterized standard cells, it is very difficult to estimate the circuit delay, area, and power, and consequently guide the optimization algorithms. The effect of this is most severe in delay optimization as delay characteristics of PTL circuits can be affected by the environment it operates in. This is because the drain and sources of transistors chains can now be connected to other signals in the circuit (as compared to  $V_{dd}$  and ground in the static CMOS pull-up pull-down network case). This makes the delay of a cell dependent on other signal delays in the circuit. The particular synthesis algorithm presented in this dissertation does not suffer from this problem because in this case each cell corresponds to a decomposed BDD whose terminals are  $V_{dd}$  and ground. However, the general problem of circuit characterization in a library-free synthesis environment however is relevant to this case as well. Efficient characterization algorithms for this would greatly increase the potential of PTL and the proposed synthesis methodology.

The PTL synthesis algorithm itself can be extended in several ways. The proposed algorithm uses decomposed ROBDDs as the logic level representation. A critical requirement of the data structure in order to directly map to it to a circuit is that a node in the logic level representation should be efficiently implementable in PTL. ROBDDs use shannon decomposition [Sha38], which translates to a multiplexer element in the corresponding circuit implementation - an element that is very compactly implemented using PTL (four MOS transistors as compared to ten in the case of static CMOS). It would be of great interest to explore what are the other such decompositions that are suitable for PTL implementation. XOR-based decomposition can be one possible decomposition strategy since XOR is also very efficiently implemented in PTL (four MOS

transistors as compared to twelve in the case of static CMOS). A considerable research on XOR-based decompositions like Fixed-Polarity Reed-Muller form [Ree54][Mul54] with positive and negative Davio expansions [Dav78], etc., exists in the area of decision diagrams [Sar93][Sas93a][Sas93b][Ste95] [Tsa94], with ordered functional decision diagrams [Keb93] and ordered Kronecker functional decision diagrams [Bec94] being some of the data structures of interest. There has also been work on different strategies for a multiplexer-based implementation of a circuit and its properties [Ash93a] [Ash93b][Tha96], as well as combining other synthesis strategies like spectral methods with BDDs [Han96]. This may be of interest from the perspective of developing better heuristics for optimizing the existing decomposition strategy in the proposed PTL synthesis algorithm. Finally, some classes of circuits are more suitable to synthesis using PTL than others. Arithmetic circuits is one class of circuits which can be efficiently implemented in PTL since it is known that they have a very compact description using the AND and XOR operators [Tsa96]. It would be of great interest to study what are all the different classes of circuit which are most amenable to synthesis using PTL.



---

## Bibliography

---

- [Acu90] E. L. Acuna, J. P. Dervenis, A. J. Pagones, F. L. Yang, and R. A. Saleh, "Simulation techniques for mixed analog/digital circuits," *IEEE Journal of Solid-State Circuits*, Vol. 25, No., pp. 353-363, April 1990.
- [Ake78] S. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, Vol. C-27, No. 6, June 1978.
- [AlA91] W. Al-Assadi, A.P. Jayasumana, and Y.K. Malaiya, "Pass-transistor logic design," *International Journal of Electronics*, Vol. 70, No. 4, 1991.
- [And90] D. Anderson, "The 68040 32-b monolithic processor," *IEEE Journal of Solid State Circuits*, Vol. 25, No. 5, pp. 1178-1189, October 1990.
- [Arn78] G. Arnout, and H. De Man, "The use of threshold functions and boolean controlled network elements for macromodeling of LSI circuits," *IEEE Journal of Solid-State Circuits*, Vol. SC-13, pp. 326-332, June 1978.
- [Ash91] P. Ashar, A. Ghosh, and S. Devadas, "Boolean satisfiability and equivalence checking using general binary decision diagrams," *Proceedings of the International Conference on Computer Design*, 1991.
- [Ash93a] P. Ashar, S. Devadas, and K. Keutzer, "Path-delay-fault testability of multiplexor-based networks," *Integration, the VLSI Journal*, Vol. 15, pp. 1-23, 1993.
- [Ash93b] P. Ashar, S. Devadas, and K. Keutzer, "Gate-delay-fault testability of multiplexor-based networks," *Formal Methods in System Design*, Vol. 2, pp. 93-112, 1993.

- [Bah95] R. I. Bahar, and F. Somenzi, "Boolean techniques for low power driven resynthesis," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 428-432, Nov. 1995.
- [Bah96] R. I. Bahar, M. Burns, G. Hachtel, E. Macii, H. Shin, and F. Somenzi, "Symbolic computation of logic implications for technology-dependent low-power synthesis," *Proceedings of the ACM International Symposium on Low Power Electronics & Design*, pp. 163-168, August 1996.
- [Ban88] R. L. Baner, A. Ng, J. Fang, and R. K. Brayton, "XPsim: a MOS VLSI simulator," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 66-69, November 1988.
- [Bar87] K. A. Bartlett, G. D. Bostick, G. D. Hachtel, R. M. Jacoby, P. H. Lightner, P. H. Moceyunas, C. R. Morrison, and D. Ravenscroft, "BOLD: A multiple-level logic optimization system," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 59-73, November 1987.
- [Bec94] B. Becker, and R. Dreschler, "Synthesis for testability: circuits derived from ordered Kronecker functional decision diagrams," *Technical Report, Universität Frankfurt*, 14/94, Fachbereich Informatik, 1994.
- [Boy78] G. R. Boyle, "Simulation of integrated injection logic," *Ph.D. Dissertation, University of California, Berkeley*, 1978.
- [Bra84] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.
- [Bra87] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "MIS: Multiple-Level Logic Optimization System," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-6, pp. 1062-1081, November 1987.
- [Bry84] R. E. Bryant, "A switch-level model and simulator for MOS digital systems," *IEEE Transactions on Computers*, Vol. C-33, No. 2, pp. 160-177, February 1984.
- [Bry86] R.E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE*

*Transactions on Computers*, Vol. C-35, No. 8, pp. 677-691, Aug. 1986.

- [Buc94] P. Buch, and T. Xue, "A comparative study of PLAs and standard cell design for low power circuits," *EE 241 Project Report, University of California, Berkeley*, May 1994.
- [Buc95a] P. Buch, J. Sanghavi and A. Sangiovanni-Vincentelli, "A parallel graph partitioner on a distributed memory multiprocessor," *Proc. of the 5th IEEE Symposium on the Frontiers of Massively Parallel Computation*, pp. 360-366, Feb. 1995.
- [Buc95b] P. Buch, S. Lin, V. Nagasamy, and E. S. Kuh, "Techniques for fast circuit simulation applied to power estimation of CMOS circuits," *Proceedings of the ACM International Symposium on Low Power Design*, pp. 135-138, April 1995.
- [Buc96a] P. Buch, "A statistical approach to transistor level power estimation," *Technical Report, UC Berkeley, UCB/ERL M96/07*, April 1996.
- [Buc96b] P. Buch, C. Lennard, and A.R. Newton, "Guided logic synthesis for low power design," *Proc. of the SRC Techcon*, Sept. 1996.
- [Buc97a] P. Buch, and E. S. Kuh, "Symphony: A fast mixed signal simulator for BiMOS analog/digital circuits," *Proc. of the 10th IEEE International Conference on VLSI Design*, pp. 403-407, Jan. 1997.
- [Buc97b] P. Buch, C. Lennard, and A.R. Newton, "Rewiring for power optimization," *Proc. of the IEEE/ACM International Workshop on Logic Synthesis*, May 1997.
- [Buc97c] P. Buch, A. Narayan, A.R. Newton, and A. Sangiovanni-Vincentelli, "On synthesizing pass transistor networks," *Proc. of the IEEE/ACM International Workshop on Logic Synthesis*, May 1997.
- [Buc97d] P. Buch, A. Narayan, A.R. Newton, and A. Sangiovanni-Vincentelli, "Logic synthesis for large pass transistor circuits," *Proc. of the IEEE International Conference on Computer-Aided Design*, November 1997.
- [Bur93] R. Burch, F. Najm, P. Yang, and T. Trick, "A monte carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, Vol. 1, No. 1, pp. 63-71, March 1993.

- [Cam97] R. Camposano, "The quarter micron challenge: integrating physical and logic design," *Proceedings of the International Symposium on Physical Design*, p. 211, April 1997.
- [Cha75] B. R. Chawla, H. K. Gummel, and P. Kozak, "Motis - a MOS timing simulator," *IEEE Transactions on Circuits and Systems*, Vol. CAS-22, No. 12, pp. 901-909, December 1975.
- [Cha92a] A. P. Chandrakasan, S. Sheng, and R. Brodersen, "Low power CMOS digital design," *IEEE Transactions on Solid-State Circuits*, Vol. 27, No. 4, pp. 473-483, April 1992.
- [Cha92b] W. H. Chang, B. Davari, M. R. Wordeman, Y. Taur, C.C-H Hsu and M. D. Rodriguez, "A high performance 0.25  $\mu$ m CMOS technology: design and characterization," *IEEE Transactions on Electron. Devices*, Vol. 39, pp. 959-966, 1992.
- [Cha92c] Y.-H. Chang, and A. T. Yang, "Analytic macromodeling and simulation of tightly coupled mixed analog-digital circuits," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 244-247, Nov. 1992.
- [Cha94] S. C. Chang, and M. Marek-Sadowska, "Perturb and simplify: multi-level boolean network optimizer," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 2-5, Nov. 1994.
- [Cha95] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, pp. 12-31, Jan. 1995
- [Che91] K. C. Chen, Y. Matsunga, M. Fujita, and S. Muroga, "A resynthesis approach for network optimization," *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 458-463, June 1991.
- [Chi77] M. J. Chien, and E. S. Kuh, "Solving nonlinear resistive networks using piecewise linear analysis and simplicial subdivision," *IEEE Transactions on Circuits and Systems*, Vol. CAS-24, No. 6, pp. 305-317, June 1977.
- [Cir87] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 534-537, Nov. 1987.

- [Con80] S. Conte, and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill Book Company, 1980.
- [Dav78] M. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill International, 1978.
- [Den94] A.-C. Deng, "Power analysis for CMOS/BiCMOS circuits," *Proceedings of the International Workshop on Low Power Design*, 1994.
- [Des69] C. A. Desoer, and E. S. Kuh, *Basic Circuit Theory*, McGraw Hill, 1969.
- [Dho92] Y. Dhong and C. Tsang, "High speed CMOS POS PLA using pre-discharge or array and charge sharing and array," *IEEE Transactions on Circuits and Systems-II*, Vol. 39, No. 8, pp. 557-564, August 1992.
- [Fan77] S. P. Fan, M. Y. Hsueh, A. R. Newton, and D. O. Pederson, "MOTIS-C: a new circuit simulator for MOS LSI circuits," *Proceedings of the International Symposium on Circuits and Systems*, pp. 700-703, April 1977.
- [Fuj72] T. Fujisawa, and E. S. Kuh, "Piecewise linear theory of nonlinear networks," *SIAM J. Appl. Math.*, Vol. 22, pp. 307-328, March 1972.
- [Gea68] C. W. Gear, "The automatic integration of stiff ordinary differential equations," *Proceedings of the IFIP Congress*, Edinburgh, Scotland, 1968.
- [Gho92] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 253-259, June 1992.
- [Gra94] P. Gray, H.-S. Lee, J. Rabaey, C. Sodini, B. Wooley, "Challenges and opportunities in low power integrated circuit design," *SRC Publication Id S94019*, November 1994.
- [Hal64] A. Hald, *Statistical Theory with Engineering Applications*, New York, John Wiley & Sons, Inc., 1964.
- [Ham94] B. Hamman, and J.-L. Chen, "Data point selection for piecewise linear curve approximation," *Computer-Aided Geometric Design*, Vol. 11, No. 3, pp. 289-301, June

1994.

- [Ham64] J.M. Hammersley, and D.C. Handscomb, *Monte Carlo Methods*, New York, John Wiley & Sons, Inc., 1964.
- [Han96] J. P. Hansen, and M. Sekine, "Synthesis by spectral translation using Boolean decision diagrams," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 248-253, June 1996.
- [Ho75] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, Vol. CAS-22, No. 6, pp. 504-509, June 1975.
- [Hos96] R. Hossain, M. Zheng and, A. Albicki, "Reducing power dissipation in CMOS circuits by signal probability based transistor reordering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.15, No. 3, March 1996.
- [HSP92] *HSPICE Version H92 User's Manual*, Meta-Software Inc., Campbell, CA.
- [Hua90] X. Huang, V. Raghvan, and R. A. Rohrer, "Awesim: a program for the efficient analysis of linearized circuits," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 534-537, November 1990.
- [Hui90] C.M. Huizer, "Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation," *IEEE European Solid State Circuits Conference*, pp. 61-64, 1990.
- [Ima94] S. Iman, M. Pedram, "Multi-level network optimization for low power," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 372-377, Nov. 1994.
- [Ima95] S. Iman, M. Pedram, "Logic-extraction and factorization for low power," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 248-253, June 1995.
- [Jae96] A. Jaekel, G.A. Jullien, S. Bandyopadhyay, "A multilevel factorization technique for pass transistor logic," *Proceedings of the 9th International Conference on VLSI Design*, pp. 339-340, January 1996.
- [Jai96] J. Jain, A. Narayan, C. Coelho, S.P. Khatri, A. Sangiovanni-Vincentelli, R.K. Brayton,

- and M. Fujita, "Decomposition techniques for efficient ROBDD construction," *Proceedings of International Conference in Formal Methods in Computer-Aided Design*, 1996.
- [Jou87] N. P. Jouppi, "Timing analysis and performance improvement of MOS VLSI designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 6, No. 7, pp. 650-665, July 1987.
- [Kan86] S. M. Kang, "Accurate estimation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, Vol. SC-21, pp. 889-891, Oct. 1986.
- [Kat65] J. Katzenelson, "An algorithm for solving nonlinear resistive networks," *Bell System Technical Journal*, Vol. 44, No. 8, pp. 1605-1620, October 1965.
- [Keb93] U. Kebshull, and W. Rosenstiel, "Efficient graph-based computation and manipulation of functional decision diagrams," *Proceedings of the European Design Automation Conference*, pp. 278-282, February 1993.
- [Keu94] K. Keutzer, and P. Vanbekbergen, "The impact of cad on the design of low power digital circuits" In *Proceedings of the 1994 IEEE Symposium on Low Power Electronics*, pp 42-45, October, 1994.
- [Keu97] K. Keutzer, A. R. Newton, and N. Shenoy, "The future of logic synthesis and physical design in deep submicron process geometries," *Proceedings of the International Symposium on Physical Design*, pp. 218-223, April 1997.
- [Kim84] Y. H. Kim, J. Kleckner, R. Saleh, and A. R. Newton, "Electrical-logic simulation," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 7-9, November 1984.
- [Len96a] Y. H. Kim, "Elogic: a relaxation-based Switch-level Simulation Technique," *MS Project Report, University of California, Berkeley*, 1985. Also: Y. H. Kim, S. Hwang, and A. R. Newton, "Electrical logic simulation and its applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, No. 1, pp. 8-22, January 1989.

- [Kim85] J. Kleckner, R. Saleh, and A. R. Newton, "Electric consistency in schematic simulation," *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 30-34, October 1982.
- [Kuk94] Y. Kukimoto, M. Fujita, and R. K. Brayton, "A redesign technique for combinational circuit based on gate reconnections," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 632-637, Nov. 1994.
- [Kak93] M. Kakumu, "Process and device technologies of CMOS devices for low voltage operation," *IEICE Transactions on Electron.* Vol. E76-C, p. 672, 1993.
- [Lan93] P. E. Landman, and J. M. Rabaey, "Power estimation for high level synthesis", *Proceedings of EDAC-EUROASIC '93*, Paris, France, pp. 361-366, February 1993
- [Lav95] L. Lavagno, P. C. McGeer, A. Saldanha, A. Sangiovanni-Vincentelli, "Timed shannon circuits: a power efficient design style and synthesis tool," *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 254-260, June 1995.
- [Lel82] E. Lelarsmee, and A. L. Sangiovanni-Vincentelli, "Relax: a circuit simulator of large scale MOS integrated circuits," *Proceedings of the 19th ACM/IEEE Design Automation Conference*, pp. 682-690, June 1982.
- [Len95] C. K. Lennard, A. R. Newton, "An estimation technique to guide low power resynthesis algorithms," *Proceedings of the ACM International Symposium on Low Power Design*, pp. 227-232, April 1995.
- [Len96a] C. K. Lennard, A. R. Newton, "On estimation accuracy for guiding low-power resynthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 644-664, June 1996.
- [Len96b] C. K. Lennard, P. Buch, and A. R. Newton, "Logic synthesis using power sensitive don't care sets," *Proceedings of the ACM International Symposium on Low Power Electronics & Design*, August 1996.
- [Len96b] J. Lillis, and P. Buch, "Table-lookup methods for improved performance-driven routing," *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis*



*of Digital Systems*, December 1997.

- [Lin70] W. Liniger, and R. A. Willoughby, "Efficient integration methods for stiff systems of ordinary differential equations," *SIAM Journal of Numerical Analysis*, Vol. 7, pp. 47-66, March 1970.
- [Lin91] A. Linz, "A low power PLA for a signal processor," *IEEE Journal of Solid State Circuits*, Vol. 26, No. 2, February 1991, pp. 107-115.
- [Lin93a] B. Lin, and H. De Mann, "Low-power driven technology mapping under timing constraints," *Proceedings of the International Workshop on Logic Synthesis*, May 1993
- [Lin93b] S. Lin, E. S. Kuh, and M. Marek-Sadowska, "Stepwise equivalent conductance circuit simulation technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 5, pp. 672-683, May 1993.
- [Lin93c] S. Lin, "Circuit simulation for VLSI and electronic packaging design," *Ph.D. thesis, University of California, Berkeley*, 1993.
- [Lin95] C. C. Lin, K. C. Chen, S. C. Chang, and M. Marek Sadowska, "Logic synthesis for engineering change," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 647-652, June 1995.
- [McG95] P.C. McGeer, K.L. McMillan, A. Saldanha, A. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 402-407, Nov. 1995.
- [Mul54] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Transactions on Electronic Computers*, EC-3, pp. 6-12, 1954.
- [Mur89] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney, "The transduction method - design of logic networks based on permissible functions," *IEEE Transactions on Computers*, Vol. 38, No. 10, pp. 1404-1424, October 1989.
- [Nag75] L. W. Nagel, "Spice2: a computer program to simulate semiconductor circuits," *Ph.D. Dissertation*, *University of California, Berkeley*, UCB/ERL M520, May

1975.

- [Naj93] F. Najm, "Transition density, a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 2, pp. 310-323, Feb. 1993.
- [Naj94a] F. Najm, "Estimating power dissipation in VLSI circuits", *Technical Report, University of Illinois at Urbana-Champaign*, May 1994.
- [Naj94b] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 2, No. 4, pp. 446-455, Dec. 1994.
- [Nev94] J.L. Neves, and A. Albicki, "A pass transistor regular structure for implementing multi-level combinational circuits," *Proceedings of the 7th Annual IEEE International ASIC Conference and Exhibit*, pp. 88-91, 1994.
- [New75] A. R. Newton, and D. O. Pederson, "Analysis time, accuracy and memory requirement trade-offs in Spice2," *Proceedings of the 11th Annual Asilomar Conference on Circuit Systems and Computers*, pp. 6-9, 1978.
- [New79] A. R. Newton, "The simulation of large-scale integrated circuits," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26, pp.741-749, September 1979.
- [New83] A. R. Newton, and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Transactions on Electronic Devices*, Vol. ED-30, No. 9, pp. 1184-1207, September 1983.
- [Nex97] The NexSIS Group, "A next generation logic synthesis program for deep submicron design," *manuscript in preparation*, 1997.
- [Ous85] J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 4, No. 7, pp. 336-349, July 1985.
- [Pap91] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*," 3rd edition, New York, McGraw-Hill Inc., 1991.

- [Pan95] S. Panda, and F. Somenzi, "Who are the variables in your neighborhood," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 74-77, Nov. 1995.
- [Par75] K. P. Parker, and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, Vol. C-24, pp. 668-670, June 1975.
- [Pea00] K. Pearson, "On the theory of contingency and its relation to association and normal correlation," *Philos. Mag. Series*, Vol. 50, pp. 157-175, 1900.
- [Pra93] S. C. Prasad, and K. Roy, "Circuit activity driven multi-level logic optimization for low power reliable operation," *Proceedings of the European Conference on Design Automation*, pp. 368-372, Feb. 1993.
- [Rab96] J. Rabaey, *Digital Integrated Circuits*, Prentice-Hall, Inc. 1996.
- [Rad85] D. Radhakrishnan, S.R. Whitaker, and G.K. Maki, "Formal design procedures for pass transistor switching circuits," *IEEE Journal of Solid-State Circuits*, Vol. SC-20, No. 2, April 1985.
- [Ree54] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Transactions on Information Theory*, PGIT-4, pp. 38-49, 1954.
- [Ric84] J. R. Rice, *The Approximation to Functions*, Vol. 1, 2, Addison-Wesley, Reading, Mass., 1984.
- [Roh96] B. Rohfleisch, A. Kölbl, B. Wurth, "Reducing power dissipation after technology mapping by structural transformations," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 789-794, June 1996.
- [Rub83] J. Rubinstein, Jr., P. Penfield, and M. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 2, No. 7, pp. 202-211, July 1983.
- [Rud87] R. Rudell, and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 6, pp. 727-750, Sept. 1987.

- [Rud93] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 42-47, November 1993.
- [Rum89] M. Rumsey, and J. Sackett, "An ASIC methodology for mixed analog-digital simulation," *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pp. 618-621, June 1989.
- [Sak80] K. Sakallah, and S. W. Director, "An activity-directed circuit simulation algorithm," *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 1032-1035, October 1980.
- [Sak90] T. Sakurai, B. Lin, and A.R. Newton, "Multiple-output shared transistor logic (MOSTL) family synthesized using binary decision diagram," *Technical Report, University of California, Berkeley*, UCB/ERL Memo M90/21, 1990.
- [Sal95] F. Salice, "Automatic synthesis of logic functions using transmission gates," *Journal of Microelectronic Systems Integration*, Vol. 3, No.1, March 1995.
- [San68] I. W. Sandberg, and H. Shichmann, "Numerical integration of stiff non-linear differential equations," *Bell System Technical Journal*, Vol. 47, No. 4, pp. 511-527, April 1968.
- [Sar93] A. Sarabi, P. F. Ho, K. Iravani, W. R. Daasch, and M. Perkowski, "Minimal multi-level realization of switching functions based on Kronecker functional decision diagrams," *Proceedings of the International Workshop on Logic Synthesis*, May 1993.
- [Sas95] Y. Sasaki, K. Yano, S. Yamashita, H. Chikata, K. Rikino, K. Uchiyama, and K. Seki, "Multi-level pass-transistor logic for low-power ULSIs," *Proceedings of the International Symposium on Low Power Electronics*, pp. 14-15, 1995.
- [Sas93a] T. Sasao, "Logic synthesis with EXOR-gates," *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 259-285, 1993.
- [Sas93b] T. Sasao, "AND-EXOR expressions and their optimization," *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 287-312, 1993.

- [Sat90] H. Sato, Y. Yasue, F. Matsunaga, and M. Fujita, "Boolean resubstitution with permissible functions and binary decision diagrams," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 297-301, June 1990.
- [Sav90a] P. Saviz, "Circuit simulation by hierarchical, waveform relaxation," *Ph.D. Dissertation, Columbia University*, 1990.
- [Sav90b] H. Savoj, R. K. Brayton, "The use of observability and external don't cares for the simplification of multi-level networks," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 297-301, June 1990.
- [Sen92] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 328-333, Oct. 1992.
- [Set86] S. Seth, B. Bhattacharya, V. D. Agrawal, "An exact analysis for efficient computation of random-pattern testability in combinational circuits," *Proc. of the 16th International Symposium on Fault Tolerant Computing*, pp. 318-323, July 1986.
- [Sha95] M. Shamanna, K. Cameron, and S.R. Whitaker, "Multiple-input, multiple-output pass transistor logic," *International Journal of Electronics*, Vol. 79, No. 1, pp. 33-45, 1995.
- [Sha38] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Transactions of AIEE*, Vol. 57, pp. 713-723, 1938.
- [She92] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 402-407, Nov. 1992.
- [Shi93] Y.-H. Shih, Y. Leblebici, and S.-M. Kang, "Illiads: a fast timing and reliability simulator for digital MOS circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 9, pp. 1387-1402, September 1993.
- [Shi94] T. Shiple, R. Hojati, A. Sangiovanni-Vincentelli, R.K. Brayton, "Heuristic minimization of BDDs using don't cares," *Proceedings of the 31st ACM/IEEE Design Automation Conference*, pp. 225-231, June 1994.

- [Ste95] B. Steinbach, and A. Wereszczynski, "Synthesis of multi-level circuits using EXOR-gates," *Proceedings of the IFIP WG 10.5 Workshop on Reed-Muller Expansion in Circuit Design*, pp. 161-168, August 1995.
- [Tau95] Taur, et al., *IBM Journal of Research and Development*, Vol. 39, p. 245, January 1995.  
(Also <http://www.research.ibm.com/0.1um>)
- [Ter83] C. J. Terman, "Rsim - a logic-level timing simulator," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 437-440, November 1983.
- [Tha96] S. Thakur, D. F. Wong, and S. Krishnamoorthy, "Delay minimal decomposition of multiplexers in technology mapping," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 254-257, June 1996.
- [Tiw93] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power," *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 74-79, June 1993.
- [Top94] S. Topcu, O. Ocah, A. Atalar, and M. A. Tan, "A novel algorithm for DC analysis of piecewise linear circuits: Popcorn," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 41, No. 8, pp. 553-556, August 1994.
- [Tsa94] C.-C. Tsai, and M. Marek-Sadowska, "Minimization of fixed-polarity AND/XOR canonical networks," *Proceedings of IEE*, Vol. 141, Pt. E, No. 6, pp. 369-374, November 1994.
- [Tsa96] C.-C. Tsai, and M. Marek-Sadowska, "Multi-level logic synthesis for arithmetic functions," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 242-247, June 1996.
- [Tsu93a] C.-Y. Tsui, M. Pedram, and A. Despain, "Technology decomposition and mapping targeting low power dissipation," *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 68-73, June 1993.
- [Tsu93b] C.-Y. Tsui, M. Pedram, and A. Despain, "Efficient estimation of dynamic power consumption under a real delay model," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 224-228, Nov. 1993.

- [Var69] J. Varga, *Matrix Iterative Analysis*, Englewood Cliffs, NJ, Prentice-Hall, 1969.
- [Vee84] H. J. M. Veendrick, "Short-circuit power dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, pp. 468-473, August 1984.
- [Vid86] L. Vidigal, S. Nassif, and S. Director, "Cinnamon: coupled integration and nodal analysis of MOS networks," *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pp. 79-185, 1986.
- [Vla82] A. Vladimirescu, and D. O. Pederson, "Performance limits of the Classic circuit simulation program," *Proceedings of the International Symposium on Circuits and Systems*, Vol. 3, pp. 1229-1232, May 1982.
- [Vis91] C. Visweswariah, and R. A. Rohrer, "Piecewise approximate circuit simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 7, pp. 861-870, July 1991.
- [Wat89] R. K. Watts, ed., *Submicron Integrated Circuits*, John Wiley & Sons, New York, 1989.
- [Wee73] W. T. Weeks, A. J. Jiminez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for Astap - a network analysis program," *IEEE Transactions on Circuit Theory*, Vol. CT-20, No. 11, pp. 628-634, November 1973.
- [Yam94] K. Yamamura, "Piecewise linear analysis of non-linear resistive networks containing Gummel-Poon models or Shichmann-Hodges Models," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E77-A, No. 1, pp. 309-316, January 1994
- [Yan80] P. Yang, I. N. Hajj, and T. N. Trick, "SLATE: a circuit simulation program with latency exploitation and node tearing," *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 353-355, October 1980.
- [Yan90] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns CMOS 16x16-b multiplier using complementary pass-transistor logic," *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 2, pp. 388-395, April 1990.

- [Yan96] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-down pass-transistor logic design," *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 6, pp. 792-803, June 1996.
- [Yu94] M. L. Yu, and B. D. Ackland, "VLSI timing simulation with selective dynamic regionization," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 195-199, November 1994.