ABSTRACT

Title of dissertation: Design Space Re-Engineering for Power Minimization in Modern Embedded Systems

Lin Yuan, Doctor of Philosophy, 2006

Dissertation directed by: Professor Gang Qu
Department of Electrical and Computer Engineering

Power minimization is a critical challenge for modern embedded system design. Recently, due to the rapid increase of system's complexity and the power density, there is a growing need for power control techniques at various design levels. Meanwhile, due to technology scaling, leakage power has become a significant part of power dissipation in the CMOS circuits and new techniques are needed to reduce leakage power. As a result, many new power minimization techniques have been proposed such as voltage island, gate sizing, multiple supply and threshold voltage, power gating and input vector control, etc. These design options further enlarge the design space and make it prohibitively expensive to explore for the most energy efficient design solution.

Consequently, heuristic algorithms and randomized algorithms are frequently used to explore the design space, seeking sub-optimal solutions to meet the time-to-market requirements. These algorithms are based on the idea of truncating the design space and restricting the search in a subset of the original design space.

While this approach can effectively reduce the runtime of searching, it may also exclude high-quality design solutions and cause design quality degradation. When the solution to one problem is used as the base for another problem, such solution quality degradation will accumulate. In modern electronics system design, when several such algorithms are used in series to solve problems in different design levels, the final solution can be far off the optimal one.

In my Ph.D. work, I develop a *re-engineering* methodology to facilitate exploring the design space of power efficient embedded systems design. The direct goal is to enhance the performance of existing low power techniques. The methodology is based on the idea that design quality can be improved via iterative "re-shaping" the design space based on the "bad" structure in the obtained design solutions; the searching run-time can be reduced by the guidance from previous exploration. This approach can be described in three phases: (1) apply the existing techniques to obtain a sub-optimal solution; (2) analyze the solution and expand the design space accordingly; and (3) re-apply the technique to re-explore the enlarged design space.

We apply this methodology at different levels of embedded system design to minimize power: (i) switching power reduction in sequential logic synthesis; (ii) gate-level static leakage current reduction; (iii) dual threshold voltage CMOS circuits design; and (iv) system-level energy-efficient detection scheme for wireless sensor networks. An extensive amount of experiments have been conducted and the results have shown that this methodology can effectively enhance the power efficiency of the existing embedded system design flows with very little overhead.

DESIGN SPACE RE-ENGINEERING FOR POWER MINIMIZATION IN MODERN EMBEDDED SYSTEMS

by

Lin Yuan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:
Professor Gang Qu, Chair/Advisor
Professor Shuvra S. Bhattacharyya
Professor Manoj Franklin
Professor Dennis M. Healy
Professor Martin Peckerar
Professor Ankur Srivastava

DEDICATION

To my parents and friends

ACKNOWLEDGMENTS

First and foremost I'd like to thank my advisor, Professor Gang Qu for his invaluable guidance and never-ending help in research, teaching and all other aspects. As an advisor and a friend, he has been a perfect role model for me in the past five years. From him, I learn not only how to be a prolific researcher, but also how to be a nice person, a great mentor, a supportive friend, and an affectionate father. It is great pleasure of me to work with and learn from such an extraordinary person.

I would like to thank Professor Shuvra Bhattacharyya, with whom I have been a teaching assistant and have worked on a research project which lead to a published paper. His unique vision in research and abundant experience in teaching have been a great source of guidance for me. I would like to thank Professor Manoj Franklin, Professor Dennis Healy, Professor Martin Peckerar, and Professor Ankur Srivastava for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript. I would like to thank Professor Ankur Srivastava for his always available help, advice or simply a word of encouragement.

My colleagues at the Embedded System Research Lab are the nicest and most supportive. Ming-Yung Ko, who is the lab administrator, did a wonderful job and helped me get familiar with lab environments and computer systems. Vida Kianzad, who is like an elder sister to me, always encouraged me when I was depressed. I also enjoyed many inspiring discussions with Neal Bambha, Dong-Ik Ko, Sankalita

Saha, Mainak Sen and Ankush Varma.

My PhD years would not have been so memorable without my friends. Pushkin Pari, Sadagopan Srinivasan, and Aditya Kalyanpur are both my labmates and my best buddies. They are really good friends and fun people. I owe special thanks to Honghao Ji and Ji Luo for their enormous help and being very considerate to me. Many thanks are due to all my old and new friends in every corner of this globe; they made my life full of exciting stories.

I feel extremely happy to have amazingly nice roommates, Wei Jing, Haibin Ling, and Mei Zhang. They are like my families and I don't even want to miss a single moment with them.

I would also like to acknowledge help and supports from the staff members at ECE Help Desk, Clifford Russell, Tarjia Johnson and Jeff McKinney. Their professional technical supports have made my work efficient and smooth.

At last, I owe my deepest thanks to my parents who have always stood by me and guided me through my career. I am really proud to have such great parents.

TABLE OF CONTENTS

List of Tables viii

List of Figures ix

1 Introduction 1
1.1 Low Power Design Challenge in Modern Embedded Systems 1
1.2 Power Minimization in Embedded System Design Flow 6
1.3 Design Space Exploration and Re-Engineering 8
1.3.1 Design Space Exploration 8
1.3.2 Design Space Re-Engineering 10
1.4 Thesis Organization 11

2 General Re-Engineering Methodology and Framework 12
2.1 A New Design Framework for Low Power 12
2.2 Re-Constructing the Design Space 14
2.3 Application 15
2.3.1 Sequential Logic Synthesis (Chapter 3) 15
2.3.2 Dual-V_{th} CMOS Circuit Design (Chapter 4) 15
2.3.3 Input Vector Control for Static Power Reduction (Chapter 5) 16
2.3.4 Energy-Efficient Detection Scheme for Wireless Sensor Networks (Chapter 6) 17
2.4 Summary 17

3 Power-Driven Sequential Logic Synthesis 18
3.1 Introduction 18
3.1.1 A Motivational Example 20
3.1.2 FSM Re-Engineering 21
3.2 Related Work 23
3.3 Preliminary 25
3.3.1 An Example of Re-constructing FSMs 27
3.4 Power-Driven FSM Re-Engineering Approach 28
3.5 FSM Re-engineering Algorithm 28
3.5.1 A Generic Approach 29
3.5.2 Genetic Algorithm Based State Duplication 31
3.5.3 Heuristic on State Selection for Duplication 34
3.5.4 Heuristic on How to Duplicate a Selected State 35
3.5.5 Determine the Minimum Switching Activity 37
3.6 Experimental Results 39
3.7 Summary 44

4 Dual-V_{th} CMOS Circuit Design for Leakage Reduction 47
4.1 Introduction . . . 47
4.1.1 A Motivational Example . . . 49
4.1.2 Main Idea and Contribution . . . 51
4.1.3 Chapter Organization . . . 52
4.2 Related Work . . . 53
4.3 Simultaneous Dual V_t Assignment and Input Vector Selection . . . 55
4.3.1 Dual V_t Assignment . . . 55
4.3.2 Input Vector Selection . . . 56
4.3.3 Combining Dual V_t Assignment and Input Vector Selection . . . 58
4.3.4 Algorithm Description and Analysis . . . 60
4.4 Experimental Results . . . 63
4.5 Summary . . . 67

5 Gate-Level Input Vector Control for Static Power Minimization 69
5.1 Introduction . . . 69
5.2 Related Work . . . 73
5.3 Leakage Reduction by Gate Replacement . . . 76
5.3.1 Basic Gate Replacement Technique . . . 77
5.3.2 A Fast Gate Replacement Algorithm . . . 80
5.4 Solving the MLV+ Problem . . . 83
5.4.1 NP-Completeness of the MLV Problem . . . 83
5.4.2 The MLV+ Problem and Outline of the Divide-and-Conquer Approach . . . 85
5.4.3 Finding the Optimal MLV for Tree Circuits . . . 86
5.4.4 Connecting the Tree Circuits . . . 90
5.4.5 Overhead Analysis . . . 93
5.5 Experimental Results . . . 94
5.6 Summary . . . 101

6 Energy Efficient Detection Scheme for Wireless Sensor Network Design 103
6.1 Introduction . . . 103
6.2 Related Work . . . 106
6.2.1 On Detection in Wireless Sensor Networks . . . 106
6.2.2 On Energy Efficiency in Sensor Network Design . . . 107
6.3 System Model . . . 108
6.4 Hybrid Detection Scheme . . . 112
6.4.1 Intuition . . . 112
6.4.2 Detection mechanism . . . 113
6.4.3 Decision rules . . . 114
6.4.4 Suboptimal algorithm . . . 118
6.5 Energy Consumption Model . . . 119
6.5.1 Data acquisition . . . 119
6.5.2 Data processing . . . 120
6.5.3 Communication . . . 121

6.6 Simulation Results 123
6.7 Summary 128

7 Conclusions 130

A List of Publications 132

Bibliography 135

LIST OF TABLES

3.1 Total switching activity reduction on re-constructed FSMs. 42

3.2 Area and power comparison between original FSM and reconstructed FSM 44

4.1 Leakage current (nA) in high-V_{th} (0.48V) and low-V_{th} (0.33V) two-input NAND gates at different inputs from SPICE simulations. 48

4.2 Leakage current (nA) in the library gates. 63

4.3 Propagation delay (ns) in the library gates. 63

4.4 Comparison of individual MLV and V_{th} assignment with simultaneous MLV and V_{th} assignment algorithm on MCNC benchmarks in terms of runtime and leakage. The red(%) column reports the reduction over the combined serial randome MLV search and V_{th} assignment. . . 64

4.5 Number of gates in the worst leakage state (# WLS), number of WLS gates at high V_{th} (# WLS*), and the total number of gates assigned high V_{th} (# V_{th}H) in MCNC circuits with serial dual-V_{th} assignment and with simultaneous dual-V_{th} assignment and input vector control. 65

4.6 Comparison of individual MLV and V_{th} assignment with simultaneous MLV and V_{th} assignment algorithm on ISCAS benchmarks in terms of runtime and leakage. The red(%) column reports the reduction over the combined serial randome MLV search and V_{th} assignment. . . 66

4.7 Number of gates in the worst leakage state (# WLS), number of WLS gates at high V_{th} (# WLS*), and the total number of gates assigned high V_{th} (# V_{th}H) in ISCAS circuits with serial dual-V_{th} assignment and with simultaneous dual-V_{th} assignment and input vector control. 67

5.1 Results on 26 small circuits with 22 or less primary inputs. 96

5.2 Results on 43 large circuits with primary inputs more than 22. 98

5.3 The percent of WLS gates in 43 circuits and the area increase with different input vector control algorithms. 100

5.4 Average performance comparison with algorithm in [1]. 101

LIST OF FIGURES

1.1 Number of transistors on chip is doubled every 18 months as predicted by Gordon Moore. 3

1.2 Trend of increasing power density. 4

1.3 Trend of increasingly significant leakage power. 5

1.4 Serial design space exploration strategy in the conventional design flow. 8

2.1 Re-engineering design framework. 13

3.1 A 5-state FSM and a functionally equivalent 6-state FSM. 20

3.2 Re-constructing an FSM by duplicating a state S. 28

3.3 FSM re-engineering for low power state encoding. 29

3.4 Pseudocode: State duplication via genetic algorithm 33

3.5 Pseudocode: Duplicate a State 38

3.6 Switching activity of POW3's encoding schemes on the original and re-constructed FSMs and the optimal encoding (Opt) on the new FSMs. Normalized to the optimal encoding on the original FSMs. . . 45

4.1 Dual-V_{th} assignment for circuit C17 and its impact on leakage reduction. 49

4.2 Pseudo-code of the simultaneous dual V_t assign and input vector selection algorithm. 60

5.1 Leakage current of (a)INVERTER, (b)NAND2 and (c)NAND3. Data obtained by simulation in Cadence Spectre using 0.18 μm process. . . 70

5.2 A motivation example for gate replacement. 71

5.3 Gate replacement and the consequence to its fanout gate. 78

5.4 Pseudo-code of the gate replacement algorithm. 81

5.5 Illustration for the proof of the NP-completeness of the MLV problem. 84

5.6 Dynamic programming to find optimal MLV in a tree circuit. 87

5.7 MLV in a circuit before and after gate replacement 90

5.8 Resolving the conflict in connecting tree circuits. 91

5.9 Leakage and WLS percentage on 43 large circuits with 22 PIs or more. X-axis lists benchmarks sorted by leakage current in divide-and-conquer approach; Y-axis shows percentage of leakage and WLS gates. 97

6.1 Wireless Sensor Network for Detection 109

6.2 Comparison of Three Schemes in Detection Accuracy 118

6.3 Dense Network: 1×1 Field 123

6.4 Intermediate Network: $\sqrt{2} \times \sqrt{2}$ Field 124

6.5 Sparse Network: 2×2 Field 125

6.6 Breakdown of Energy Consumption for Dense Network 127

6.7 Comparison of Energy per Node for Dense Network 128

Chapter 1

Introduction

1.1 Low Power Design Challenge in Modern Embedded Systems

With the advances of transistor integration capability and System-on-Chip (SoC) design technology, modern embedded systems can be implemented on a tiny silicon chip. For example, the wireless sensor developed at Berkeley is in the size of a nickel, yet it integrates almost a million transistors on chip [111]. This trend of technology scaling makes it possible for designers to implement a sophisticated embedded system on small and portable device and these portable embedded systems have ever become more and more popular in today's market such as cell phones, personal digital assistants (PDA), MP3 players, digital cameras, and medical sensors etc.

One of the major challenges for modern embedded system design is the power efficiency. Most portable devices are sustained by batteries; in many cases, frequently recharging the batteries are not possible or convenient. Although a substantial improvements have been made in battery technology, the increase in battery capacity can not keep pace with the rapid increase of power requirements. At the same time, the performance of embedded systems have improved dramatically and they are burning more and more power. For example, the recent embedded processors developed at Freescale are running at a clock frequency of 3 GHz [113]; the

Transmeta Crusoe processor has a maximum frequency of 1 GHz [114]. Therefore, reducing power consumption in the embedded system has been deemed as a crucial approach to extend the life-time of embedded systems.

The other reason that drives the low-power design solution for embedded systems is the continuing transistor technology scaling, which follows the famous Moore's Law [79]. Figure 1.1 shows the trend of the number of transistors integrated in microprocessors. In accordance to this trend is the power density increase on chip as shown in Figure 1.2. As more and more transistors being integrated on the chip, the power density is increasing dramatically, which will not only shorten the life-time of the system, but also cause high junction temperature that may trigger hardware failure and performance degradation.

There are two main sources of power dissipation in embedded systems: **dynamic power** and **static power**.

Dynamic power is caused by the capacitance charging and discharging in the circuits. It can be described by the equation:

$$P_{dyn} = \alpha C_L \cdot V_{dd}^2 \cdot f \tag{1.1}$$

where α is the switching activity; C_L is the effective loading capacitance of the circuit; V_{dd} is the supply voltage; and f is the clock frequency.

Static power is mainly contributed by the leakage current flowing in the CMOS circuit when it is at standby mode (there are also leakage currents even when the circuit is switching):

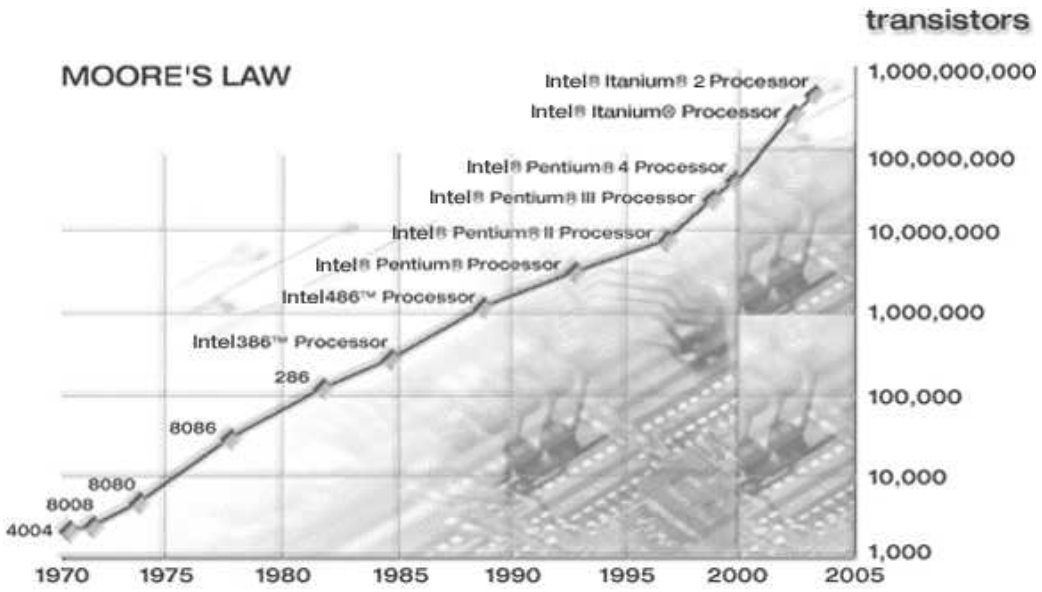


Figure 1.1: Number of transistors on chip is doubled every 18 months as predicted by Gordon Moore.

$$P_{stat} = V_{dd} \cdot I_{leakage} \tag{1.2}$$

where the leakage current $I_{leakage}$ consists of gate leakage and subthreshold leakage. Based on the BSIM3 MOS transistor model [112], the subthreshold leakage current of a MOSFET can be modeled as:

$$I_{sub} = Ae^{\frac{q}{n'kT}(V_G - V_S - V_{TH_0} - \gamma' V_s + \eta V_{DS})}(1 - e^{\frac{-qV_{DS}}{kT}}) \tag{1.3}$$

where $A = \mu_0 C_{ox} \frac{W_{eff}}{L_{eff}} (\frac{kT}{q})^2 e^{1.8}$; C_{ox} is the gate oxide capacitance per unit area; μ_0 is the zero bias mobility; n' is the subthreshold swing coefficient of the transistor; V_{TH_0} is the zero bias threshold voltage. The gate leakage is only a small portion of the

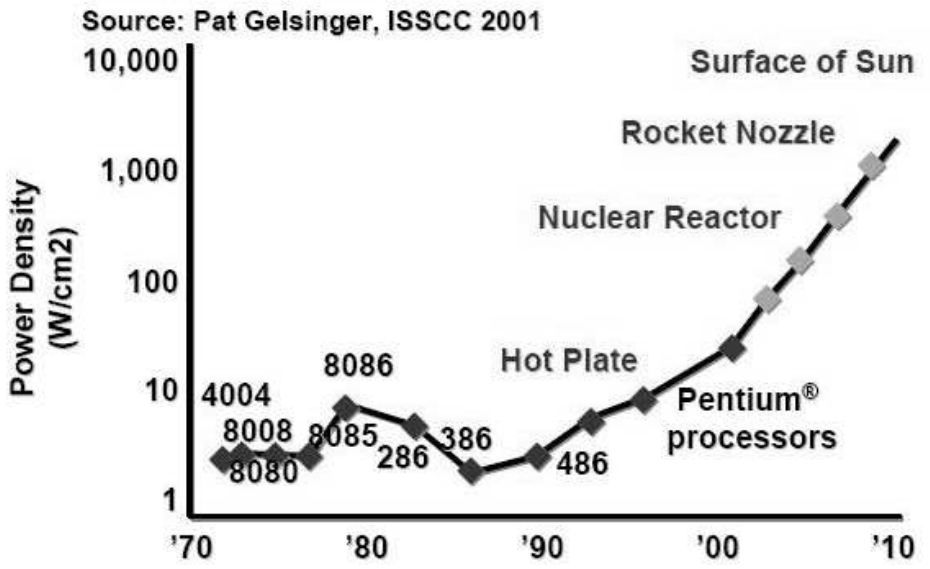


Figure 1.2: Trend of increasing power density.

total leakage and is projected to be controlled by high-K material [45]. Therefore, leakage reduction techniques are mainly focused on minimizing the subthreshold leakage.

Traditionally, dynamic power consumption is the dominant part and many low power techniques have been proposed to reduce it. One of the most popular approaches for dynamic power reduction is voltage scaling. This is based on the quadratic dependence of dynamic power on supply voltage. However, the circuit delay is inversely proportional to supply voltage as shown in Equation (1.4):

$$delay \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{1.4}$$

where α is between 1.0 and 1.2. As V_{dd} is scaled down, the performance of the

system will also decrease. In order to meet the performance requirements, threshold voltage V_t also has to be scaled down. However, the reduction of threshold causes exponential increase in subthreshold leakage as shown in equation (1.3.

Due to this reason, leakage power has become a significant part of power consumption in today's embedded system. Figure 1.1 shows that leakage is going to be the dominant source of power dissipation in 65nm technology node and beyond. Therefore, a holistic approach is needed to minimizing the total power in the circuit. This enlarges the design space of power minimization in embedded system and makes this problem even more complicated.

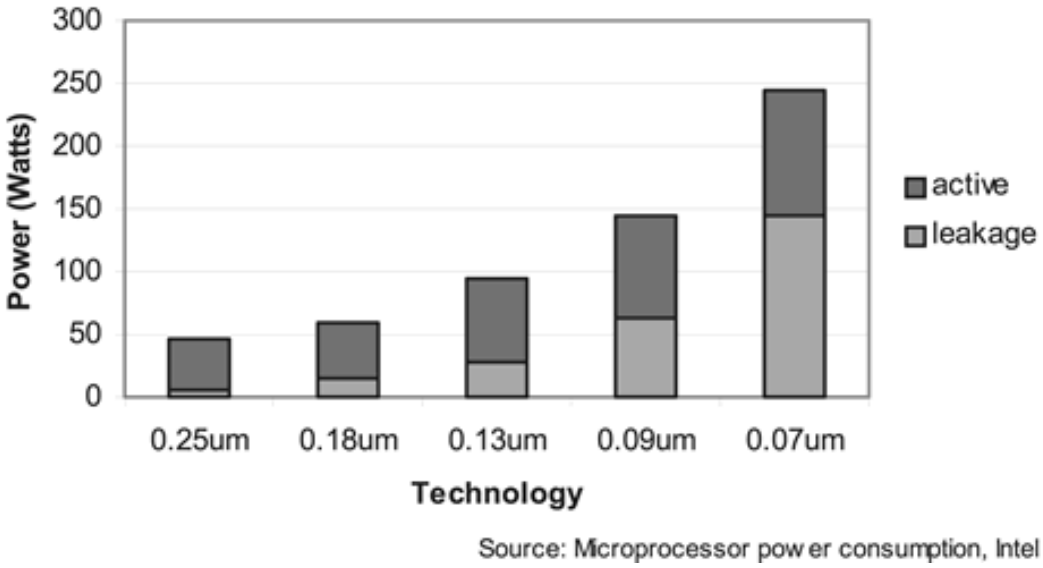


Figure 1.3: Trend of increasingly significant leakage power.

1.2 Power Minimization in Embedded System Design Flow

In order to effectively minimize power, power reduction techniques are carried out at each level of the system design flow:

System level: Dynamic voltage scaling (DVS) is one of the most effective techniques at system level. It employs an operating system (OS)-supported voltage scheduler. Based on the workload and tasks' deadlines, the scheduler scales the voltage to a pre-determined level at run-time such that each task can complete before its deadline and the overall dynamic power is minimized [7, 36, 37, 38, 49, 73, 74, 83, 104]. Recently, due to the increasing significance of leakage power, leakage-aware DVS algorithms have been proposed to minimize the sum of dynamic and leakage power [41, 56, 108]. In addition to DVS, multiple supply voltages can also applied statically to each functional blocks of the system [17, 91]. IBM has proposed a voltage-island solution for System-on-Chip (SoC) design. Each block is powered by a different voltage source depending on its performance requirement [54].

Module level: At module level, the goal of power minimization is to reduce power consumption in functional-units and memory modules [23]. Power-aware synthesis algorithms have been proposed in the following procedures: resource allocation (deciding the numbers and types of functional units and registers available for synthesis) and assignment (binding an operation to a specific instance of a functional unit) [109], functional-unit selection (selection of a functional-unit type to implement an operation) [27], and scheduling (determining the cycle-by-cycle behavior of a circuit by assigning operations to control steps) [19]. The basic idea of these algorithms is

to (i) reduce the switching activities of functional-units when they are not on critical paths. (ii) turn off modules whenever they are not producing useful outputs.

Gate level: Gate level power minimization are conducted through logic synthesis procedures, with the optimization objective redefined to be power consumption. For dynamic power minimization, the synthesis algorithms are targeted at reducing the switching activities at the fanins and fanouts of logic gates. These techniques include finite state machine (FSM) minimization and state encoding [8, 47, 102, 105], boolean multi- and two-level logic optimization [14, 40], technology mapping [89], precomputation logic [2], and retiming etc. For leakage power minimization, the algorithms include power gating, FSM decomposition [24] , and input vector control [18, 106] etc.

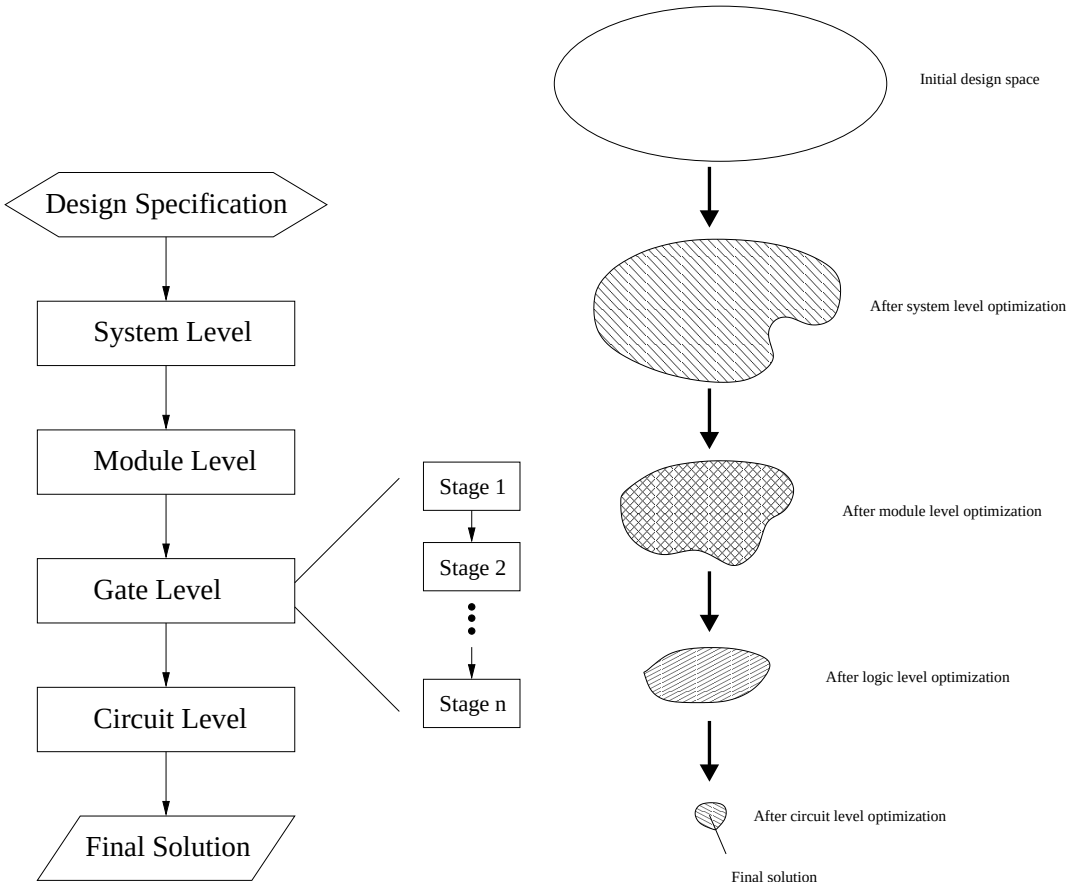
Circuit level: At circuit-level , power reduction can be further achieved by changing the characteristics of transistors and structure of the circuits. For example, one of the approaches to reduce dynamic power is transistor sizing. By choosing the optimal size for a subset of transistor, the overall power consumption in the circuit can be minimized [10]. In addition, more effective leakage power minimization techniques can be applied at circuit level. For example, multiple-V_{th} CMOS technique inserts sleep transistors with large threshold voltage at either the V_{dd} or GND to reduce the leakage current at standby mode [4, 12]; dual-V_{th} assigns transistors on critical and non-critical paths different threshold voltage [96, 94]; reverse body bias connect the gate with substrate of a MOSFET to reduce gate leakage [97].

As one can see, there are myriad options for power minimization even at one

level of the design. Achieving a global optimal solution across various levels in the design flow is not possible. In fact, solution from one design level is often used as the entry point for the design at its next level. Next subsection elucidates the design space exploration strategy in this design flow and introduce our methodology.

1.3 Design Space Exploration and Re-Engineering

1.3.1 Design Space Exploration



(a) Embedded system design flow. (b) Design space exploration.

Figure 1.4: Serial design space exploration strategy in the conventional design flow.

Due to the vast design space for power minimization in embedded system, the exploration methodology usually follows a serial strategy. That is, even within one design level, the optimization is performed in consecutive stages; the solution from one stage is used as the base for the next stage. This strategy is shown in Figure 1.4(a). For example, in power-driven sequential synthesis, FSM minimization is performed first; then the minimized FSM is encoded using power-drive state encoding algorithm.

If we view this process from a design space point of view, we can see that the design space shrinks as the optimization proceeds and eventually, the solution will be chosen from a restricted solution pools. This is shown in Figure 1.4(b).

In each design level, the design space is large and finding the optimal solution is usually a NP-hard problem [26]. Therefore, many heuristic and randomized algorithms are used to explore the design space. For example, a greedy algorithm is often used to solve the binary covering problem in logic synthesis [30]; in high-level synthesis, genetic algorithm is frequently used for resource allocation and binding [34]. Comparing to a complete search in the original design space, applying these algorithms in a serial fashion is much more efficient in truncating the design space and finding the design solutions fast.

However, this approach often removes good design solutions without enough caution. It becomes much worse when such truncation of good solutions happens in the early design stage because the design space from the earlier stage will be used as the initial space for later stages. In this case, the solutions obtained after the last stage may be far off the optimal ones in the global design space. For example,

in sequential logic optimization level, a conventional optimization procedure is to first minimize the number of states in FSMs followed by state encoding algorithms. However, as it has been pointed out in [33], this serial optimization strategy may result in inferior solutions. Based on our experimental results, the solution can be 17% worse than the optimal ones in the non-minimized FSMs.

1.3.2 Design Space Re-Engineering

In my Ph.D. work, I propose a re-engineering approach to explore the design space efficiently and effectively at the same time. Our goal is to enhance the performance of the existing low power techniques. Our approach can be described in three phases: (1) apply the existing technique to obtain a sub-optimal solution; (2) analyze this solution and expand the design space accordingly; and (3) re-apply the technique to re-explore the enlarged design space.

The novelty of our approach is in the second phase when we re-construct the design space. Particularly, we start with the analysis of the solution obtained by the existing technique. We first evaluate the solution based on a cost function that models the design objective. Then we study which part and what structure of the current solution contribute most to the cost. Next, we re-construct the design space such that the new solutions will not have such part or structure.

As the design space expansion is directed intelligently based on the analysis of the previous solution, this approach not only provides the potential of finding higher quality solutions, but also makes the design space re-exploration efficient.

Therefore, it opens doors for further optimization with relatively small run-time overhead.

1.4 Thesis Organization

The remainder of this thesis is organized in the following way. In Chapter 2, I demonstrate a general framework on the re-engineering methodology. Then I apply this methodology to four low power design problems: the power-driven sequential synthesis (Chapter 3); dual-V_{th} CMOS circuit design for leakage reduction (Chapter 4); gate-level input vector control for static power reduction (Chapter 5); and energy efficient wireless sensor network design (Chapter 6). I conclude my thesis in Chapter 7.

Chapter 2

General Re-Engineering Methodology and Framework

2.1 A New Design Framework for Low Power

In this chapter, I will elaborate the *re-engineering* methodology and illustrate the design framework using this methodology. Our focus is on power minimization; however, this methodology can be also applied to solve other optimization problems.

Given a problem $\mathcal{P}$, the design space $\mathcal{S}$ consists of all the solutions to $\mathcal{P}$ that satisfy certain design constraints. A cost C is defined for each solution with respect to an objective (e.g. power consumption). The optimization process aims to find the solutions which have the least cost based on the optimization objective. Current synthesis algorithms or tools achieve this goal via step-by-step design space truncating; each step will reduce the design space $\mathcal{S}$ and the output is used as the initial design space for the next step. The re-engineering methodology is based on the idea that some global optimal solutions may be lost during the serial design space constraining process. The word *re-engineering* means that it iteratively enlarge the design space and re-explore it seeking for better design solutions in the optimization. In the power minimization scenario, the solutions that consumes the least power are desired.

The *re-engineering* design framework is illustrated in Figure 2.1. This three-phase approach can be used to improve the performance of power minimization

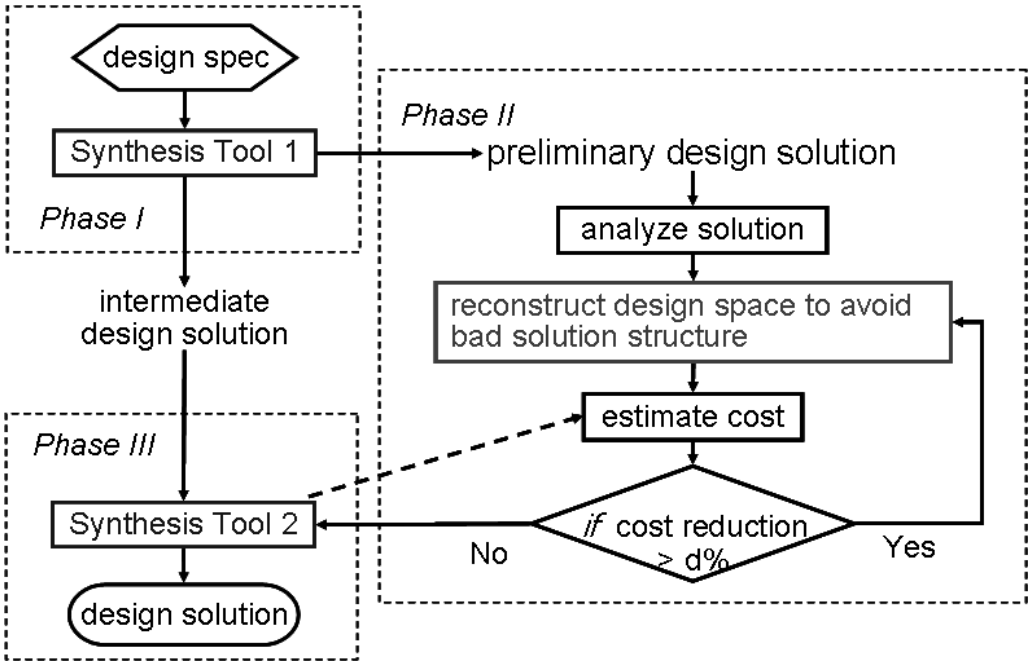


Figure 2.1: Re-engineering design framework.

design tools. First, an existing design synthesis tool is applied on the problem to obtain an "optimal" solution (i.e. the best based on the tool we use). The second phase is re-constructing the design space based on this synthesis solution. In the third and last phase, the re-constructed design space is re-explored for a new power efficient design solution.

2.2 Re-Constructing the Design Space

The key part of the re-engineering methodology is how to re-construct the design space. Initially, we start with solution analysis where we evaluate the solution based on a given cost function. Such cost function is determined based on the abstraction level of the optimization objective. First of all, such cost function must be able to accurately reflect the optimization objective; second, it must be easy to calculate. For example, total switching activity is often used as a cost function for dynamic power minimization.

Then we identify the particular structure of the current solution that causes a large cost. In fact, the solution structure is an output of the previous exploration algorithm "truncating" the design space. During the exploration process, a certain solution structure will be used as criteria for design space truncating. For example, state minimization, a step in the sequential FSM synthesis, will always keep the set of FSMs with the minimal number of states and exclude the others.

Hence, in order to remove the "bad" solution structure that results in large cost, we need to reverse the previous synthesis steps and revise the current design space. That is, we will enlarge the current design space in a way to include certain solutions that may help to reduce the cost. Note that however, when enlarging the design space, we do not expand it arbitrarily; instead, we still rely on the outputs from the previous exploration and revisit those parts where better solutions may lie.

After the design space is re-constructed, a re-exploration is conducted on the new space in Phase III. This exploration can use the same algorithm as in the first

phase; it can also be a new exploration based on the re-constructing process in Phase II.

2.3 Application

In the rest of this thesis, I will apply this framework to several power minimization problems at different levels of the design. The basic idea of solving these problems with re-engineering methodology is described below:

2.3.1 Sequential Logic Synthesis (Chapter 3)

This problem is at the gate level of the design. The serial steps are state minimization followed by state encoding. The state minimization step removes equivalent and/or compatible states in the original FSMs and the design space for state encoding is restricted to minimized FSMs. Based on the observation made in [33] we take one step back by introducing redundant states in the minimized FSM. We found that adding the redundant states may help the state encoding tool to find a better design solution with smaller cost (total switching activity in this case).

2.3.2 Dual-V_{th} CMOS Circuit Design (Chapter 4)

The problem in dual-V_{th} CMOS circuit design is to assign a high and low V_{th} values to transistors in the circuit in order to minimize the leakage. Conventionally, such process is performed after an input vector is determined for the primary inputs of the circuit. However, the input vectors also affect the leakage currents in the

circuit and even the best V_{th} assignment solution based on a given input vector may not be as good as the one based on another input vector. In this problem, we propose an iterative algorithm to find the best V_{th} assignment and input vector simultaneously.

2.3.3 Input Vector Control for Static Power Reduction (Chapter 5)

Technology mapping in logic synthesis are often targeted at reducing dynamic power and/or improving performance. Leakage has not been considered. However, the leakage currents in different CMOS gates are quite different. Meanwhile, at standby mode, leakage power can be minimized by choosing a particular input vector to the primary inputs. Such input vector is often chosen based on an already mapped circuit. Following the re-engineering framework, we first obtain a sub-optimal MLV solution by a heuristic algorithm. Based on this input vector, we check the inputs to each logic gate in the circuit and find the ones that result in the largest leakage current. Then we replace these worst-leakage-state gates by another gate in the library, such that the output function of the circuit remains the same, while less leakage currents are generated. With the gate replacement, the input vector space is re-explored to find the new MLV in the modified circuits.

2.3.4 Energy-Efficient Detection Scheme for Wireless Sensor Networks (Chapter 6)

Several detection schemes are available for wireless sensor network design at system level. However, once a detection scheme is fixed, the power reduction at this level is limited. We propose a hybrid detection scheme that can trade off energy with detection accuracy. Based on the different accuracy requirements of different applications, an more flexible and more energy efficient system can be built for the sensor networks.

2.4 Summary

In this chapter, I demonstrate the general design flow with *re-engineering* methodology. Four low power design problems are described as examples to apply this methodology. More detailed description of each problem and our solutions are provided in the rest chapters of this thesis.

Chapter 3

Power-Driven Sequential Logic Synthesis

3.1 Introduction

Finite state machine (FSM) is the most commonly used model for microcontroller design in embedded systems. Logic synthesis, which has the goal of converting the symbolic description of the FSM to a hardware implementation, traditionally starts with FSM state minimization and state encoding in order to optimize design objectives such as area, delay, and testability. For example, De Micheli et al. [63] formulate the minimum area state encoding problem as generating a minimum (multi-valued) symbolic cover of the FSM and propose a heuristic row encoding technique in [64]. Villa et al. [93] use the notion of face-posets to tackle this problem and propose a state encoding technique for two-level implementation. State encoding techniques for multi-level logic minimization have been studied in [20] and [60] where the goal is to reduce the number of literals in the Boolean output and next-state functions.

With the increasing popularity of portable computing and personal communication applications, power dissipation has become critical in the design of sequential circuits. Hence, low power state encoding techniques were proposed in accordance with the design focus shifting to low power.

In light of the well-known fact that digital CMOS circuit's power dissipation

is proportional to the switching activity, state encoding is then re-formulated to minimize the number of state bit switches per transition for low power FSM synthesis. This problem is NP-hard and many heuristic algorithms have been proposed mainly based on the idea of assigning codes with small Hamming distance to pairs of states that have a high transition probability. Such techniques include state encoding with minimal code length [8, 78, 90], non-minimal code length [59, 68] and variable code length [88]; state re-encoding approaches [28, 92] and techniques that try to minimize power and area simultaneously [48, 69].

However, these work all start with the minimized FSM and seek for the best encoding for the existing states to reduce switching activity. On the other hand, there is a much longer history on the study of conducting state minimization and assignment at one step (see, for example, [6, 31, 55]), but reducing switching activity or power has never been the goal for any of these approaches.

As we will see in the following motivational example, the best solution that minimizes the switching activity does not necessarily come from the minimized FSM. A similar observation, that state encoding with the minimal code length may not be optimal in terms of switching activity or power, has also been reported earlier [59, 68, 88]. This motivates the proposed concept of **FSM re-engineering**, where we re-construct the FSM to improve the solution's quality. More specifically, we first apply a FSM synthesis technique to obtain a synthesis solution; we then identify the structure in the FSM that might prevent us from getting better solutions and re-construct the FSM accordingly; the re-engineered FSM will be re-synthesized to generate new (and often better) solution.

3.1.1 A Motivational Example

We take the example from a paper on power-driven FSM state encoding [47] to show the potential of the proposed FSM re-engineering approach.

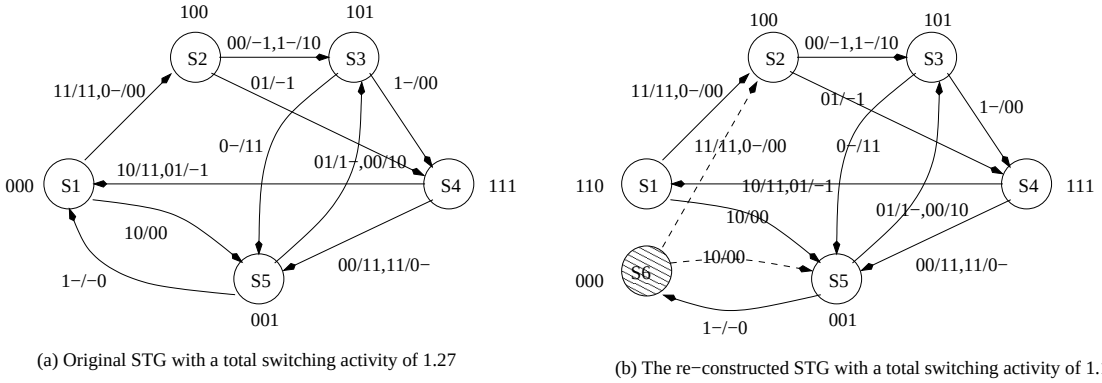


Figure 3.1: A 5-state FSM and a functionally equivalent 6-state FSM.

The state transition graph (STG) in Figure 3.1(a) represents a 2-input 2-output FSM with five states {*S1,S2,S3,S4, S5*}. Each edge represents a transition with the input and output pair shown along the edge. The FSM has already been minimized.

We re-construct this FSM by introducing state *S6* as shown in Figure 3.1(b). One can easily verify that these two STGs are functionally equivalent. In fact, state $S6$ is an equivalent state of $S1$. We then exhaustively check all the possible state encoding schemes for both FSMs and report the one that minimizes total switching activity in Figure 3.1 as shown next to each state.

When we calculate the switching activity, an indicator of power efficiency of the encoding scheme, we observe that it drops from 1.27 to 1.17 (or a 7.9% reduction) after we add state $S6$. Note that the encoding in the original 5-state FSM is optimal

obtained from exhaustive search. This implies that we lose the most energy-efficient encoding for this FSM (and its functionally equivalent FSMs) once it is minimized!

FSM re-engineering not only gives the theoretical opportunity to build FSM with better energy efficiency, it can also be applied to existing low-power encoding algorithms. For example, when we use POW3 [8] instead of the exhaustive search to encode the original 5-state FSM, it gives a coding with switching activity 18.9% higher than the optimal. However, when we use POW3 to encode the equivalent 6-state FSM, it successfully finds a coding that is only 5.4% away from the optimal.

3.1.2 FSM Re-Engineering

FSM re-engineering refers to the procedure of re-constructing an FSM that is functionally equivalent to a given FSM. The goal of FSM re-engineering is to enable synthesis and optimization tools to find better solution for the given FSM by synthesizing and optimizing the re-constructed FSMs.

In the context of low power state encoding, the proposed FSM re-engineering approach takes an encoded FSM as input and outputs a functionally equivalent FSM with reduced switching activity. The novelty of this approach, which separates it from other low power state encoding and re-encoding techniques, is that it investigates the solution space over the entire set of equivalent FSMs rather than restricting to the minimized FSM.

FSM synthesis normally starts with state minimization, which in general results in simplier function implementation, less hardware, and shorter delay. How-

ever, this may not be necessary for power efficiency because power is proportional to the switching activity, not the number of states. Leaving redundancy such as equivalent states in the FSM can be helpful in reducing switching activity. For example, state $S1$ in Figure 3.1(a) originally has four edges and contributes a lot to the total switching activity because states $S1$ and $S4$ have the largest Hamming distance. Duplicating state $S1$ solves this problem as we have seen in Figure 3.1(b).

Finally, we mention the following concerns one may have before we elaborate our FSM re-engineering approach.

- **Area and delay overhead:** Implementing non-minimized FSM may require increased hardware which may also cause area or delay overhead. However, this is not always true. For example, a 36-state FSM and a 42-state FSM need the same number of latches (flip flops, or state registers). Furthermore, we mention that synthesis on minimized FSM does not guarantee the optimality of area and delay either. One example is the one-bit hot encoding we will mention in the next section.
- **Search cost:** Although quality of the solution can be improved theoretically as we search a larger solution space, we have to pay a higher search cost. Note that the FSM re-construction is done after the first round of synthesis and driven by the optimization objective. Therefore, our search is actually guided in a subset of FSMs that could yield good solutions with better chance.

In the rest of this chapter, we apply the proposed FSM re-engineering framework to low power state encoding problem. However, one can apply it for optimiza-

tion of other design objectives such as area and testability.

Section 2 surveys the most relevant work on FSM low power state encoding and shows their difference from the proposed FSM re-engineering framework. The notation and problem formulation are given in Section 3. The power-driven FSM re-engineering approaches, a genetic algorithm and a fast heuristic, are presented in Section 4. Experimental results are reported in Section 5 and Section 6 concludes.

3.2 Related Work

Dynamic power dissipation in CMOS circuits is composed of power consumed in sequential logic and combinational logic. Power dissipated in the combinational logic mainly depends on the complexity of the Boolean logic functions and their gate level implementation. Power dissipation in sequential logic is due to capacitance charging and discharging in state registers caused by the state bits switching, which is often described as

$$P = \frac{1}{2} V_{dd}^2 f \sum_{i \in sb} C(i) E(i) \tag{3.1}$$

where V_{dd} is supply voltage, f is clock frequency, $C(i)$ is the capacitance of the register storing the ith state bit, and $E(i)$ is the expected switching activity of the ith register. $C(i)$ is technology dependent and remains, in general, constant for all the state bits.

There have been a number of power-driven state encoding algorithms to reduce the switching activity $E(i)$ and hereby power. Roy and Prasad propose a simulated annealing based algorithm to improve any given state encoding scheme

[78]. Washabaugh et al. suggest to first obtain state transition probability, then build a weighted state transition graph, and finally apply branch and bound for state encoding [95]. Olson and Kang present a genetic algorithm, where in addition to the state transition probability, they also consider area while encoding in order to achieve different area-power trade-offs [69]. Benini and De Micheli present POW3, a greedy algorithm that assigns code bit by bit. At each step, the codes are selected to minimize the number of states with different partial codes [8]. Iman and Pedram developed a power synthesis methodology and created a complete and unified framework for design and analysis of low power digital circuits [40].

Unlike these power-driven state encoding algorithms, low power state re-encoding techniques start from an encoded FSM and seek for a better coding scheme to reduce switching activity. Hachtel et al. recursively use weighted matching and mincut bi-partitioning methods to re-assign codes [28]. Veeramachaneni et al. propose to perform code exchange locally to improve the coding scheme's power efficiency [92]. Our FSM re-engineering approach is conceptually different from re-encoding in that we look to change the topology of the FSM, not only re-assign codes to the existing states.

The above work takes two common assumptions, 1) they look for codes with the minimal length, that is, the number of bits to represent a state will be $\lceil \log n \rceil$ for any n-state FSM; 2) their encoding (or re-encoding) algorithms are applied after state minimization is done. There are a couple of recent work on non-minimal length encoding algorithms showing that power may be improved with code length longer than this bound [59, 68]. These methods require extra state register(s) in the

FSM implementation which will add to the hardware cost and cause area increase. However, none of the papers have reported the area overhead. Our approach is essentially different from theirs in that we do not introduce extra state bits (when the number of states is not 2^k). Therefore, the area overhead in our approach expects to be much less. Besides, as we have mentioned earlier, our technique is a stand-alone FSM encoding enhancement. FSM re-engineering can be applied to non-minimal length encoding algorithms to find better solutions as well.

Finally, we mention the one-bit hot encoding where each state in an n-state FSM receives an n-bit code with exactly one bit to be 1. This encoding scheme can greatly simplify the logic implementation of the FSM and could also reduce the switching activity because now every pair of states will have a Hamming distance equal to two. However, it requires a code of length the same as the number of states and this makes it impractical for FSMs of large size.

3.3 Preliminary

We consider the standard state transition graph (STG) representation of an encoded FSM $G = (V, E)$, where a node $v_i \in V$ represents a state s_i with code C_i in the FSM M, and a directed edge $(v_i, v_j) \in E$ represents a transition from state s_i to state s_j with transition probability P_{ij} (please refer to section 4.0 for calculation of P_{ij}). We simplify this directed weighted graph G to an undirected weighted graph $\tilde{G} = (V, \tilde{E}, \{C_i\}, \{p_{ij}\})$:

- V, the set of states, which is the same as in G;

- $\tilde{E}$, the set of edges. An edge $(v_i, v_j) \in \tilde{E}$ if and only if $(v_i, v_j) \in E$, or $(v_j, v_i) \in E$, or both;
- C_i, the weight of node $v_i \in V$, which is the code of state s_i;
- p_{ij}, the weight of edge $(v_i, v_j) \in \tilde{E}$, $p_{ij} = P_{ij} + P_{ji}$.

Denote $H(v_i, v_j)$ as the Hamming distance between the codes, two bitstreams C_i and C_j, of states s_i and s_j under the given encoding scheme. The total switching activity of the encoded FSM can be calculated as

$$\sum_{(v_i,v_j)\in\tilde{E}} p_{ij} H(v_i, v_j) \tag{3.2}$$

Recall that two FSMs, M and M', are equivalent if and only if they always produce the same sequence of outputs on the same sequence of inputs, regardless of the topological structure of their STGs. We formally formulate the FSM re-engineering problem as:

> *Given an encoded FSM M and its corresponding graph $\tilde{G} = (V, \tilde{E}, \{C_i\}, \{p_{ij}\})$, construct an equivalent FSM M' and encode it such that in the corresponding graph $\tilde{G'} = (V', \tilde{E'}, \{C'_i\}, \{p'_{ij}\})$, we maximize the total switching activity reduction:*
>
> $$\sum_{(v_i,v_j)\in\tilde{E}} p_{ij} H(v_i, v_j) - \sum_{(u_i,u_j)\in\tilde{E'}} p'_{ij} H(u_i, u_j) \tag{3.3}$$

The FSM re-engineering problem targets the re-construction and encoding of a functionally equivalent FSM for low power FSM implementation. Clearly, it is NP-hard because it requires the best state encoding for the re-constructed FSM M', which is an NP-hard problem. Furthermore, when we restrict M' to be the same

as M, the problem becomes "determining a new encoding scheme to minimize the total switching activity", which becomes the existing FSM re-encoding problem.

The novel contribution of the FSM re-engineering problem is that it re-constructs the original (minimized and encoded) FSM to allow us explore a larger design space for power-efficient FSM encoding. In this chapter, we focus on the FSM re-construction and defer the state encoding problem to existing algorithms. We give an example on how to re-engineer an FSM and explain why it can reduce the switching activity.

3.3.1 An Example of Re-constructing FSMs

We have already seen from Figures 3.1 how to add a new state to the FSM without altering its functionality. Figure 3.2 illustrates a systematic way to do so. We see that a new state, S', is added as a duplicate of state S as follows: S' goes to the same next state under the same transition condition as state S; the transitions from other states to state S in the original STG will be split such that some of them still go to state S while the rest go to the new state S'.

To see the advantage of this non-minimized FSM, we consider a scenario where state S has a large Hamming distance to one of its previous states Sp_j and the transition from Sp_j to S contributes a lot to the total cost. In the re-constructed FSM, we can redirect the next state of this transition to S' and assign S' a code with a small Hamming distance to Sp_j.

For example, in Figure 3.2, no matter which code we assign to state S, it will

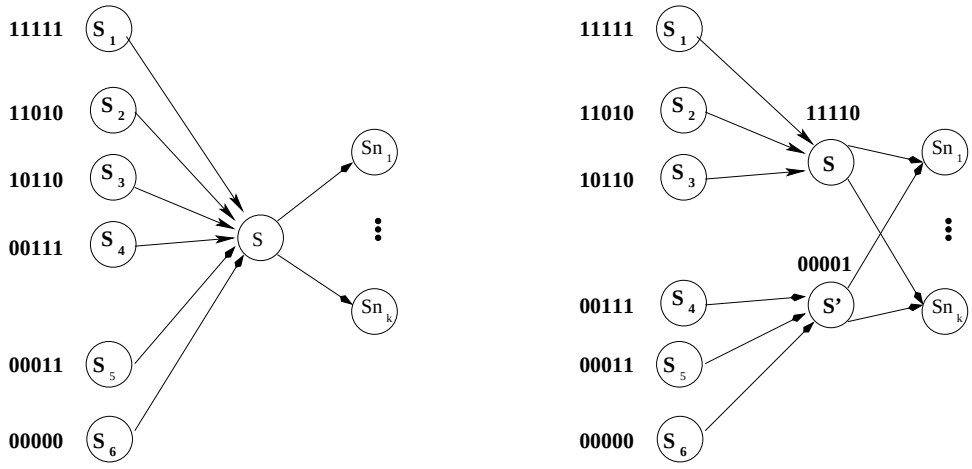


Figure 3.2: Re-constructing an FSM by duplicating a state S.

have a Hamming distance three or larger to at least one of its previous states. (To see this, notice that both codes 11111 and 00000 are assigned to its previous states). However, in the re-constructed FSM, we can assign code 11110 and 00001 to state S and its duplicate S', respectively. This ensures that S will have Hamming distance one from all of its previous states, and S' will have Hamming distance two from S_4 and distance one from all the other previous states.

3.4 Power-Driven FSM Re-Engineering Approach

3.5 FSM Re-engineering Algorithm

In this section, we elaborate the FSM re-engineering approach by showing how the state duplication technique can improve state encoding algorithms. We first propose two heuristic algorithms, based on Hamming distance, on how to select a state for duplication and how to duplicate the selected state. We then present a genetic algorithm for state duplication to target power minimization. Finally, we

describe an integer linear programming (ILP) method that can find the most power-efficient state encoding to evaluate our proposed FSM re-engineering approach.

3.5.1 A Generic Approach

Figure 3.3 outlines the proposed low power state encoding approach by FSM re-engineering. We first compute the original FSM's total switching activity for a reference. Then we re-construct a functionally equivalent FSM and encode it for reduced switching activity. We will use the state duplication technique as an example to illustrate the three key steps for this approach:

1. select the best candidate state for duplication;
2. decide how to duplicate the selected state;
3. estimate the (maximum) switching activity reduction after the state duplication.

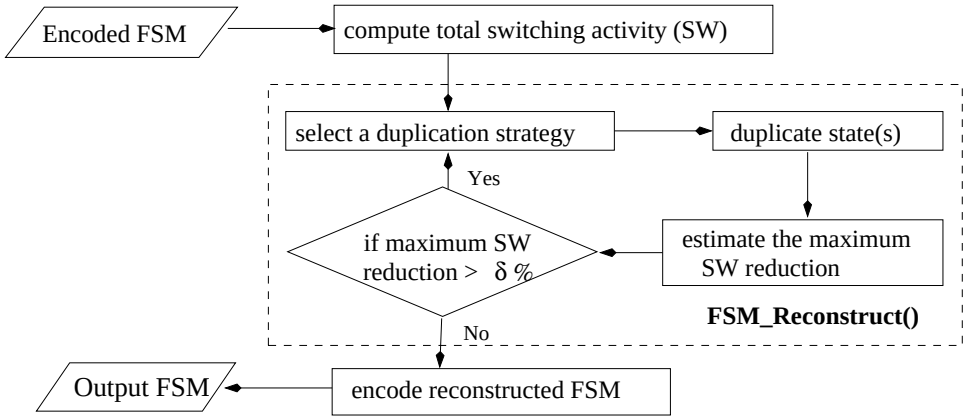


Figure 3.3: FSM re-engineering for low power state encoding.

The strength of FSM re-engineering, as we have discussed earlier, is to improve the performance of FSM synthesis and optimization tools/algorithms. This can be seen from Figure 3.3 as we use the same algorithm, which gives us the input encoded FSM, to encode the re-constructed FSM and produce the encoded FSM in the two lower boxes. In our simulation, POW3 developed by Benini and De Micheli [8] is used as the state encoding scheme.

In this section, we first describe the method to compute switching activity. Next, we present a generic algorithm and a fast heuristic to select states for duplication. We then explain a heuristic on how to duplicate a given state. Finally, we give an integer linear programming formulation of the problem from which the minimum switching activity can be obtained to demonstrate the potential of the proposed FSM re-engineering approach.

4.0 Compute FSM's Switching Activity

As we have mentioned earlier, the proposed FSM re-engineering method seeks for a functionally equivalent FSM that provides opportunity to a low power state encoding scheme so that it can find coding with reduced switching activity.

According to Equation (2), the state transition probability of each edge and the Hamming distance between the two states of each edge must be determined before the calculation of total switching activity. The former measures how frequently each transition occurs and the latter gives the amount that each transition contributes to the total switching activity. The Hamming distance between the two states of

each transition can be conveniently determined after state encoding is performed.

To compute the transition probability, it is necessary to have the input distribution at each state, which can be obtained by simulating the FSM at a higher level of abstraction [95]. This gives us $p_{j|i}$, the conditional probability that the next state is s_j if the current state is s_i. Then we build a Markov chain based on these conditional probabilities to model the FSM. The Markov chain is a stochastic process whose dynamic behavior depends only on the present state and not on how the present state is reached [29]. We now can obtain the steady probability P_i of each state s_i corresponding to the stationary distribution of the Markov chain. The state transition probability P_{ij} for the transition $s_i \rightarrow s_j$ is given by

$$P_{ij} = p_{j|i} P_i \tag{3.4}$$

3.5.2 Genetic Algorithm Based State Duplication

As we have seen in Figure 3.2, we make it possible to assign the same state more than one codes, one for that state and the rest for its duplicate(s), by duplicating that state. However, as it has also been implied in Section 3, choosing an optimal state duplication strategy is also NP-hard. The reason is that it is necessary to encode the duplicated states optimally first to determine whether a state duplication strategy is optimal. This necessary condition itself is already known as NP-hard.

Figure 3.4 depicts the proposed genetic algorithm that searches for a good state duplication strategy. First, since duplicating only a state with only one previous state does not help in reducing the Hamming distance between this state and its

previous state. We eliminate all the states with only one previous state from the queue of states to be duplicated (lines 1-3). For the 5-state FSM in Figure 3.1(a), the candidate queue for state duplication is $\{S1, S3, S4, S5\}$.

A state duplication scheme is represented by a boolean vector of the same length as the above candidate queue. A bit '1' at the i^{th} position of the vector indicates that the i^{th} candidate state is duplicated and a bit of '0' means that the scheme chooses not to duplicate this state. For example, the 6-state FSM in Figure 3.1(b), where state $S1$ is duplicated, corresponds to vector 1000. Each vector is referred as a *chromosome*.

According to each *chromosome*, we duplicate the states (lines 7-9) and calculate its *fitness* (line 10), which is defined as the total switching activity according to that *chromosome*. The smaller the total switching activity, the better the *chromosome*. We start with an initial population of N randomly generated chromosomes (line 5). Children are created by the *roulette wheel* method in which the probability that a *chromosome* is selected as one of the two parents is proportional to its fitness (line 13). With certain ratio, *crossover* is performed among parents to produce children by exchanging substrings in their *chromosomes*. A simple *mutation* operation flips a bit in the *chromosome* with a given probability known as *bit mutation rate* (line 14). When the population pool is full, i.e., the number of new *chromosomes* reaches N, the algorithm stops to evaluate fitness of each individual for the creation of next generation. This process is repeated for MAX_GEN times and the best *chromosome* gives the optimal duplication strategy.

We will discuss how to calculate or estimate the *fitness* of each *chromosome*,

Genetic Algorithm

```
/* Traverse STG and duplicate states. */
 1. for each state in STG
 2.    if it has more than one incoming edge
 3.       put it in candidate queue;
 4. chromosome_length = the size of candidate queue;
 5. initialize N random vectors;
 6. while generation < MAX_GEN
 7.    for each chromosome v_k
 8.       if v_k[i] == 1
 9.          duplicate the i^th candidate state;
10.       v_k.fitness = total switching activity;
11.    do
12.       sort chromosome by non-decreasing fitness;
13.       roulette wheel selection to select parents;
14.       crossover & mutate to create children;
15.    until number of new chromosomes = N
```

Figure 3.4: Pseudocode: State duplication via genetic algorithm

that is, the total switching activity for a new FSM with certain states duplicated.

To calculate switching activity in each step after state duplication, one way is to encode the re-constructed FSM and compute the total switching activity using Equation (2) as stated above in section 4.0. This gives us the actual gain in switching activity reduction by duplicating a set of states. When it is expensive to apply the state encoding algorithm on the entire FSM, we use the following alternative to locally assign the new state the "best" code (might not be feasible) and calculate the lower bound for switching activity.

Lemma 1. Let $\{x_i : (x_{i1}x_{i2}\cdots x_{in})\}$ be the set of states that have transition to/from state s and their codes. Let $p_{x_i s}$ be the transition probability between states x_i and s. The total switching activity is minimized at state s when it has code $c_1c_2\cdots c_n$,

where

$$c_j = \begin{cases} 1 & \text{if } \sum_{x_i} p_{x_i s}(1 - 2x_{ij}) < 0 \\ 0 & \text{otherwise} \end{cases}$$

[Proof]. From the definition, the switching activity at the j-th bit will be $\sum_{x_i} p_{x_i s} x_{ij}$ if $c_j = 0$, and $\sum_{x_i} p_{x_i s}(1 - x_{ij})$ if $c_j = 1$. Comparing these two values, we conclude that c_j should be assigned 1 if $\sum_{x_i} p_{x_i s}(1 - x_{ij}) < \sum_{x_i} p_{x_i s} x_{ij}$, which yields the result as above.

3.5.3 Heuristic on State Selection for Duplication

While genetic algorithm can find a very good state duplication strategy, it may take a long time to converge for FSMs of large size. In fact, the length of *chromosome* (i.e., the size of the queue for states to be duplicated) can be close to the size of the original FSM. (In the worst case, it is only two less than the number of states.) Therefore, we propose a heuristic that select the states for duplication efficiently.

As we have seen from Figure 3.2, states with large (average) Hamming distance from its previous states will benefit because they will have less previous states in the re-constructed FSM, which allows the encoding scheme to find a better code to reduce the Hamming distance. Outgoing edges to the next states and the codes of the next states do not have the same importance because each duplicate state will be connected to the same set of next states to preserve the correct functionality.

For each state s_i, we define:

$$r(s_i) = \sum_{(v_j,v_i)\in E} H(v_i, v_j)/\text{indgree}(v_i) \tag{3.5}$$

where node v_i represents state s_i in the STG and the sum is taken over the number of all the incoming edges (v_j, v_i) at node v_i.

This value measures the average Hamming distance between state s_i and all its previous states. We duplicate one state at a time and each time we select the state according to the following rules:

1. select the state with the largest r-value.
2. if there is a tie, select the state with fewer previous and/or next states.
3. if the tie still exists, break it by selecting a state randomly.

Rule 1. helps us to locate the state(s) such that state duplication can give us large gain in reducing Hamming distance. Rule 2. helps us reduce the size of the re-constructed FSM because each duplicated gate needs to be connected to all the next states and some of the previous states. This could eventually help the encoding algorithm to find a better encoding scheme.

3.5.4 Heuristic on How to Duplicate a Selected State

We now present our algorithm that duplicates the selected state. Ideally, we want to duplicate the state in such a way that the new FSM will maximally reduce the switching activity when encoded optimally. Apparently, this requires solving the NP-hard state encoding problem optimally. Instead, we focus on how to duplicate a state to minimize switching activity locally.

More specifically, let s be the state we select for duplication, PS and NS be the sets of previous states and next states of s respectively in the original FSM. The state duplication procedure 1) creates a state s' that also has NS as its next states, and 2) splits PS into PT_1 and PT_2 and make them as the previous states for s and s' in the new FSM. The goal of such local state duplication is to minimize

$$\sum_{t \in PT_1} P_{ts} H(t,s) + \sum_{t \in NS} P_{st} H(t,s) + \sum_{t \in PT_2} P_{ts'} H(t,s') + \sum_{t \in NS} P_{s't} H(t,s')$$

where P_{ts} is the transition probability from state t to state s and $H(t,s)$ is the Hamming distance between the two states.

The challenge is how to partition the previous states PS into two subsets. Our solution, as shown in Figure 3.5, is based on the fact that the two states in PS with the largest Hamming distance should belong to different partitions. We find, in line 3, states v_k and v_l that have the largest Hamming distance and put them into PT_1 and PT_2 as their respective centers (lines 4-5). For each of the other states $t \in PS$, we include it to the subset whose center is closer to t (lines 6-9). After we finish the partition, we re-compute the centers c_1 and c_2 of the two subsets (line 10) following the method described in Lemma 2 below. We then re-partition set PS based on these new centers and continue if the new partition results in reduced total Hamming distance (line 12).

The following lemmas show the correctness of this approach.

Lemma 2. In any optimal partition, state s and its duplicate s' will have the codes of the two centers.

[Proof]. Suppose that one partition has k states with codes $\{x_{i1}x_{i2}\cdots x_{in} : i = 1, 2, \cdots, k\}$ and they will have state s as their next state in the re-constructed FSM. We want to find the code $c_1c_2\cdots c_n$ for state s to minimize the total Hamming distance

$$\sum_{i=1}^{k} H(s, x_i) = \sum_{i=1}^{k}\sum_{j=1}^{n} |x_{ij} - c_j| = \sum_{j=1}^{n}(\sum_{i=1}^{k} |x_{ij} - c_j|)$$

Because each bit is independent, the above is minimized if and only if $\sum_{i=1}^{k} |x_{ij} - c_j|$ is minimized for each $j = 1, 2, \cdots, n$. Let a be the number of 1's in $\{x_{ij} : i = 1, 2, \cdots, k\}$ and b be the number of 0's. $\sum_{i=1}^{k} |x_{ij} - c_j| = b$ if $c_j = 1$ and $\sum_{i=1}^{k} |x_{ij} - c_j| = a$ if $c_j = 0$. Clearly, it is minimized when c_j is defined as the majority of $\{x_{ij} : i = 1, 2, \cdots, k\}$.

Lemma 3. The optimal partition is reached in time linear to the size of set PS, i.e., the number of previous states of state s.

[Proof]. Because of its discrete nature, every time the loop (lines 6-12) is repeated, the total Hamming distance is reduced by at least 1. Therefore, this loop will stop after being repeated finite times. Furthermore, the largest Hamming distance from s (or its duplicate s') to any state in PS is n. If there are k states in PS, then the loop will not be executed more than kn times.

3.5.5 Determine the Minimum Switching Activity

There are two reasons for us to determine the optimal encoding scheme for a given FSM. First, it allows us to test the quality of low power state encoding heuristics. Second, comparing the minimum switching activity of the original FSM

Local Algorithm to Duplicate a State

/* Duplicate state s */
1. **for** each pair s_i and s_j in PS, the previous states of s
2. compute the Hamming distance $H(s_i, s_j)$;
3. pick s_1 and s_2 s.t. $H(s_1, s_2) = \max_{s_i, s_j \in PS} \{H(s_i, s_j)\}$;
4. $PT_1 = \{s_1\}; PT_2 = \{s_2\}$;
5. $c_1 = s_1; c_2 = s_2$;
6. **for** each state $t \in PS$
7. if $(H(t, c_1) < H(t, c_2))$
8. $PT_1 = PT_1 \cup \{t\}$;
9. else $PT_2 = PT_2 \cup \{t\}$;
10. re-compute c_1 and c_2, the centers of PT_1 and PT_2;
11. $H_{total} = \sum_{t \in PT_1} H(t, c_1) + \sum_{t \in PT_2} H(t, c_2)$;
12. if (H_{total} decreases) **goto** line 6;
13. **for** each state $t \in PT_1$
14. add t as a previous state of state s;
15. **for** each state $t \in PT_2$
16. add t as a previous state of state s';
17. **for** each state $t \in NS$, the next state of s
18. add t as a next state of state s';

Figure 3.5: Pseudocode: Duplicate a State

with that of the re-constructed FSM provides us insight of FSM re-engineering approach's potential power efficiency.

The power-driven state encoding problem can be formulated as follows: *finding a code $x_{i1}x_{i2}\cdots x_{in}$ for each state $x_i, i = 1, \ldots, k$, of a k-state FSM, such that*

$$\sum_{l=1}^{n} |x_{il} - x_{jl}| \geq 1, i \neq j \tag{3.6}$$

and the following (total switching activity) is minimized

$$\sum_{1 \leq i < j \leq k} p_{ij} \sum_{l=1}^{n} |x_{il} - x_{jl}| \tag{3.7}$$

where $p_{ij} = P_{ij} + P_{ji}$ is the total transition probability between states x_i and x_j as

we have defined earlier.

Equation (6) enforces that no two states can have the same code. Expression (7) is the same as the switching activity given in Equation (2) because the Hamming distance between states x_i and x_j is defined as $H(x_i, x_j) = \sum_{l=1}^{n} |x_{il} - x_{jl}|$.

We introduce (Boolean) variables $d_{ij}^{(l)} = |x_{il} - x_{jl}|$ and $d_{ii}^{l} = 0$ for $1 \leq i < j \leq k$ and $1 \leq l \leq n$. Equations (6) and (7) can be re-written in the following linear form:

$$\sum_{l=1}^{n} d_{ij}^{(l)} \geq 1 \tag{3.8}$$

$$\sum_{0<i\leq j\leq k} p_{ij} \sum_{l=1}^{n} d_{ij}^{(l)} \tag{3.9}$$

The definition of $d_{ij}^{(l)}$ is equivalent to the following:

$$\begin{aligned} x_{il} + x_{jl} + (1 - d_{ij}^{(l)}) &\geq 1 \\ x_{il} + (1 - x_{jl}) + d_{ij}^{(l)} &\geq 1 \\ (1 - x_{il}) + x_{jl} + d_{ij}^{(l)} &\geq 1 \\ (1 - x_{il}) + (1 - x_{jl}) + (1 - d_{ij}^{(l)}) &\geq 1 \end{aligned}$$

The problem then becomes a (0-1) integer linear programming (ILP) problem and we can use the off-the-shelf ILP solver to solve it and thus determine the minimum switching activity.

3.6 Experimental Results

We simulate the FSM re-engineering framework on MCNC benchmark suite using POW3 as the low-power state encoding algorithm. For simplicity, we control the state duplication technique such that the encoding bits remains minimal.

Therefore, no state will be duplicated for FSMs with exactly 2^k states. The 26 applicable benchmarks have states from 5 to 48. Our simulation is designed to compare POW3's performance before and after FSM re-engineering using the following metrics: switching activity (calculated from Equation (2)), power and area (simulated using SIS), overhead over the optimal (from solving the ILP problem). We also compare the performance enhancement of POW3 by FSM re-engineering with reported literatures on comparable cases.

Switching Activity Comparison

Table 3.1 reports the switching activity in original FSMs and the re-engineered FSMs, both encoded by POW3. The second column is the length of the code; the third column lists the number of states in the original FSM; the fourth column gives the number of states duplicated by heuristic and genetic algorithm approach. In columns 5 to 7 are the switching activities in original FSMs and FSMs re-engineered by heuristic and genetic algorithm. Columns 8 and 9 show the switching activity reduction in these two approaches respectively.

In the 26 benchmarks, 21 FSMs re-engineered by our genetic algorithm based approach can achieve 0.2 to 34.4% switching activity reduction with an average 8.9% after pow3 encoding. By using our heuristic re-engineering technique, we can reduce switching activities in 17 benchmarks by an average of 6%. We mention that this improvement is significant. First, it is achieved over the encoding by POW3, a state-of-the-art low power encoding algorithm. Second, POW3 itself can achieve an average 12% switching activity reduction over area-driven encoding algorithms

[8]. Finally, we compare this result with one of the existing non-minimal length low-power encoding algorithm [68] based on improvements over POW3. Column 10 and 11 list the switching activity reduction percentage reported in [68]. The table entry filled with an asterisk means result on that benchmark is not reported in their paper. One can see, our heuristic method outperforms their fast approach; and our genetic algorithm based technique is better than their greedy approach.

We also notice that five benchmarks have no improvement after we re-engineer them using genetic algorithm. The reason is that the encoding on the original FSMs are very close to or have already achieved the minimum switching activity. For example, POW3 generates a Gray code for *bbtas*, which is the optimum in switching activity. In these cases, the genetic algorithm based strategy correctly chooses not to duplicate any state because no duplication gives the least total switching. However, in our heuristic approach, for two benchmarks, the algorithm duplicates one or more states that results in negative gain in switching activity reduction. This is due to the inaccurate estimation of switching activity reduction in the proposed algorithm (as discussed at the end of Section 4.1). We mention that this problem can be avoided if we run POW3 every time to decide whether a state should be duplicated.

Power and Area Comparison

As one may observe from Table 3.1, although the code length does not increase, we do duplicate on average 1.7 and 2.1 states in each approach. What is the impact of this to area and power when we implement the re-constructed FSM? Table 3.2 reports this on the circuit implementation obtained by SIS package. We

Table 3.1: Total switching activity reduction on re-constructed FSMs.

FSM	*Bits*	*States*	Dups (heu/ga)	switching activity by pow3			red(%)		red(%) in [68]	
				orig	heu	ga	heu	ga	fast	greedy
example	3	5	3/2	1.5229	1.2703	1.236	16.6	18.8	*	*
s8	3	5	1/2	0.2128	0.1553	0.1396	27	34.4	-16.9	0
ex3	3	5	3/2	1.2	1.075	1.068	10.4	10.9	14.1	19.5
s27	3	6	1/2	0.8866	0.8866	0.8489	0	4.3	0	0
bbtas	3	6	1/0	0.4435	0.4565	0.4435	-2.9	0	0	0
beecount	3	7	0/0	0.5027	0.5027	0.5027	0	0	0	2.1
dk14	3	7	0/1	1.1671	1.1671	1.1235	0	3.7	10.5	10.5
ex5	4	9	2/2	1.1972	1.0442	1.0442	12.8	12.8	0	0
lion9	4	9	2/1	0.5626	0.4571	0.4984	18.8	11.4	20	20
ex7	4	10	1/1	1.0085	0.9487	0.9487	5.9	5.9	0	0
bbara	4	10	0/0	0.3	0.3	0.3	0	0	3.3	6.7
train11	4	11	1/1	0.5540	0.5087	0.5087	8.2	8.2	0	0
modulo12	4	12	2/2	0.5833	0.5	0.5	14.3	14.3	*	*
mark1	4	12	1/1	0.9493	0.9342	0.9195	1.6	3.1	-4.3	-2.2
ex4	4	14	1/0	0.5921	0.6074	0.5921	-2.6	0	7.7	7.7
dk512	4	15	1/1	1.6012	1.4167	1.357	11.5	15.3	7.4	19.6
s208	5	18	13/0	0.4751	0.4751	0.4751	0	0	*	*
s1	5	20	1/8	1.2535	1.1986	1.1633	4.4	7.2	3.8	15.8
ex1	5	20	2/3	0.9823	0.9366	0.8597	4.7	12.5	0.8	2.4
donfile	5	24	3/6	1.5208	1.3906	1.3657	8.6	10.2	-13.6	-6.4
pma	5	24	0/1	0.9112	0.9112	0.8495	0	6.8	*	*
dk16	5	27	2/2	1.9169	1.849	1.7512	3.5	8.6	1.6	9.2
styr	5	30	2/2	0.5302	0.5239	0.4325	1.2	18.4	1.7	6.8
s510	6	47	1/4	0.9245	0.8868	0.8113	4.1	12.2	*	*
planet	6	48	1/8	1.5268	1.4375	1.3469	5.8	11.8	10.8	20
s1488	6	48	0/5	0.3462	0.3462	0.3455	0	0.2	*	*
Average switching activity reduction							6.0%	8.9%	2.5%	6.9%

use the standard *script.rugged* to simplify the circuits and *lib2* library for technology mapping. The area is obtained by *map -s* command. The power is measured in μW using the sequential power estimation package in SIS, assuming a 5V power supply and 20MHz clock frequency.

We are able to get the area and power information from SIS on 14 benchmarks as reported in Table 3.2. We see that an average 7.9% power reduction is achieved at the cost of only 1.3% area increase in the FSMs reconstructed by genetic algorithm. Interestingly, more than one third of the circuits have area reduced after

state duplication. The negative power reduction occurs when the power increase in the combinational part of the circuits exceeds the reduction in the sequential part.

Again, we compare our power-saving results with the data reported in [68]. The comparison is made based on the improvement in dynamic power consumption in SIS. We copied their results from [68] and listed in column 8 and 9. An asterisk in a cell means the power improvement data is not reported in the paper for that benchmark. We see that our methods can achieve greater power-saving improvements over POW3 in almost all the benchmarks than both approaches presented in [68]. Even though, in their paper, they reported an average 17% power improvement over all of their benchmarks, however, this number is from the better one of their two different encoding schemes and is achieved at the cost of increased code length (or equivalently, the number of state registers) and the area change is not reported.

Comparison with Optimal Encodings

For a subset of benchmarks, we are able to find the optimal encodings for both the original and the re-constructed FSMs. This allows us to quantitatively judge the quality of the encodings obtained by POW3. Figure 3.6 depicts the switching activity of optimally encoded new FSM, POW3's encoding on the new FSM (produced by both heuristic and genetic algorithm), and POW3's encoding on the original FSM (from bottom to top). These numbers are all normalized to the switching activity of the original FSM with the optimal encoding.

We see that although FSM re-engineering has the potential to reduce the minimum switching activity by only 2.5% on average, the power efficiency of POW3

Table 3.2: Area and power comparison between original FSM and reconstructed FSM

Circuit	Area		increase	Power (μW)		decrease	Power decrease in [68]	
	orig.	re-eng.	%	orig.	re-eng.	%	fast	greedy
example	43616	44544	2.1%	280.4	287.7	-2.6%	*	*
ex3	46400	50112	8%	314	282.5	10%	5.2%	7%
ex5	70528	80272	13.8%	405.2	467.9	-15.5%	0%	0%
lion9	38976	45472	16.7%	178.3	165.7	7.1%	2.1%	-3.5%
ex7	78416	70064	-10.7%	405.8	287.7	29.1%	11.1%	14.6%
train11	47792	49184	2.9%	212.3	172	19%	12.3%	24.5%
mark1	94656	99760	5.4%	280.7	316.9	-12.9%	-4.2%	-21.3%
dk512	79344	81200	2.3%	430.1	408.1	5.1%	3.3%	11.7%
s1	321088	313664	-2.3%	1388.7	1210.1	12.9%	*	*
ex1	234784	213904	-8.9%	744.9	643.5	13.6%	6%	-10%
dk16	282112	254736	-9.7%	1547.3	1341.7	13.3%	4.8%	9.6%
styr	407856	421312	3.3%	1347.6	1213.1	10%	*	*
s510	302064	283040	-6.3%	923.1	799.7	13.4%	*	*
planet	504832	514112	1.8%	2042.1	1881.3	7.9%	*	*
Average			1.3%			7.9%	4.5%	3.6%

is greatly enhanced. From Figure 3.6, POW3 finds codes for the original FSMs that have switching activity from 11.6% to 48.6% higher than the optimal with average 27.0%. However, when we encode the new FSMs (by heuristic and genetic algorithm) using POW3, the average overhead drops to 12.1% and 6.7%. It even finds an coding that achieves the optimal in *ex3* and codings better than the original optimal in benchmarks *example*, *s8* and *lion9*.

3.7 Summary

The concept of FSM re-engineering is introduced in this chapter. It is a generic framework for FSM synthesis based on the observation that minimizing the number of states in an FSM may lose the optimal solutions, or make it harder to find such

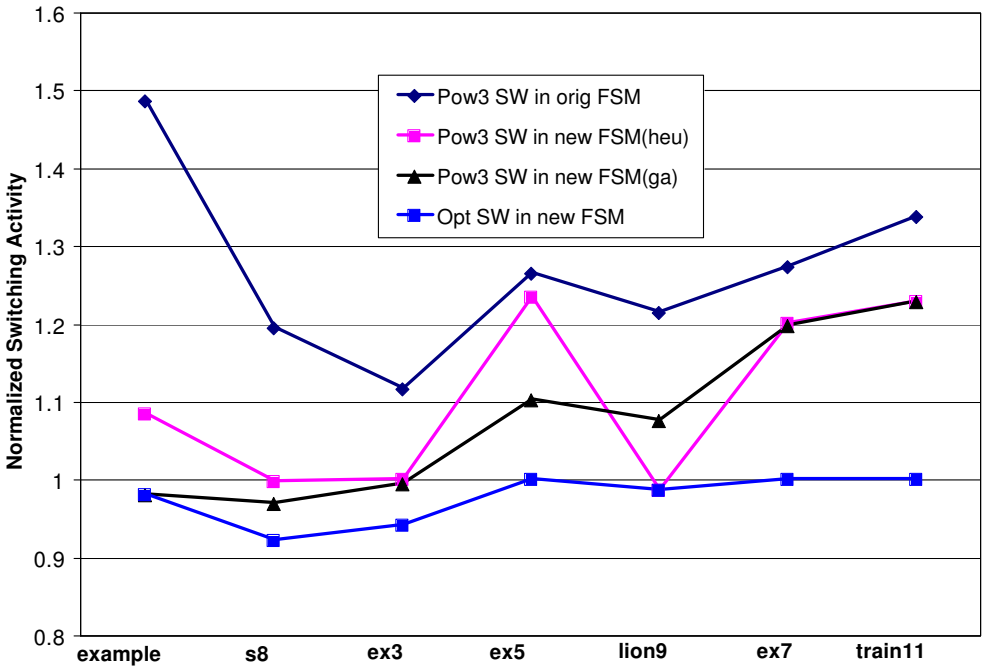


Figure 3.6: Switching activity of POW3's encoding schemes on the original and reconstructed FSMs and the optimal encoding (Opt) on the new FSMs. Normalized to the optimal encoding on the original FSMs.

solutions, for many FSM related optimization problems. To keep the discussion concrete, we study the low power state encoding problem by using a state duplication based FSM re-engineering technique. Our technique does not necessarily provide a power efficient state encoding scheme. Instead, we demonstrate its strength in enhancing the performance of any given power-driven encoding algorithms. We apply this on MCNC benchmark using POW3 as the encoding tool. Experimental results show that POW3's power in reducing circuit's total switching activity has

almost been doubled by the proposed FSM re-engineering approach. Simulation on SIS indicates that an average 7.9% power reduction is achievable with only 1.3% area increase and no additional state registers. We further use an integer linear programming formulation to identify the optimal coding that achieves the minimum switching activity, where we find that the re-engineered FSMs have better optimal codes.

Chapter 4

Dual-V_{th} CMOS Circuit Design for Leakage Reduction

4.1 Introduction

Many of the portable embedded systems often remain in standby mode for a considerable amount of time and dissipate leakage power during such idle period. Leakage power has also increased exponentially in the recent years due to technology scaling and can reach as high as 50% of the total chip power at 65nm technology. As a result, many leakage reduction techniques have been proposed recently [43]. For example, dual threshold voltage process uses devices with higher threshold voltage along non-critical paths to reduce leakage current while maintaining the performance [96]; Multiple-threshold CMOS (MTCMOS) technique places a high V_t device in series with low V_t circuitry, creating a sleep transistor [4, 12, 65]; in dynamic threshold MOS (DTMOS) [5], the gate and body are tied together and the threshold voltage is altered dynamically to suit the operating state of the circuit; controlling the body bias voltage to minimize leakage is discussed in [66]; leakage minimization for cache design is proposed in [45].

In this chapter, we propose an integrated approach that simultaneously considers both dual threshold voltage and input vector in a single optimization loop. Dual threshold voltage (V_t) technique uses two sets of library gates: one implemented with low V_t transistors that have smaller propagation delay but higher leakage power,

and the other with high V_t transistors that have larger delay and lower leakage. In dual V_t design, low V_t gates are placed on the critical paths to guarantee the timing/speed and high V_t gates are used on non-critical paths to reduce leakage. This technique has exhibited excellent results in leakage power reduction while meeting the performance requirements.

Input vector control is another effective approach to static leakage power reduction. It is based on the transistor stack effect – the leakage current in CMOS gates depends (heavily) on the inputs on the gates. For instance, a 2-input NAND gate with input '11' has more than 30 times higher leakage than the same gate with input '00' (see Table 4.1). The input vector control technique seeks for an input vector that minimizes the circuit's total leakage and applies it whenever the circuit is idle. Many algorithms have been proposed to find such minimum leakage vector (MLV) (see a brief survey in Section 2).

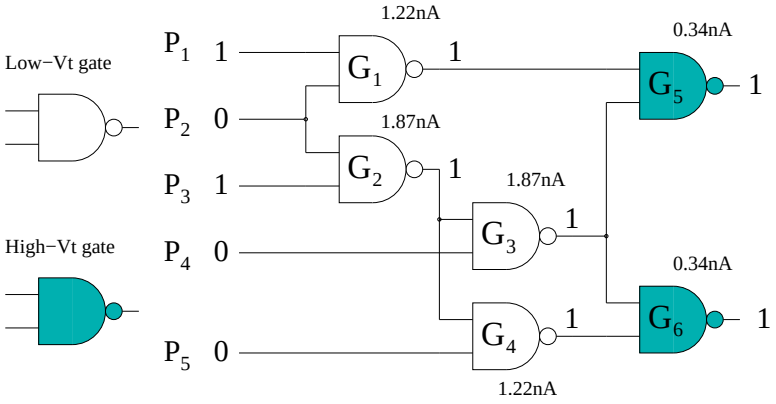
Table 4.1: Leakage current (nA) in high-V_{th} (0.48V) and low-V_{th} (0.33V) two-input NAND gates at different inputs from SPICE simulations.

INPUT	@ Low-V_{th}	@ High-V_{th}	Leakage reduction
0 0	0.19	0.01	0.18
0 1	1.87	0.05	1.82
1 0	1.22	0.03	1.19
1 1	**6.27**	0.34	**5.93**

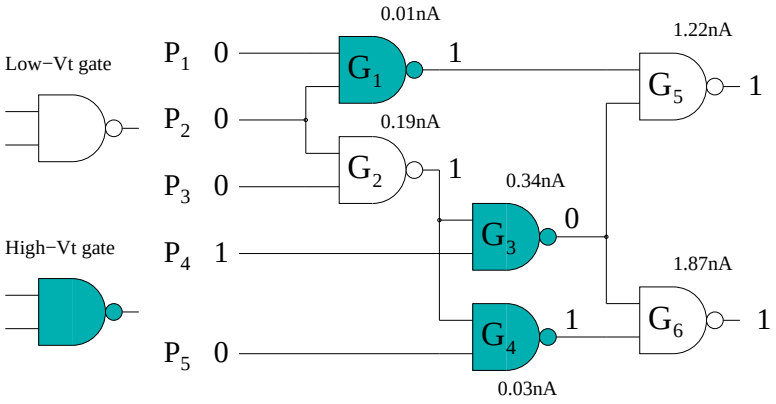
At standby mode, both techniques can be applied to reduce the static leakage power. For dual V_t design, each gate has the option of being implemented with low V_t or high V_t. For input vector control, each primary input signal can either '0' or '1'. This yields a design space that is exponential to the total number of logic gates and primary inputs, which is prohibitively large for any complete search [58].

The following example shows how the leakage can be reduced by applying these two techniques separately and together.

4.1.1 A Motivational Example



(a) On a random input vector; the total leakage reduces from 18.72 nA to 6.86nA.



(b) on the minimum leakage input; the total leakage reduces from 10.96 nA to 3.66nA.

Figure 4.1: Dual-V_{th} assignment for circuit C17 and its impact on leakage reduction.

We consider the same ISCAS benchmark circuit C17 used in [106] as shown

in Figure 4.1(a). Existing dual V_t assignment algorithms are conducted based on a given input vector such as a random input vector, the MLV obtained from any input vector control technique, or statistic values on the inputs. For example, on input vector '10100', the total leakage when we put low V_t (0.33V) on all the gates is $L_0 = 18.72nA$. Assuming that we can increase V_t of one gate to 0.48V on each path without violating the timing requirement, an exhaustive search for the dual V_t assignment will assign gates G_5 and G_6 to high V_t, resulting in a total leakage $L_1 = 6.86nA$.

Meanwhile, exhaustive search on all the 32 input vectors indicates that the MLV is '00010'. The circuit's total leakage with all gates at 0.33V is $L_2 = 10.96nA$. When we combine these two techniques by applying the dual V_t approach under the MLV, gates G_1, G_3, and G_4 will be assigned 0.48V as shown in Figure 4.1(b). The total leakage reduces to $L_3 = 3.66nA$.

Finally, our proposed simultaneous dual V_t and input vector assignment finds the same dual V_t assignment as shown in Figure 4.1(b) with a different input vector '00011'. This solution has a total leakage $2.29nA$, 38% less L_3, the result when we exhaustively search MLV and then assign dual V_t. It is also the overall optimal solution when we exhaustively search for input vector selection and dual V_t assignment simultaneously.

4.1.2 Main Idea and Contribution

From the above example, we see that none of the dual V_t assignment or the MLV selection or a simple serial combination of the two will be able to find the optimal solution for leakage reduction. Repetitively applying MLV selection and dual V_t assignment could improve the quality of the solution. However, due to the high complexity of both dual V_t assignment and MLV search algorithms, such iterative strategy will take extremely long time before it converges and therefore is not practical for large circuits.

Our proposed heuristic approach exploits the interdependency of input vector and dual V_t assignment by considering them simultaneously in a single optimization loop. It is based on the following two observations:

- First, raising V_t has very different impact on the leakage reduction for the same gate with different inputs. As shown in the last column of Table 4.1, when we increase the V_t for the NAND2 gate from 0.33V to 0.48V, the leakage reduction is 5.93 nA under inputs '11', but only 0.18nA when the inputs are '00'.

- Second, according to a recent study on all the MCNC and ISCAS benchmarks [106], even after selecting input vectors carefully, more than 40% of the circuit's total leakage are dissipated on gates with inputs that make these gates dissipating the most leakage, although there are only about 15% such gates.

The first observation suggests that the effectiveness of dual V_t technique is directly related to the input vector and it is beneficial to assign high V_t to gates

with inputs that make these gates dissipating the most leakage, which are referred to as gates at their worst leakage state (WLS). The second observation suggests that significant amount of leakage can be reduced by controlling only such WLS gates.

In our approach, we determine whether to assign '0' or '1' to each primary input one by one. When making this decision, we consider the dual V_t option and assign high V_t to certain gates. After all the primary inputs get a value, we obtain an input vector. Under such input vector, we apply an existing dual V_t algorithm [94] to assign V_t to the gates that have not received V_t.

The key contribution that separates our approach from the dual V_t assignment and input vector selection algorithms is the seamless integration of these two techniques. Comparing with the serial combination of an input vector selection algorithm and the dual V_t assignment algorithm [94], we are able to achieve an average leakage reduction of 12% and 19% on popular ISCAS and MCNC benchmarks, respectively. The worst case run time for our approach is $O(k^2 \cdot n^2)$, where k is the number of primary inputs and n is the number of logic gates.

4.1.3 Chapter Organization

The rest of the chapter is organized as follows: Section 2 reviews the related work in dual V_t assignment and input vector control. In Section 3, we present our algorithm and analyze its performance. In Section 4, we report the experiment results. We conclude the chapter in Section 5.

4.2 Related Work

A dual-V_t assignment algorithm is presented in [96] where the V_t assignment is done at the logic gate level, i.e., all transistors in a gate are either assigned high V_t or low V_t. Their approach starts with all gates at low V_t; a breadth-first traversal is performed on the circuit from the primary inputs to the primary outputs. Each gate is examined and assigned high V_t if it does not violate the timing constraints in the circuit. This greedy algorithm may preclude many solutions where another set of other gates on and off critical paths could have been assigned high V_t with more leakage reduction and no performance degradation. A more effective approach is proposed in [94], where the problem of dual V_t assignment is formulated as the Max-Cut problem in a weighted direct cyclic graph. A heuristic is used to partition the circuit graph according to the logic topological level of the gates. Dual V_t assignment is conducted on gates that are at one level at a time. Furthermore, a dual-V_t assignment algorithm together with supply voltage scaling and gate sizing is proposed in [87]. Their approach creates extra slack at some gates via gate sizing and voltage scaling; a V_t assignment is then carried out simultaneously. More recently, a dual-V_t design method for FPGAs is studied in [52].

On the other hand, input vector control for leakage minimization has also received plenty of attention. This problem is shown to be NP-complete in [106]. A random search algorithm is proposed in [32]. The underlying transistor stack effect that causes leakage dependency on inputs are explained in [42]; a greedy heuristic algorithm is also proposed to assign the inputs. In [77]. the authors

developed a heuristic algorithm based on node controllability. It can achieve a fairly good results with much shorter run-time. In addition to the heuristic approaches, an integer linear programming (ILP) approach has been adopted in [25] to attack this problem. Furthermore, pseudo boolean enumeration method for input vector assignment is described in [18].

All the above works address the dual V_t and input vector assignment separately. In [58], the simultaneous V_t and input vector assignment is considered. The authors formulate the problem as search in a nested binary tree: one for dual V_t assignment and the other for input vector assignment. A branch-and-bound approach is taken in the search. It must be pointed out that even a lower bound computation method is given, no results are reported using the complete branch-and-bound method due to its infeasibility for large circuits. Instead, two heuristics are proposed: heuristic 1 only does one downward traversal based on the lower bound; heuristic 2 traverses the input assignment search tree subject to a fixed runtime constraint, while the dual-V_t gate assignment search tree is kept to a single downward traversal. Our algorithm differs from their's in following aspects: first, their dual-V_t assignment does not exploit the circuit topology and the greedy single downward traversal may preclude many possible good V_t configurations; second, their algorithm target at assigning V_t to transistors in the gate, our V_t assignment algorithm is restricted at logic gate level. The key advantage of our approach is that we leverages circuit topology and the fact that most of the leakage power are produced by a small portion of gates in their worst leakage states.

4.3 Simultaneous Dual V_t Assignment and Input Vector Selection

In this section, we will describe the dual V_t assignment and input vector selection problems, explain the intuition and the challenge for conducting these two problems simultaneously, and demonstrate our algorithm.

4.3.1 Dual V_t Assignment

The dual V_t assignment problem can be stated as: *given a combinational circuit with a timing constraint at the primary outputs and a technology library with high-V_t and low-V_t gate implementations, map each gate in the circuit to either the low-V_t or the high-V_t implementation, such that the delay of the circuit satisfies the timing requirement and the total leakage power is minimized.*

To compute the delay of the circuit, a static timing analysis is performed. Typically, the arrival time at the primary inputs are set to be zeros. Following the topological order, the arrival time at each gate g is computed as the maximal arrival time of its fanin gates plus the propagation delay at g. The delay of the circuit is the maximal arrival time at the primary outputs. Similarly, given a required time at the primary outputs, the required time at each gate can be calculated following the reverse topological order: the required time at each gate g is the minimum of the required time at its fanout gates i minus the propagation delay in i. In addition, a slack is associated with each gate g, which is defined as the required time minus the arrival time at g.

When the threshold voltage V_t of a CMOS gate is changed from low V_t to high

V_t, the leakage current in the gate is reduced significantly at the cost of increased propagation delay. If the delay constraint at the gate is still satisfied (i.e., the delay increase is within the slack), the assignment of high V_t to that gate is said to be feasible. The challenge is, whenever a gate g is assigned V_t high, the slacks of all the other gates on the paths that pass through g will be affected, and it is not clear which gate should be assigned V_t high to maximize the leakage reduction.

In [94], a levelization-based heuristic algorithm is proposed. Given a combinational circuit, the level of a primary input is zero and the level of a gate is one more than the maximum of the levels of all its fanin gates. The circuit is partitioned into k partitions with all the gates in one partition having the same level. Initially, all the gates in the circuit are assigned low V_t. Then high V_t are assigned to those gates that can achieve the maximal leakage reduction while satisfying the delay constraints. This assignment is conducted at one level of gates at a time. If the sum of leakage reduction in feasible gates at that level is the largest, the gates at that level will be assigned high V_t first. The advantage of this level-based V_t assignment is that all the feasible gates at one level simultaneously without violating the timing. In this case, only one static timing analysis is needed after each V_t assignment for one level.

4.3.2 Input Vector Selection

The input vector selection problem can be stated as: *given a combinational circuit and the leakage current of each gate under different input combinations,*

determine an input vector at the primary inputs (PIs) such that the total leakage current of all the gates in the circuit is minimized. The obtained input vector is referred as the minimum leakage vector (MLV).

A complete survey of the different approaches to solve this problem can be found in [106]. Here we give two definitions, *leakage observability* and *worst leakage state*, both of which will be used when we elaborate our approach on simultaneous dual V_t assignment and input vector selection.

As we have shown in the previous sections, leakage current in a circuit depends on the input vector at its PIs. To measure the impact of the value at one PI on the circuit's total leakage, the term "leakage observability" has been defined in [42]. In this chapter, we will extend this definition to circuit with dual V_t technology.

Definition 1. *Leakage Observability: given a partially assigned input vector $\vec{w}$, the leakage observability of a PI pin i is:*

$$L_{obs}(i,w) = \begin{cases} 0 & \text{if } i \in \vec{w}; \\ |Lavg^1(i,\vec{w}) - Lavg^0(i,\vec{w})| & \text{otherwise.} \end{cases} \tag{4.1}$$

where $Lavg^v(i,\vec{w})$ is the portion of total leakage in the circuit attributable to PI pin i being forced to the value v (v = 0 or 1).

To compute $Lavg^v(i,\vec{w})$, we need to calculate $Lavg^v_n(i,\vec{w})$ for each gate n. For a single gate, $Lavg^v_n(i,\vec{w})$ is the average leakage for all possible input states of gate n, given that part of PIs that belong to vector $\vec{w}$ have been assigned and PI pin i is assigned value v. Consider the NAND2 gate G_3 in Figure 4.1(a). When $\vec{w}$ = {11xxx}, $Lavg^0_3(3,\vec{w}) = \frac{1}{2}\cdot(L(3,\text{`}11') + L(3,'10'))$ and $Lavg^1_3(3,\vec{w}) = \frac{1}{2}\cdot(L(3,\text{`}01') +$

$L(3,'00'))$, where L(k, $\vec{u}$) represents the leakage of a gate k under the input $\vec{u}$.

In a circuit with dual-V_t technology, V_t becomes another necessary parameter to calculate the leakage. For each gate in the library, a SPICE simulation can be conducted at priori to estimate the leakage current in each gate with both high V_t and low V_t implementation for each of the possible input values. These leakage current values can be stored in a look-up table.

Definition 2. *Worst Leakage State (WLS): when the inputs to a gate g results in the largest leakage among all the possible inputs, it is said that g is in the WLS or g is a WLS gate.*

Due to the stack effect, when a gate is in the WLS, its leakage current is much higher (in the order of 10X) than the same gate with other input states. In [106], it is observed that in the MCNC and ISCAS benchmarks, WLS gates account for only about 15% of the total gates, but account for more than 40% of the total leakage current. Therefore it is promising to focus the leakage reduction on the WLS gates and a gate replacement technique is proposed to reduce WLS gates, and hence the total leakage, in the circuit.

4.3.3 Combining Dual V_t Assignment and Input Vector Selection

With both the dual V_t option and input vector control, we can expect higher leakage reduction by solving the following problem: *given a combinational circuit with a timing constraint at the primary outputs, a technology library with high-V_t and low-V_t gate implementations and their corresponding leakage current under different*

input combination for each logic gate, select a value for each primary input and assign a V_t for each logic gate such that the delay of the circuit satisfies the timing requirement and the total leakage is minimized.

As we have seen in the motivational example, dual V_t assignment and input vector selection alone will not give us the optimal solution due to their assumptions: In input vector selection, we need to know the V_t assignment for each individual gate because the leakage currents are different at different V_t. This implies that we need to perform dual V_t assignment first. On the other hand, the dual V_t assignment algorithms need to assume an input value for each gate due to the stack effect as shown in Table 4.1, which suggests that the input vector needs to be decided first.

A simple combination of the two approaches also fails to find the optimal solution. Another approach is to iteratively repeat an input vector selection algorithm and a dual V_t assignment algorithm until there is no more leakage reduction can be achieved. However, the search space for input vector selection is 2^k, where k is the number of PIs; and that for the dual V_t assignment is 2^n, where n is the number of gates in the circuit; the time for this iterative improvement approach to converge is not clear either. Therefore, this approach is not practical.

The challenge remains as how to combine these two leakage reduction techniques efficiently and effectively.

4.3.4 Algorithm Description and Analysis

Our approach selects the value for each PI one by one and explicitly looks for WLS gates and aggressively assigns them high V_t as long as the timing constraint is not violated. This is based on the following facts: (1) the input values determine whether a gate is at its WLS or not; (2) a gate at WLS dissipates large amount of leakage; (3) applying high V_t to a WLS gate saves more leakage.

Input: gate-level circuit L; two threshold voltage V_{t-Low} and, V_{t-High}.
Output: a minimum leakage input vector to the circuit L^* with dual V_t assignmeng.
Simultaneous V_t and MLV Assignment Algorithm:

```
1. assign V_{t-Low} to each gate g in L;
2. while there are unassigned primary inputs (PI)
3.     for each unassigned PI i
4.         set the value of i, val(i) = 0;
5.         propagate val(i) = 0 in the circuit;
6.         while there are gates in WLS
7.             extract the first WLS gate g_j from the WLS list;
8.             if(assign(g_j, V_{t-High}) == TRUE)
9.                 update slack using static timing analysis
10.        Lavg_i(L, 0) = total leakage in the circuit
11.        set val(i) = 1;
12.        repeat lines 5 to 9 with val(i) = 1
13.        Lavg_i(L, 1) = total leakage in the circuit;
14.        Lobs(i) = |Lavg_i(L, 0) - Lavg_i(L, 1)|;
15.        k = the index of PI with the largest Lobs(i);
16.        set the value of the k_th PI to be the one with smaller Lavg_i;
17. apply dual V_t assign on the rest of circuit;
```

Figure 4.2: Pseudo-code of the simultaneous dual V_t assign and input vector selection algorithm.

Figure 4.2 gives the pseudo code of our algorithm. The program starts with all the PIs unassigned and all gates at low V_t (line 1). Then it assigns PI pins one at a time based on the leakage observability at the PIs to obtain an input vector (the

outer while loop from line 2 to line 16). Specifically, for each PI, we try both input values 0 and 1. A logic simulation is performed to propagate that value throughout the circuit (line 5). Meanwhile we can collect all the WLS gates in the circuit into the list. We remove the WLS gates one by one from the list and assign them high V_t if such assignment will not violate the timing constraint (the inner while loop from line 6 to line 9). When there are no WLS gates left in the list (there may still be WLS gates remaining in the circuit, these WLS gates cannot be assigned high V_t due to timing violation), we stop and compute the average leakage of the circuit (line 10). Similarly, we can calculate the average leakage for the PI taking the other opposite value (lines 11 and 12). We compute the leakage observability for each PI (line 14) and find the one with the largest leakage observability (line 15). We assign this PI the value, '0' or '1', that results in smaller average leakage (line 16). Consequently, all the high V_t assignment based on this value propagation are finalized. This procedure repeats until all the PIs have been assigned. At the end, a standard dual V_t assignment method can be performed on the circuit to assign the rest of gates high V_t, if possible (line 17).

Algorithm complexity analysis

Let k and n be the number of PIs and the number of logic gates in the circuits, respectively. Propagating an input value to the entire circuit in line 5 takes $O(n)$ time. The timing analysis in line 9 takes $O(n)$ time assuming that we have built a topological order of the gates outside the while loop. The number of WLS gates is of

$O(n)$, so the complexity of the inner while loop (lines 6-9) is $O(n^2)$. Calculating the total leakage in line 10 takes $O(n)$ time. Lines 12 and 13 have the same complexity of lines 5-10. Finding the PI pin with the largest leakage observability in line 15 takes $O(k)$ time. The for loop (lines 3-16) repeats for each unselected PI and is bounded by $O(k)$. Each iteration of the outer while loop (lines 2-16) will determine the value of one PI, so it will be repeated no more than k times. Therefore, the worst case run time of this algorithm is $O(k^2 \cdot n^2)$.

Run time improvement

There are several ways to improve the run time of the above algorithm. For example, the timing analysis step in line 9 is time consuming and the above algorithm performs timing analysis for each WLS gate. We can reduce this partitioning the WLS gates by their levels as defines in [94] and explained in Section 3.1. Then we only need to do one timing analysis for all the WLS gates in the same level. This reduces the run time for the inner while loop (lines 6-9). Another possibility is to calculate the leakage observability without performing the high V_t assignment. Then we do the timing analysis only on the selected PI, the one with the largest leakage observability, to decide which value we will assign to this PI. This takes the timing analysis out of the for loop (lines 3-16) and will reduce the overall run time complexity from $O(k^2 \cdot n^2)$ to $O(k \cdot n^2)$.

Table 4.2: Leakage current (nA) in the library gates.

Gate Type	Input	Low V_{th}	High V_{th}
INV	0	1.87	0.05
	1	3.14	0.17
NAND2	0 0	0.19	0.01
	0 1	1.87	0.05
	1 0	1.22	0.03
	1 1	6.27	0.34
NOR2	0 0	3.75	0.20
	0 1	2.41	0.14
	1 0	3.14	0.17
	1 1	0.67	0.05

Table 4.3: Propagation delay (ns) in the library gates.

Gate Type	Low V_{th}	High V_{th}
INV	0.13	0.22
NAND2	0.16	0.27
NOR2	0.18	0.30

4.4 Experimental Results

We implemented the simultaneous dual-V_{th} and input vector assignment algorithm in SIS [86] and ran the simulation on a Sun Ultra 5.10 workstation with 256MB memory. Ten MCNC [115] benchmark circuits and eleven ISCAS [39] benchmark circuits are used to test our algorithm; these circuits are implemented with the same technology library used in [94]. The leakage current and average propagation delay in each library gates are shown in Table 4.2 and 4.3 respectively.

All the gates in a circuit are initially assigned low V_{th}. The largest arrival time at the primary outputs is used as the timing constraint for the circuit. We first obtain an MLV to the circuit by uniformly generating 50,000 random input vectors and choosing the one that results in the minimum total leakage. We select 50,000 in

Table 4.4: Comparison of individual MLV and V_{th} assignment with simultaneous MLV and V_{th} assignment algorithm on MCNC benchmarks in terms of runtime and leakage. The red(%) column reports the reduction over the combined serial randome MLV search and V_{th} assignment.

Circuit	# PI	# Gate	50K Random Search		Dual-V_{th} on MLV		Simultaneous V_{th} and MLV.		
			runtime	leakage	runtime	leakage	runtime	leakage	red(%)
i1	25	52	27.6	86.9	0.1	38.5	0.3	36.0	6%
i2	201	242	133.85	411.6	3.6	262.4	75.1	145.6	45%
i3	132	132	81.4	302.9	0.1	296.4	19.0	240.2	19%
i4	192	308	161.3	672.2	3.3	270.5	75.9	201.3	26%
i5	133	445	218.9	972.5	4.9	932.9	51.2	619.7	34%
i6	138	764	353.05	1354.7	6.1	1103.3	85.4	1056.7	4%
i7	199	1011	465.55	2133.8	10.4	1695.6	367.0	1265.1	25%
i8	133	3764	1836.5	8400.2	45.4	5440.5	693.0	5004.0	8%
i9	88	1218	554.75	2603.2	13.5	1525.4	58.1	1535.1	-1%
i10	257	3366	1647.25	8499.3	100.9	1736.4	1526.6	1340.1	23%
Average			548.0		18.8		295.2		19%

order to ensure that the random search strategy runs at least as long as our approach for a fair comparison. It has also been shown that with a 99.3% confidence ratio, the number of input vectors that have smaller leakage current is less than 0.01% of the entire vector space. Based on the MLV, the levelization-based dual-V_{th} assignment algorithm [94] is applied on the circuits driven by the MLV. We report the results in terms of total leakage current and CPU runtime in this case; and compare them with the ones achieved by our simultaneous dual-V_{th} and input vector assignment algorithm in Table 4.4 and 4.6, for the MCNC and ISCAS benchmarks, respectively.

Table 4.4 reports the results in ten MCNC benchmarks. The first column lists the circuit names; the second column lists the number of PIs in each circuit and the third column lists the number of gates after technology mapping. The fourth and fifth column show the runtimes and total leakage by the 50K random search MLV algorithm; the next two columns show the runtimes and leakage after dual-

Table 4.5: Number of gates in the worst leakage state (# WLS), number of WLS gates at high V_{th} (# WLS*), and the total number of gates assigned high V_{th} (# V_{th}H) in MCNC circuits with serial dual-V_{th} assignment and with simultaneous dual-V_{th} assignment and input vector control.

Circuit	dual-V_{th} based on MLV			Simultaneous V_{th} and MLV Assign.		
	# WLS	# WLS*	# V_{th}H	# WLS	# WLS*	#V_{th}H
i1	2	1	32	3	2	32
i2	31	12	75	7	1	75
i3	1	0	4	0	0	4
i4	34	19	212	24	19	212
i5	36	2	19	28	24	40
i6	31	2	220	31	2	220
i7	103	3	248	95	59	299
i8	481	183	1434	480	230	1473
i9	81	11	566	80	10	531
i10	477	393	2845	462	446	2862

V_{th} assignment algorithm is applied based on the MLV. One can see that dual-V_{th} algorithm can achieve an average 35% leakage reduction in the circuit.

The last three columns show the results achieved by the simultaneous V_{th} and input vector assignment algorithm. The runtimes are in the same order of the serial approach (i.e., random MLV search followed by dual-V_{th} assign); the total leakage is 48% smaller than that with MLV only and 19% smaller than that with the serial MLV and dual V_{th} assignment. This means by assigning V_{th} concurrently with input vector, we can improve the performance of dual V_{th} technique by 37% on average.

This improvement comes from the reduction of the number of WLS gates that are at low V_{th}, which is reported in Table 4.5. The second to fourth columns show the number of gates in WLS ($\#WLS$), the number of WLS gates that are assigned high V_{th} (# WLS*) and the total number of gates assigned high V_{th} (# V_{th}H) in circuits with dual-V_{th} assignment following MLV. We found that on average 11% of the

Table 4.6: Comparison of individual MLV and V_{th} assignment with simultaneous MLV and V_{th} assignment algorithm on ISCAS benchmarks in terms of runtime and leakage. The red(%) column reports the reduction over the combined serial randome MLV search and V_{th} assignment.

Circuit	# PI	# Gate	50K Random Search		dual-V_{th} on MLV		Simultaneous V_{th} and MLV.		
			runtime	leakage	runtime	leakage	runtime	leakage	red(%)
C17	5	6	0.9	10.31	0.0	10.13	0.2	10.13	0%
C6288	32	2400	1000.0	6660	3.5	4522.45	53.6	2956.74	35%
C1908	33	771	301.1	1764.73	1.4	748.18	10.4	724.13	3%
C432	36	282	112.9	703.54	0.6	324.41	4.1	312.88	4%
C1355	41	552	231.4	1310.16	1.4	789.08	13.0	721.98	9%
C499	41	567	227.0	1412.67	1.7	768.39	13.9	787.09	-2%
C3540	50	1526	641.8	3850.03	6.5	1081.12	58.9	929.25	14%
C880	60	442	186.9	1047.43	2.9	240.71	12.6	199.95	17%
C5315	178	2513	1126.1	6146.03	22.3	2092.48	537.4	1414.4	32%
C7552	207	3381	1530.8	8402.34	58.4	1887.14	1155.9	1636.76	13%
C2670	233	1087	510.5	2776.4	24.4	533.79	405.6	490.86	8%
Average			533.6		11.2		206.0		12%

total gates are in their worst leakage states, contributing 32% of total leakage before dual-V_{th} assignment is performed; After V_{th} assignment, 28% of the WLS gates are assigned high V_{th} and the total leakage contributed by the WLS gates becomes 13%. In three columns are the same set of data in circuits with simultaneous dual-V_{th} assignment and MLV. We see that even though the total number of gates assigned high V_{th} are similar in both approaches, the percentage of WLS gates assigned high V_{th} in our algorithm is 56% higher than that in the serial approach.

Table 4.6 and Table 4.7 show the same set of results on ISCAS benchmark circuits. On average, our simultaneous algorithm can achieve 12% of leakage reduction over the serial MLV and V_{th} assignment approach. However, this reduction is smaller than that in MCNC benchmarks. This is partly due to the reason that the ISCAS benchmarks have fewer number of PIs and larger logic levels in general, which

makes the input vector at the primary inputs less important [16]. Currently, we are investigating on how to improve this approach's performance in circuits with large logic depth. One possibility is to combine our approach with the gate replacement technique [106] and/or the internal point control technique [1]. Note that in both ISCAS and MCNC benchmarks, there is a circuit in each set that produces more leakage in simultaneous assignment algorithm than in the traditional approach (the negative % reduction). Also, in circuit C17, there is no WLS gate assigned V_{th}H because the timing constraint we enforced at the circuit is very tight.

Table 4.7: Number of gates in the worst leakage state (# WLS), number of WLS gates at high V_{th} (# WLS*), and the total number of gates assigned high V_{th} (# V_{th}H) in ISCAS circuits with serial dual-V_{th} assignment and with simultaneous dual-V_{th} assignment and input vector control.

Circuit	dual-V_{th} based on MLV			Simultaneous V_{th} and MLV Assign.		
	# WLS	# WLS*	# V_{th}H	# WLS	# WLS*	#V_{th}H
C17	1	0	1	1	0	1
C6288	698	196	992	690	564	904
C1908	118	67	499	117	87	473
C432	23	7	163	32	20	158
C1355	113	42	245	110	57	205
C499	90	46	267	100	73	210
C3540	202	160	1150	209	194	1154
C880	66	51	373	55	46	373
C5315	366	241	1814	378	356	1864
C7552	543	440	2824	522	469	2832
C2670	166	137	942	165	148	925

4.5 Summary

In this chapter, we proposed a simultaneous dual-V_{th} and input vector assignment algorithm to reduce static leakage current in the circuits when they are at

standby mode. Our algorithm is based on the observation that gates that are in their worst leakage input state contribute the most to the total circuit leakage. Our algorithm iteratively finds an input vector and a V_{th} assignment solution that either remove gates from the worst leakage states, or assign high V_{th} to those gates that can not be assigned out states due to logic dependencies. The experimental results show that this simultaneous algorithm can reduce on average 19% and up to 45% of total leakage current over the traditional individual dual V_{th} and input vector assignment approaches.

Chapter 5

Gate-Level Input Vector Control for Static Power Minimization

5.1 Introduction

As the VLSI technology and supply/threshold voltage continue scaling down, leakage power has become more and more significant in the power dissipation of today's CMOS circuits. For example, it is projected that subthreshold leakage power can contribute as much as 42% of the total power in the 90nm process generation [43]. Many techniques thus have been proposed recently to reduce the leakage power consumption. Dual threshold voltage process uses devices with higher threshold voltage along non-critical paths to reduce leakage current while maintaining the performance [96]. Multiple-threshold CMOS (MTCMOS) technique places a high V_{th} device in series with low V_{th} circuitry, creating a sleep transistor [65]. In dynamic threshold MOS (DTMOS) [5], the gate and body are tied together and the threshold voltage is altered dynamically to suit the operating state of the circuit. Another technique to dynamically adjust threshold voltages is the variable threshold CMOS(VTCMOS) [53]. All of these approaches require the process technology support.

The input vector control (IVC) technique is applied to reduce leakage current at circuit level with little or no performance overhead [21]. It is based on the well-known transistor stack effect: a CMOS gate's subthreshold leakage current varies

(a)

INPUT	Leakage(nA)
0	best:100.3
1	worst:**227.2**

(b)

INPUT	Leakage(nA)
00	best: 37.84
01	2nd worst: 100.30
10	95.17
11	worst: **454.50**

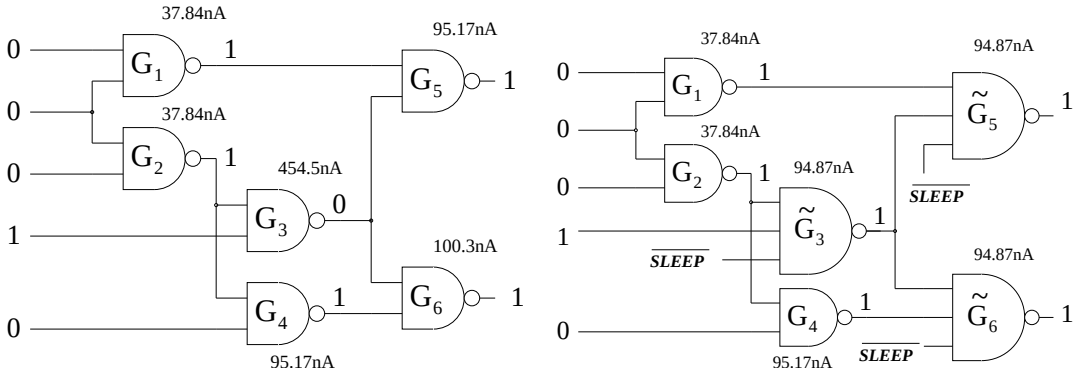
(c)

INPUT	Leakage (nA)
000	best: 22.84
001	37.84
010	37.84
011	2nd worst: 100.30
100	37.01
101	95.17
110	94.87
111	worst: **852.40**

Figure 5.1: Leakage current of (a)INVERTER, (b)NAND2 and (c)NAND3. Data obtained by simulation in Cadence Spectre using 0.18 μm process.

dramatically with the input vector applied to the gate [42]. Recently, Lee et al. made the similar observations on gate oxide leakage that it is also dependent on the input vectors to a CMOS gate [57]. We note that the maximal and minimal leakage vectors are the same for both subthreshold leakage and gate leakage. In our study, we use Cadence Spectre to measure the overall leakage current in a CMOS gate that includes both subthreshold leakage and gate leakage. Figure 5.1 lists the overall leakage current in INVERTER, NAND2 and NAND3 gates under all the possible input combinations. We see that the worst case leakage (marked in bold) is much higher than the other cases. The idea of IVC technique is to manipulate the input vector with the help of a sleep signal to reduce the leakage when the circuit is at the standby mode [32]. The associated minimum leakage vector (MLV) problem seeks to find a primary input vector that minimizes the total leakage current in a given circuit. [1, 9, 18, 25, 32, 42, 77]. The MLV problem is NP-complete and both exact and heuristic approaches have been proposed to search for the MLV. A detailed survey is given in Section II.

In this chapter, we consider how to enhance IVC technique with little or no re-design effort. In particular, we study the **MLV+ problem** that seeks to modify a given circuit and determine an input vector such that the circuit's functionality is maintained at the active mode and the circuit leakage is minimized when the circuit is at standby mode. Our solution to this problem is based on the concept of gate replacement that is motivated by the large discrepancy between the worst leakage and the other cases (see Figure 5.1). The essence of gate replacement is to replace a logic gate that is at its worst leakage state by another library gate. This is illustrated by the following example.



(a) Original MCNC benchmark circuit C17 with total leakage 831.08nA under the optimal MLV.

(b) New circuit C17 with three gates replaced and total leakage 476.88nA under the same MLV.

Figure 5.2: A motivation example for gate replacement.

Consider circuit C17 from the MCNC91 benchmark suite [115] (Figure 5.2(a)). An exhaustive search finds the MLV {0,0,0,1,0}, with the corresponding minimum total leakage current of 831.08nA. Note that gate G_3 has its worst leakage current (454.5nA) with input {1,1}, which contributes more than half of the total leakage. In fact, we have observed that a significant portion of the total leakage is often

caused by the gates that are in their worst leakage state (see Table 5.2 in Section V).

Instead of controlling the primary inputs, we consider replacing these leakage-intensive gates. In particular, we replace the NAND2 gate G_3 by a NAND3 $\tilde{G}_3$ where the third input $\overline{SLEEP}$ is the complement of the $SLEEP$ signal (Figure 5.2(b)). At active mode, $\overline{SLEEP} = 1$ and $\tilde{G}_3$ produces the same output as G_3. But at the standby mode, $\overline{SLEEP} = 0$ and $\tilde{G}_3$ has a leakage of $94.87nA$ (Figure 5.1(b)), which is much smaller than G_3's $454.5nA$.

However, this replacement also changes the output of this gate at the sleep mode and affects the leakage on gates G_5 and G_6. In this case, we replace them in a similar fashion. As a result, the new circuit's total leakage becomes $476.88nA$, a 43% reduction from the original $831.08nA$ in Figure 5.2(a).

The proposed gate replacement technique is conceptually different from the existing input vector control methods. In fact, they are complementary to each other. Specifically, IVC method considers the entire circuit and searches for an appropriate input vector in favor of small leakage. The gate replacement technique targets directly at the logic gates that are in their worst leakage state (WLS) under a specific input vector and replace them to reduce leakage. This chapter has the following contributions:

1. We examine the effectiveness of IVC methods[1] in multilevel circuits. For all the 69 MCNC91 benchmarks, we obtain the optimal MLV for small circuits

[1]IVC-based approaches such as internal control point insertion [1] will be discussed in Section II

and the best over 10,000 random input vectors for large circuits. The number of gates in their WLS are on average 15% and 17% respectively, but they contribute more than 40% of the circuit's total leakage.

2. Motivated by the above observation, we propose the technique to replace gates that are in their WLS by other library gates that will generate less leakage current at those states. Unlike other leakage reduction techniques such as MTCMOS and DTMOS, this modification of the circuit does not require changes of process technology in the design flow. Hence, it will not increase the design complexity or the leakage sensitivity.

3. We implement a fast gate replacement algorithm that gives an average of 10% leakage reduction for a fixed input vector. This algorithm's run time complexity is linear to the number of gates in the circuit in average cases and quadratic in the worst case.

4. We develop a divide-and-conquer approach to combine gate replacement and IVC. It reduces the leakage by 17% and 24% over the optimal/sub-optimal MLV mentioned in 1) with little area and delay overhead. The number of gates in their WLS is dropped to 4% and 9% respectively.

5.2 Related Work

In this section, we mainly survey the efforts on input vector control (IVC)-based leakage reduction techniques. A survey on other leakage minimization tech-

niques can be found in [21].

The effect of circuit input logic values on leakage current was observed by Halter and Najm [32]. The underlying reason of this effect was explained by Johnson et al. [42] as the transistor stack effect. Authors in [32] proposed a technique to insert a set of latches with MLV stored in to the primary inputs of a circuit, forcing the combinational logic into a low-leakage state when the circuit is idle. Many algorithms have been proposed to find such minimum leakage vectors (MLV). Based on the nature of these algorithms, they can be classified into the following groups:

Heuristic Algorithms: These include the random search algorithm developed by Halter and Najm [32] and the genetic algorithm proposed by Chen et al. [16].

Johnson et al. [42] defined *leakage observability* for each primary input as the degree to which the value of a particular input is observable in the magnitude of leakage current. They iteratively chose the input with the largest leakage observability and assigned it with a value that results in the smallest leakage. The input combination constructed in this greedy fashion was taken as the MLV.

In [77], Rao et al. introduced the concept of *node controllability*, which is defined as the minimum number of inputs that have to be assigned to particular values to ensure that a node (or gate) is in a specific state. Based on this, they proposed a fast greedy heuristic to determine the values of the primary inputs that minimize the node's leakage.

Exact Algorithms: The MLV problem can be modeled as a pseudo Boolean Satisfiability (SAT) problem. This formulation allows us to apply the off-the-shelf SAT solvers to find the MLV for leakage reduction [1, 3].

Gao and Hayes [25] formulated the MLV problem as an integer linear programming(ILP) problem. They first use pseudo-Boolean functions to represent leakage current in different types of cells with the general sum-of-products form. Then they apply the well-known Boole-Shannon expansion [30] to linearize the objective function and constraints. At last, they use an off-the-shelf ILP solver to solve the ILP optimization. For large circuits, the authors proposed a simplified mixed-integer linear programming formulation that uses selective variable-type relaxation to reduce the runtime.

Based on the pseudo Boolean formulation of the leakage in CMOS gates, two implicit pseudo boolean enumeration algorithms are presented in [18]. The input space enumeration method leverages integer valued decision diagrams and works well for small circuits. The hyper-graph partitioning based recursive algorithm represents a given circuit as a hyper-graph, partitions it, and uses divide-and-conquer to solve the problem. The trade-off between dynamic and leakage power in choosing the MLV has also been discussed.

Internal Point Control: Due to the ineffectiveness of IVC technique for circuits with large logic levels, Abdollahi et al. proposed a technique to directly control the value of internal pins to reduce leakage [1]. Their first approach inserts multiplexers at the input pins of each gate. The $SLEEP$ signal selects the correct input in active mode and chooses the input values that produce low leakage current in standby mode. This approach can reduce leakage in the CMOS gates significantly; however, the inserted multiplexers will also generate leakage current and introduce extra delay and area. In their second approach, they modify the library gates by adding $SLEEP$

signal-controlled transistors in the gate to select the low-leakage inputs for its fanout gates. They reported an average leakage reduction of 25% within 5% delay penalty and no more than 15% area increase. However, since the structure of the gates is changed, a new set of library gates are needed.

Our gate replacement technique belongs to the class of internal point control, but is conceptually different from [1] in the following aspects: 1) They treat each input pin of the gates as potential place to insert multiplexers, while we consider only roots of each tree. The search space is reduced substantially. 2) Their purpose of modifying a gate G is to produce the low-leakage input for G's fanout gate while we aim to reduce leakage current at G itself. 3) They modify gates whenever necessary while we restrict our algorithm to replace gates only by the available gates in the library, and hence do not require gate structure modification. However, these two approaches can be combined as we will discuss in more details in Sections III and IV.

5.3 Leakage Reduction by Gate Replacement

A logic gate is at its worst leakage state (WLS) when its input yields the largest leakage current. Regardless of the primary input vector, a large number of gates are at WLS, particularly when the circuit has high logic depth. Take the 69 MCNC91 benchmarks for example. For each of the 69 circuits, when we apply the optimal (or sub-optimal) MLVs to these circuits, 16% of the gates on average remain at WLS, producing more than 40% of the circuit's total leakage. A detailed

report can be found in Section V. In this section, we describe the gate replacement technique that targets directly the leakage reduction in WLS gates.

5.3.1 Basic Gate Replacement Technique

As we have shown in the motivation example in Section I, the proposed gate replacement technique replaces a gate $G(\vec{x})$ by another library gate $\tilde{G}(\vec{x}, SLEEP)$, where $\vec{x}$ is the input vector at G, such that

1. $\tilde{G}(\vec{x}, 0) = G(\vec{x})$ when the circuit is active ($SLEEP = 0$);
2. $\tilde{G}(\vec{x}, 1)$ has smaller leakage than $G(\vec{x})$ when the circuit is in standby ($SLEEP = 1$).

The first condition guarantees the correct functionality of the circuit at active mode. The second condition reduces the leakage on gate G at the standby mode, but it may change the output of this gate. Note that, although we do not need to maintain the circuit's functionality at the standby mode, this change may affect the leakage of other gates and should be carefully considered.

Figure 5.3(a) shows that the replacement of G by $\tilde{G}$ changes the output from 0 to 1. For simplicity, we assume that G's fanout only goes to gate H which can be either a NAND or a NOR or an INVERTER. In Figure 5.3(b) and (d), we see that such change does not affect the output of gate H and therefore it won't affect any other gates in the circuit. Let $L(G(11))$ be the leakage of gate G with input 11, we can conveniently compute the leakage reduction by this replacement, which is $L(G(11)) + L(H(00)) - L(\tilde{G}(110)) - L(H(10))$ in the case of (b) for example.

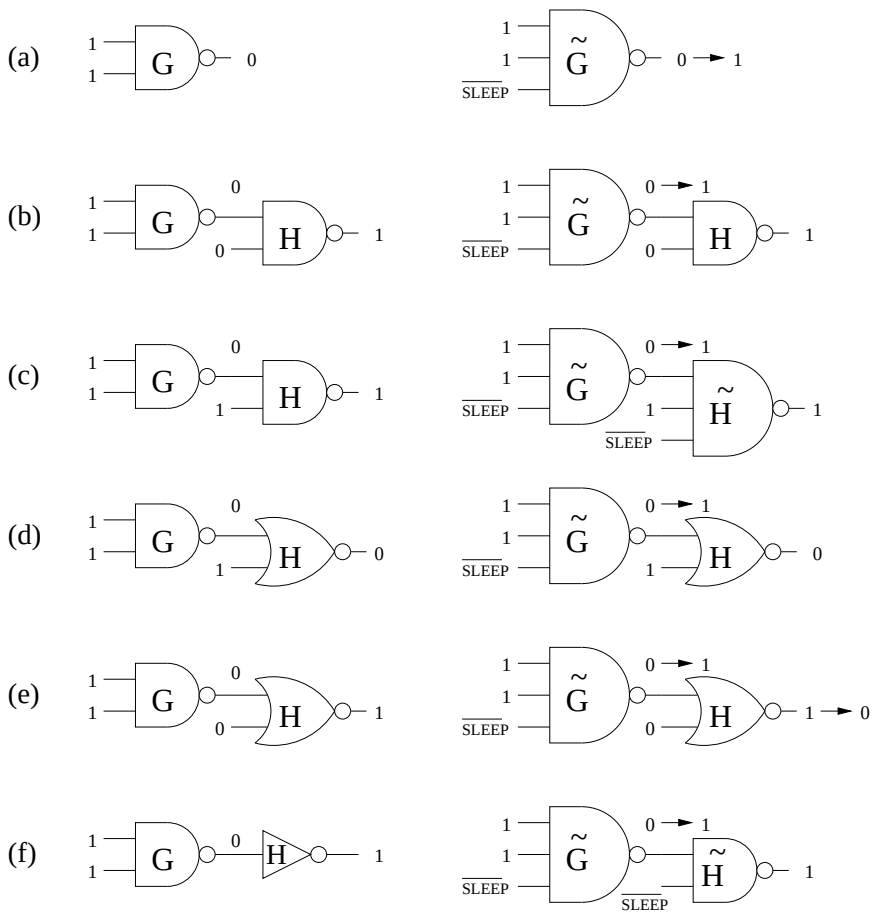


Figure 5.3: Gate replacement and the consequence to its fanout gate.

In Figure 5.3(c), the replacement at gate G not only changes the output of gate H, it also puts H at its WLS. Our solution is to replace the NAND2 gate H by an NAND3 $\tilde{H}$. This preserves the output of H and the leakage change will be $L(G(11)) + L(H(01)) - L(\tilde{G}(110)) - L(\tilde{H}(110))$. Similarly, in Figure 5.3(f), we replace the INVERTER by a NAND2 gate. Finally, in Figure 5.3(e), the replacement of G moves both gates G and H away from their WLS. It also changes the output of the NOR gate H, which we can conduct similar analysis.

Remarks:

- **General Fanout** The above analysis is applicable to G's fanout gate H of any type. The change of G's output either does not affect H's output (Figure 5.3 (b) and (d)) or changes H's output. In the latter case, we either change H's output back (Figure 5.3 (c) and (f)) or continue the analysis starting from H (Figure 5.3 (e)).

- **Beyond library gates** If the library does not have a replacement for G, we can add one transistor into the N or P sections of G to meet conditions 1 and 2. This is similar to the gate modification method proposed in [1]. However, they attempt to control the output of the modified gate in order to reduce the leakage in its fanout gate by producing the desirable signal. Our gate replacement targets directly the leakage reduction of the current gate.

- **Multiple fanouts** When gate G has multiple fanouts, we analyze each of them and then consider their total leakage when we compute the leakage change due to the replacement of gate G.

- **Compatibility** The gate replacement technique does not change the primary input vector of the circuit. This implies that we can combine it with existing MLV searching strategies to further reduce leakage. The MLV+ problem is based on this observation and is discussed in details in next section.

- **Power overhead** There is not much dynamic power overhead because the SLEEP signal remains constant at active mode and will not cause any additional switching activities. The leakage in gates $\tilde{G}$ and G may be different

at active mode. Such difference becomes negligible when the circuit stays at standby mode long enough [1].

- **Other overhead** Gate replacement may introduce delay and area overhead. This overhead can be controlled by restricting the replacement off critical path and transistor resizing. Gate replacement does not add new logic gates and thus requires little or no effort to redo the place-and-route.

5.3.2 A Fast Gate Replacement Algorithm

Based on the above gate replacement technique, we propose a fast algorithm that selectively replaces gates to reduce the circuit's total leakage for a given input vector. Figure 5.4 gives the pseudo-code of this algorithm.

We visit the gates in the circuit by the topological order. We skip all the gates that are not at WLS and the gates that have already been visited or marked (line 16) until we find a new gate G_i at WLS (line 2). Lines 3-9 find a subset of gates $\mathcal{S}$ and temporarily replace them. $\mathcal{S}$ includes all the unmarked gates whose leakage and/or output is affected by the replacement we attempt to do on gate G_i and other gates in $\mathcal{S}$. We then compute the total leakage change caused by the replacement of gates in $\mathcal{S}$ (line 10) and adopt these replacements if there is a leakage reduction (lines 11-13). Otherwise, we simply mark gate G_i as visited and do not make any replacement (line 14). We then look for the next unmarked gate at WLS and this procedure stops when all the gates in the circuits are marked.

Correctness: The topological order guarantees that when we find a gate at its

Input: $\{G_1, G_2, \cdots\}$: gates in a circuit sorted topologically,
$\{x_1, x_2, \cdots\}$: an input vector,
$SLEEP$: the sleep signal.
Output: a circuit of the same functionality when $SLEEP = 0$ and with less leakage when $SLEEP = 1$.
Gate Replacement Algorithm:

```
1. for each gate G_i ∈ {G_1, G_2, ···}
2.     if (G_i is at WLS and not marked)
3.       include G_i in the selection S;
4.       while (there is new addition to S)
5.         for each newly selected gate G in S
6.           if (there exists library gate G̃ meets the conditions
             in Section III-A)
7.             temporarily replace G by G̃;
8.             if (output of G is changed due to this replacement)
9.               include G's unmarked fanout gate G_j in S;
10.      compute the total leakage change of gates in S;
11.      if (there is leakage reduction)
12.        mark all gates G_j in the selection S;
13.        make the replacements in lines 7,9,or 10 permanent;
14.      else mark gate G_i only;
15.      empty the selection S;
16.    else mark G_i if it has not been marked yet;
```

Figure 5.4: Pseudo-code of the gate replacement algorithm.

WLS, all its predecessors have already been considered. The replacement at line 7 ensures that the functionality will not change at the active mode. The subset $\mathcal{S}$ constructed in the **while** loop (lines 3-9) is the *transitive closure* of gates that are affected by the replacement action at gate G_i. Therefore, we only need to compute the leakage change on gates within $\mathcal{S}$ (line 10). We make the replacement only when this leakage change is in favor of us, so the new circuit will have less leakage in standby mode.

Complexity: Let n be the number of gates in the circuit. The **for** loop is linear to n. Inside the for loop, the computation of leakage change and the marking of all

gates in $\mathcal{S}$ (line 10-15) is linear to $|\mathcal{S}|$, the number of gates in $\mathcal{S}$. The **while** loop (lines 3-9) stops when there is no new addition to $\mathcal{S}$ and this will be executed no more than $|\mathcal{S}|$ times. As we have discussed in section 3.1 (see Figure 5.3), in most cases, $\mathcal{S}$ includes only G and its fanout gates. However, it may include all the gates of the circuit in cases similar to Figure 5.3 (e) and so $|\mathcal{S}|$ cannot be bounded by any constant. That is, $|\mathcal{S}|$ is $O(n)$ in the worst case and $O(k)$ on average, where k is the maximal fanout of the gates in the circuit. Consequently, the complexity of this gate replacement algorithm is $O(n^2)$ in the worst case and $O(kn)$ on average.

Improvement: There are several ways to improve the leakage reduction performance of the above gate replacement heuristic. The tradeoff will be either increased design complexity, or reduced circuit performance, or both. First, one can consider gates that are not in the library as we have commented in the remarks in Section III-A (line 6). However, this requires the measurement of leakage current, area and delay in these new gates as they are not available in the library. A second alternative is to insert control point at one of G's fanins. For example, one can find the fanin y such that replacing y by its complement y' gives G the largest leakage reduction. If $y = 0$, replace it by $OR(y, SLEEP)$; if $y = 1$, replace it by $AND(y, \overline{SLEEP})$. However, the addition of new gates may require the repeat of placement and routing and will incur more area and delay penalty in general. Third, one may also consider both the library gate replacement and control point insertion at the same time and choose the one that gives more leakage reduction. Finally, whenever we replace gate G_i, we also make the replacement for all the other gates in the selection $\mathcal{S}$

permanent (line 13). We have tested a couple of alternatives and they give limited improvement in leakage reduction at very high cost of run time complexity.

The incentive to keep the run time complexity of this gate replacement algorithm low is that it will be combined with IVC technique under the following divide-and-conquer approach to solve the MLV+ problem.

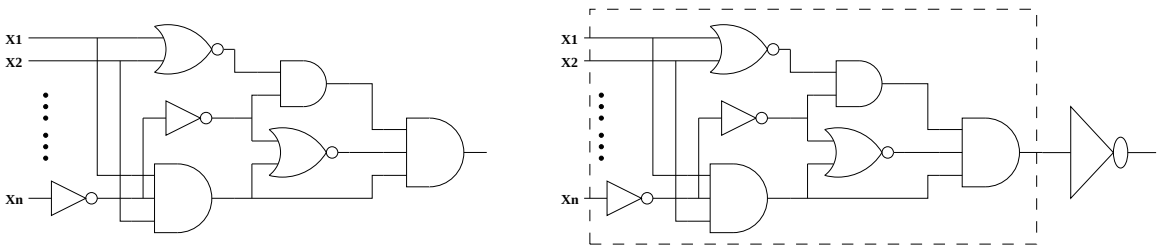
5.4 Solving the MLV+ Problem

Recall that the minimum leakage vector (MLV) problem seeks the input vector that minimizes the circuit's total leakage. It has been claimed that this problem is NP-complete for general circuits [1, 18, 42, 77]. But no formal proof has been given to our knowledge. In this section, we first give a brief proof of the NP-completeness of the MLV problem and then define the MLV+ problem, an extension of the MLV problem. Our main focus will be on the divide-and-conquer approach that solves the MLV+ problem.

5.4.1 NP-Completeness of the MLV Problem

The MLV problem can be defined as follows: given a combinational circuit consisting of primary inputs (PIs), primary outputs(POs), internal logic gates connected by nets/wires, and the leakage current of each gate under different input combinations, determine an input vector at the PIs such that the total leakage current of all the gates in the circuit is minimized.

Theorem: The MLV problem is NP-complete.



(a) A circuit for satisfiability test.

(b) Reducing the satisfiability test to MLV.

Figure 5.5: Illustration for the proof of the NP-completeness of the MLV problem.

Proof. On one side, we have already mentioned a couple of exact algorithms that solve the MLV problem by reducing it to NP-complete problems such as pseudo Boolean satisfiability and integer linear programming.

On the other side, we show that the NP-complete CIRCUIT-SAT problem [26] can be reduced to the MLV problem. Consider an arbitrary circuit shown in Figure 5.5(a), to test whether the circuit is satisfiable (i.e., producing a logic '1' at its output), we construct a new circuit by adding a big inverter at its output (Figure 5.5(b)). The inverter is big in the sense that it has a huge leakage value L when its input is '0' and a small leakage ϵ when its input is '1'. Actually, we can set L to be the sum of ϵ and the leakage of each gate in the circuit when it is in its WLS. Now we solve the MLV problem for this modified circuit. If the total leakage is less than L, clearly the original circuit is satisfiable and the MLV is one input vector that makes the circuit output logic '1'. Otherwise, because that the only way for the total leakage to be larger than L is when the input to the big inverter is '0', the original circuit is not satisfiable. □

5.4.2 The MLV+ Problem and Outline of the Divide-and-Conquer Approach

In the previous section, we have seen that leakage current can be further reduced over the MLV by the proposed gate replacement technique. We have also mentioned that this technique is independent of the input vector and can be combined with the MLV method. We hence formulate the following **MLV+ problem**:

> *Given a combinational circuit with PIs, POs, the internal logic gates that implement the PI-PO functionality, and the leakage current of each library gate under its different input patterns, determine a gate level implementation of the same PI-PO functionality without changing the place-and-route and an input vector at the PIs that minimizes the total leakage.*

Apparently , this is an extension of the MLV problem with the relaxation of modifying circuit by gate replacement. It enlarges the search space of MLV and provides us with the opportunity of finding better solution. For a circuit of k PIs and n internal logic gates, the search space for the original MLV problem is the 2^k different input combinations. Under the above MLV+ formulation, the search space becomes $2^k \cdot \Pi_{i=1}^{n} l_i$, where l_i is the number of library gates that can replace gate i, including gate i itself. Assuming that half of the gates have one replacement, then the solution space for MLV+ problem will be $2^{n/2}$ times larger than the solution space for the MLV problem. Even when we restrict the gate replacement technique only to gates that are at their WLS, this will be significant because (1) a circuit

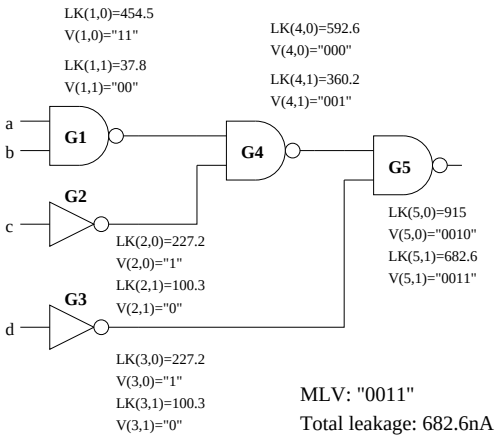
normally has more gates than PIs ($n >> k$) and (2) the percentage of gates in WLS is considerably high (16% on the MCNC91 benchmark when MLV is applied, and will be higher as the logic depth of the circuit increases).

As we have analyzed in the previous section, the MLV+ problem not only enlarges the solution space for the IVC method, it also has the great potential in improving the solution quality (in terms of leakage reduction) because of the stack effect. However, one challenge is how to explore such enormous solution space for better solutions. Given the NP-completeness of the MLV problem, we consider special circuits where the MLV+ can be solved optimally and develop heuristics for the general case. In the rest of this section, we describe details of our proposed divide-and-conquer approach that consists of the following phases:

1. decompose a general circuit into tree circuits.
2. find the MLV for each tree circuit optimally by dynamic programming.
3. apply the gate replacement technique to the MLV for each tree to further reduce leakage.
4. connect the tree circuits by a genetic algorithm.

5.4.3 Finding the Optimal MLV for Tree Circuits

A tree circuit is a single output circuit in which each gate, except the primary output, feeds exactly one other gate. A general combinational circuit can be trivially decomposed into non-overlapping tree circuits [30]. This is illustrated in Figure 5.8. The circuit in (a) is not a tree because gate G_3 has two fan-out gates G_5 and G_6.



LK(1,0) = L(G1("11")) = 454.5 V(1, 0) = "11"
LK(1,1) = min(L(G1("11")), L(G1("10")), L(G1("00"))) = L(G1("00")) = 37.8
V(1, 1) = "00"

LK(2, 0) = L(G2("1")) = 227.2 V(2, 0) = "1"
LK(2, 1) = L(G2("0")) = 100.3 V(2, 1) = "0"

LK(3, 0) = L(G3("1")) = 227.2 V(3, 0) = "1"
LK(3, 1) = L(G3("0")) = 100.3 V(3, 1) = "0"

LK(4, 0) = L(G4("11")) + LK(1, 1) + LK(2,1)=592.6
V(4, 0) = "000"
LK(4, 1) = min{ L(G4("10"))+LK(1,1)+LK(2,0), L(G4("01"))+LK(1,0)+LK(2,1),
L(G4("00"))+LK(1,0)+LK(2,0)} = L(G4("10"))+LK(1,1)+LK(2,0)=360.2
V(4, 1) = "001"

LK(5, 0) = L(G5("11"))+LK(4,1)+LK(3,1) = 915
V(5, 0) = "0010"
LK(5, 1) = min{ L(G5("10"))+LK(4,1)+LK(3,0), L(G5("01"))+LK(4,0)+LK(3,1),
L(G5("00"))+LK(4,0)+LK(3,0)} = L(G5("10"))+LK(4,1)+LK(3,0) = 682.6
V(5, 1) = "0011"

Figure 5.6: Dynamic programming to find optimal MLV in a tree circuit.

By splitting at the fanout of G_3, we get three trees with G_3, G_5 and G_6 being the root of each tree respectively.

We consider a tree circuit with gates $\{G_1, G_2, \cdots, G_n\}$ sorted in the topological order, which is preserved by the tree decomposition.

Let $L(G_i(\vec{x}))$ be the leakage current in the gate G_i when vector $\vec{x}$ is applied at G_i's fanins. Each gate G_i can be treated as the root of a sub-tree circuit. Let $LK(i, z)$ be the minimum total leakage of the tree circuit when it outputs logic value z at root G_i and $\vec{V}(i, z)$ be the input vector to the tree circuit that achieves $LK(i, z)$. We develop a dynamic programming approach to compute the pairs $(LK(i, 0), \vec{V}(i, 0))$ and $(LK(i, 1), \vec{V}(i, 1))$ for each gate G_i. The MLV for the tree circuit rooted at gate G_n, with gates $\{G_1, G_2, \cdots, G_n\}$ sorted in the topological order, can then be determined conveniently.

1. For each input signal to the tree, define

$$LK(0, z) = 0, \quad \vec{V}(0, z) = z \tag{5.1}$$

2. For each gate $G_i (i = 1, 2, ..., n)$, let

$$LK(i, z) = \min_{\forall \vec{x},\, s.t. G_i \text{ outputs } z} (L(G_i(\vec{x})) + \sum_{j=1}^{t} LK(i_j, x_{i_j})) \tag{5.2}$$

$$\vec{V}(i, z) = \cup_{j=1}^{t} \vec{V}(i_j, x_{i_j}^0) \tag{5.3}$$

where $\{x_{i_1}, x_{i_2}, \cdots, x_{i_t}\}$ are the fanins of G_i from gates $\{G_{i_1}, G_{i_2}, \cdots, G_{i_t}\}$ respectively and the input combination $\{x_{i_1}^0, \cdots, x_{i_t}^0\}$ achieves $LK(i, z)$.

3. The minimum leakage of the tree circuit with gates $\{G_1, \cdots, G_n\}$ is given by

$$\min\{LK(n, 0), LK(n, 1)\} \tag{5.4}$$

and the MLV will be either $\vec{V}(n, 0)$ or $\vec{V}(n, 1)$ accordingly.

A step-by-step illustration of the dynamic programming can be found in Figure 5.6.

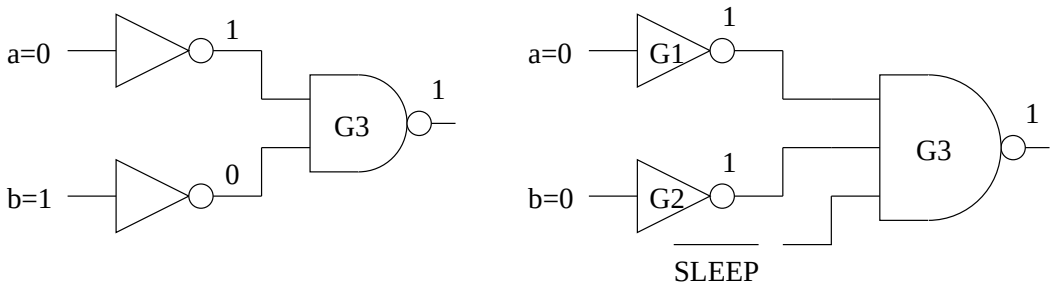
Correctness: We show the correctness of the recursive formula in Equation (2) and (3). To compute $LK(i, z)$, we need to consider all the possible combination of fanins $\{x_{i_1}, \cdots, x_{i_t}\}$ that produces output z at gate G_i. For each such combination, the minimum leakage in the subtree rooted at G_i is the sum of leakage at gate G_i and the minimum leakage at each of its fan-in gate G_{i_j} with output x_{i_j}, $LK(i_j, x_{i_j})$. Equation (2) takes the overall minimum leakage and gives the correct $LK(i, z)$. Assume that this minimum leakage is achieved when G_i has fanins $x_{i_1} = x_{i_1}^0, ..., x_{i_t} = x_{i_t}^0$. Note that $\vec{V}(i_j, x_{i_j}^0)$ is the input vector for the subtree circuit rooted at G_j to produce $x_{i_j}^0$ with the minimum leakage $LK(i_j, x_{i_j})$. The tree structure of the circuit guarantees that the subtrees rooted at $\{G_{i_1}, ..., G_{i_t}\}$ will not share

any common inputs. Therefore, $\vec{V}(i,z)$ is the simple concatenation of $\vec{V}(i_j, x^0_{i_j})$ as given in Equation (3).

Complexity: Equations (1) and (4) take constant time. For each gate G_i, we need to compute $(LK(i,0), \vec{V}(i,0))$ and $(LK(i,1), \vec{V}(i,1))$ by equations (2) and (3). This requires the enumeration of all the 2^t different combinations of G_i's t fanins. For the first time, we need to perform t additions in equation (2). If we enumerate the rest $2^t - 1$ cases following a Gray code, we only need to update $L(G_i(\vec{x}))$(two operations), replace one $LK(i_j, x_{i_j})$ (two operations) and compare the result with the current minimum leakage, a total of five operations. Therefore, we need $t + 5 \cdot (2^t - 1)$ operations for each G_i and this gives a complexity of $O(K \cdot n)$, where K is a constant depending on the largest number of fanins in the circuit.

After obtaining the MLV for the tree circuit, we perform the gate replacement algorithm proposed in Section III to further reduce leakage. Note that, although the MLV is optimal, this does not guarantee us an optimal solution for the MLV+ problem on the tree circuit. For example, consider the circuit in Figure 5.7, the algorithm finds the optimal MLV {a=0, b=1} with leakage $422nA$. Gate 2 is at its WLS and the gate replacement algorithm does not give any improvement. The input vector {0,0} gives the maximum leakage $654nA$; however, when we apply gate replacement technique and replace G_3, the leakage is reduced to $295nA$. In fact, {0,0} is the optimal solution for the MLV+ problem. [2].

[2]We conjecture that the MLV+ problem remains NP-hard for tree circuit. Because we have already lost the optimality when we do the tree decomposition, we will not discuss in details on how to find better solutions to MLV+ on tree circuits. For the same reason, we did not focus on how to improve the fast gate replacement algorithm in Section III-B



minimum leakage = LK(3,1)=422.6nA
MLV = "01"

minimum leakage = LK(3,1)=295.5nA
MLV = "00"

Figure 5.7: MLV in a circuit before and after gate replacement

5.4.4 Connecting the Tree Circuits

In the previous phase, we have determined the output and required input for each individual tree circuit to yield the minimum leakage. The goal of this phase is to combine all the tree circuits to solve the MLV+ problem for the original circuit. The root of each tree circuit may have multiple fanouts that go to other tree circuits as input. Since we treat the tree circuits independently, conflict occurs if the output of a tree circuit and the value required by its fanout gates are not consistent. For example, in Figure 5.8 (a), the circuit is decomposed into three tree circuits T_1, T_2 and T_3. T_1 outputs '1' when its MLV is applied, while T_2 and T_3 require '0' and '1' from T_1 in their respective MLVs. So we have a conflict.

There are basically three ways to resolve this conflict:

(I) enforcing T_1's output at all the fanout gates (Figure 5.8 (b));

(II) changing T_1's output and enforcing this new value at all the fanout gates (Figure 5.8 (c));

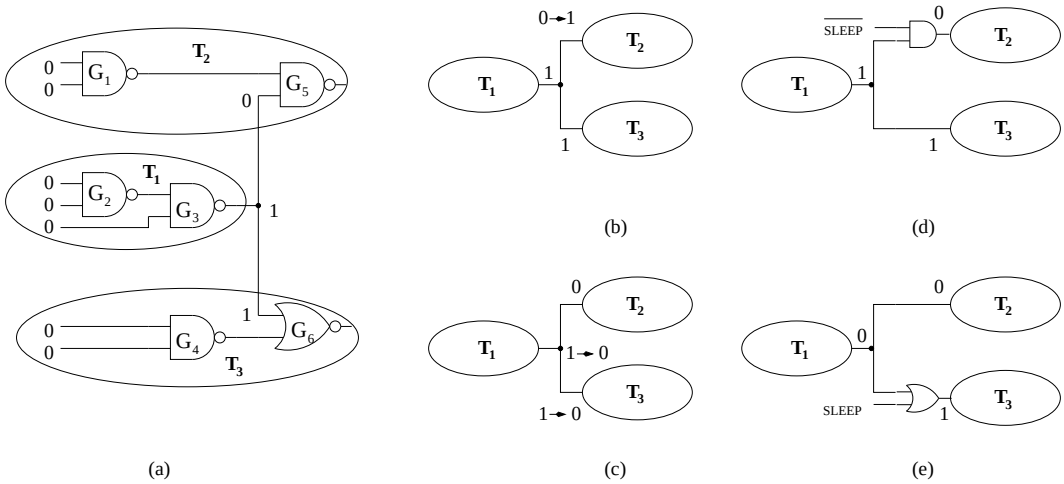


Figure 5.8: Resolving the conflict in connecting tree circuits.

(III) inserting an AND gate to allow them to be inconsistent (Figure 5.8 (d)). Similarly, if T_1 output '0' and some of its fanouts require '1', we can add an OR gate as shown in Figure 5.8 (e)).

To decide which one we should use to resolve the conflict, we apply each of them and re-evaluate the circuit's total leakage. In (I), this requires the re-computing of the minimum leakage and the MLV for tree circuit T_2 under the condition that its input from T_1 is logic '1'. The dynamic programming algorithm in Section IV-B can be trivially modified for this purpose. In (II), we need to do the same procedure for tree circuit T_3. Besides, we have to replace the pair $\{LK(n,1), \vec{V}(n,1)\}$ for tree circuit T_1 by $\{LK(n,0), \vec{V}(n,0)\}$.

Both (I) and (II) resolve the conflict by sacrificing the minimum leakage of tree circuits under the provably optimal MLV. In (III), we successfully connect the tree circuits while preserving the minimum leakage and MLV for each tree with the help of the $SLEEP$ signal-controlled AND or OR gates. The cost is that we have to

add the leakage of the inserted AND or OR gate into the total leakage. We mention that this gate addition also preserves the correctness of the circuit at active mode when $SLEEP$=0.

It is now easy to make a decision on which method to adopt to resolve a single conflict: use the one that gives the minimum leakage. However, the decision at one conflict may affect the existence of conflict at other places in the circuit. For example, method (I) in Figure 5.8 (b) could change the output of tree T_2 and directly affect whether there is a conflict at the root of T_2.

We use a genetic algorithm (GA) to resolve the conflicts and connect all the tree circuits. A solution by the GA is in the form of a binary bit stream, each bit indicates whether there is a conflict at the root of a tree and which method to use to resolve it. In particular, a '1' means there is a conflict and method (III) should be used; a '0' means that there is either no conflict or we should use the better one of methods (I) and (II) to resolve the conflict.

The GA follows a standard routine where we start with a population of N random bit streams (referred to as *chromosomes*). Based on each bit stream, we resolve the conflict, apply the dynamic programming algorithm in Section IV-B to re-compute the minimum leakage of a tree circuit when methods (I) and (II) are used, run the gate replacement algorithm in Figure 5.4 on the entire circuit, and compute the circuit's total leakage. The *fitness* for a bit stream is calculated from the leakage value. The smaller the leakage, the larger the *fitness*. We sort all the chromosomes according to their fitness and create the next generation by the *roulette wheel* method. In this method, the probability that a *chromosome* is selected as one

of the two parents is proportional to its fitness. *Crossover*, which refers to the exchange of substrings in two chromosomes, is performed among parents to produce children. A simple *mutation* operation, which flips a bit in the chromosome at the *bit mutation rate*, is also used. The GA continues to generate a total of N new chromosomes and starts for the next generation. This process repeats for certain number of times (50 in our simulation) and the best chromosome is returned as the optimal solution.

5.4.5 Overhead Analysis

As the control gates are introduced in the tree-connecting stage of the algorithm, they also require sleep signal to control. Hence, we need to consider the extra power these control gates and sleep signal may consume, and their effect on the overall power saving. In this subsection, we will discuss the power overheads.

1) Control gates: The control gates will consume extra dynamic power and leakage power. In this chapter, we only consider the leakage power overhead of the inserted gates and ignore their dynamic power due to the following reasons. First, the number of inserted control gates only accounts for 5% to 6% of the total number of gates in the circuit. Second, they are simple 2-input AND and OR gates, which have a relatively small intrinsic capacitance at the node compared to other gates. Third, the switching activities in these control gates are very limited because one of the two inputs is the sleep signal, which changes only at the moment when the circuit switches between active mode and sleep mode. As dynamic power is dependent

on physical capacitance and switching activities, we consider this dynamic power overhead is negligible.

As for leakage power, we measured the average leakage current in control gates over all possible inputs. In our algorithm, we add this extra leakage current to the objective function, i.e., the overall leakage current to be minimized. Therefore, the leakage saving achieved in our algorithm has already considered this overhead.

2) Sleep signal: Both the gate replacement and the control gates require the sleep signal to drive them during active and sleep mode. The generation of the sleep signal may consume extra power. However, due to the fact that our experiment was conducted at the logic synthesis level before placement and routing, it is not practical to obtain such power data quantitatively. On the other hand, the sleep signal is required by many other leakage minimization techniques, such as [1], [5], and [65]. Hence, in this chapter, we expect the generation of the sleep signal to be similar to those approaches and we believe this problem can be better solved at the physical level of circuit design.

5.5 Experimental Results

We implemented the gate replacement and divide-and-conquer techniques in SIS environment [86] and applied them on 69 MCNC91 benchmark circuits. Each circuit is synthesized and mapped to a 0.18 μm technology library. We use Cadence Spectre to simulate the leakage current for all the library gates under every possible input vector. The supply voltage and threshold voltage are 1.5V and 0.2V, respec-

tively. The measured leakage current includes both subthreshold and gate leakage. The simulations are conducted on a Ultra SPARC SUN workstation.

Our results are compared with traditional input vector control methods in terms of leakage saving, run time, area and delay penalty. The 69 benchmarks including 26 small circuits with 22 or fewer primary inputs (Table 5.1) and 43 large circuits (Table 5.2). For each small circuit, we find the optimal MLV by exhaustive search. For each large circuit, we choose the best MLV from 10,000 distinct random input vectors. It is reported that this will give us a 99% confidence that the vectors with less leakage is less than 0.5% of the entire vector population [32, 77]. To have a fair comparison with [1], we also collect the average leakage of 1,000 random input vectors for each large circuit.

Table 5.1 reports the results for the 26 small circuits. Column 4 lists the leakage current for each circuit when the best MLV is applied. Even in this case, an average of 15% of the gates are at WLS as shown in column 5. The fast gate replacement algorithm is able to move about half of these gates from their WLS (column 7). This results in a 13% leakage reduction with only 4% area increase (columns 6 and 8). We mention that we restrict ourselves to replace only gates off critical paths. This leaves 8% of the gates in the circuits at their WLS, but it also guarantees us that there is no delay overhead.

The last four columns show that the divide-and-conquer algorithm gives a 17% leakage reduction over the best MLV at the cost of 9% more area. We incorporate delay constraints in the genetic algorithm to ensure that the delay overhead to be within 5%. The two columns in the middle are the number of tree circuits in each

Table 5.1: Results on 26 small circuits with 22 or less primary inputs.

circuit	pi	gate	exhaustive		gate replace			divide-and-conquer				
	#	#	leak(nA)	wls	imprv	wls	ar inc	imprv	wls	# tr	cg	ar inc
b1	3	13	2195	23%	2%	15%	5%	2%	10%	5	0%	5%
cm42a	4	25	2941	0%	0%	0%	0%	8%	0%	18	4%	8%
C17	5	6	831	17%	43%	0%	17%	43%	0%	4	0%	17%
cm82a	5	28	5017	21%	29%	4%	12%	40%	1%	10	4%	18%
decod	5	22	1921	0%	0%	0%	0%	8%	0%	21	5%	3%
cm138a	6	19	1760	0%	0%	0%	0%	1%	0%	12	5%	5%
z4ml	7	66	12246	24%	25%	11%	11%	37%	4%	20	5%	17%
f51m	8	136	26038	26%	37%	7%	12%	48%	4%	25	3%	14%
9symml	9	166	34018	26%	20%	17%	5%	38%	8%	18	8%	14%
alu2	10	356	64153	21%	2%	20%	0%	21%	5%	89	7%	11%
x2	10	44	6159	9%	15%	2%	3%	12%	2%	18	9%	10%
cm85a	11	38	4925	8%	14%	3%	3%	13%	3%	16	0%	3%
cm151a	12	34	5745	24%	9%	18%	4%	3%	18%	5	3%	5%
alu4	14	728	133127	25%	1%	21%	1%	15%	4%	166	7%	10%
cm162a	14	45	6947	18%	2%	9%	3%	0%	9%	13	4%	12%
cu	14	49	6182	12%	16%	6%	2%	9%	5%	21	6%	7%
cm163a	16	43	6376	19%	2%	9%	3%	1%	9%	11	5%	13%
cmb	16	42	5405	10%	11%	5%	2%	4%	4%	8	2%	6%
parity	16	75	12764	20%	11%	15%	5%	15%	7%	15	7%	20%
pm1	16	39	3474	3%	0%	0%	1%	-2%	0%	16	3%	3%
t481	16	1945	251184	2%	1%	1%	0%	26%	0%	17	2%	1%
tcon	17	41	6491	20%	43%	0%	14%	41%	0%	9	2%	17%
pcle	19	74	12594	20%	32%	4%	6%	32%	4%	22	0%	6%
sct	19	92	11811	18%	14%	9%	4%	10%	6%	24	4%	6%
cc	21	48	5823	13%	6%	10%	1%	6%	9%	22	0%	1%
cm150a	21	72	12270	15%	4%	14%	1%	1%	10%	9	7%	10%
Average				15%	13%	8%	4%	17%	4%		4%	9%

case and the number of control gates we have used to connect these trees. Only in three cases, we have inserted more than five control gates. Note that the addition of control gates may decrease the delay because it reduces the fanouts of the gate. The area increase comes from the addition of control gates and the replacement of "smaller" gates by "bigger" library gates.

Figure 5.9 reports the leakage and wls gates reduction in the 43 large circuits (x-axis) with 22 PIs or more. We replace the infeasible exhaustive search by the best solution from a random search of 10K input vectors. The fast gate replacement

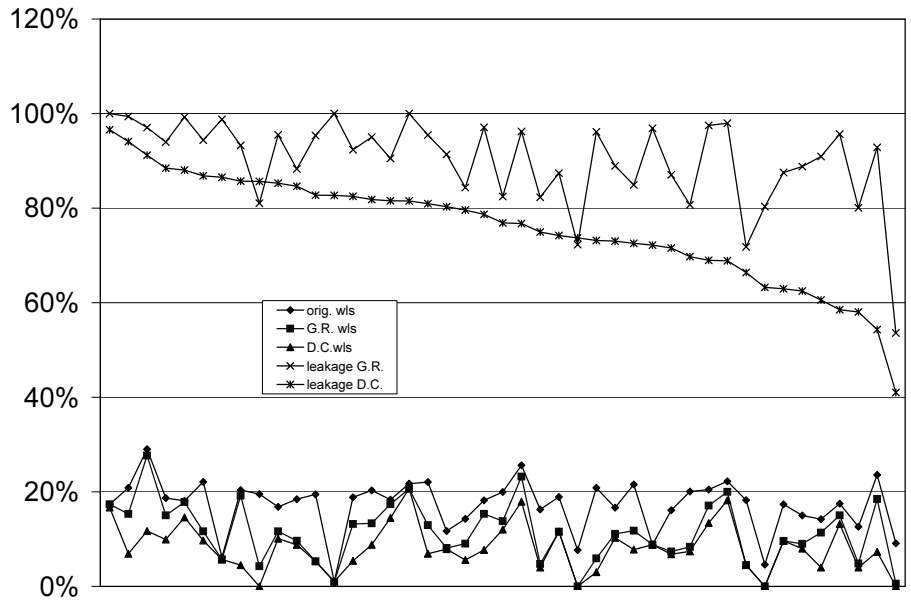


Figure 5.9: Leakage and WLS percentage on 43 large circuits with 22 PIs or more. X-axis lists benchmarks sorted by leakage current in divide-and-conquer approach; Y-axis shows percentage of leakage and WLS gates.

algorithm are restricted only on gates off critical paths; for the divide-and-conquer approach, we set the maximal delay increase to be 5%.

The benchmarks are sorted by the total leakage achieved by the divide-and-conquer method normalized to the best over 10K random search, which is shown one of the two curves at the top part of the figure. The average leakage reductions are 10% by gate replacement only (leakage G.R.) and 24% by divide-and-conquer method (leakage D.C.). The maximal leakage reductions are 46.4% and 60% respectively. The three curves at the bottom give the ratio of WLS gates. On average,

Table 5.2: Results on 43 large circuits with primary inputs more than 22.

circuit	pi	gate	random (10k)		gate replace (G.R.)		div & conq (D.C.)		over 1K average	
	#	#	leak(nA)	time(s)	imp(%)	time(s)	imp(%)	time(s)	G.R.(%)	D.C.(%)
cordic	23	102	18434.0	9.9	15.1	0.01	27.4	10.1	28.4	38.8
ttt2	24	207	33801.5	22.7	9.5	0.02	18.4	72.6	30.9	37.7
i1	25	39	5250.6	5.4	27.7	0	26.3	6.0	45.5	44.4
pcler8	27	90	14670.1	10.0	11.1	0.01	27.0	14.9	35.2	46.8
c8	28	164	26083.0	17.4	19.0	0.01	14.4	21.5	31.7	27.8
C6288	32	2400	480084.2	222.0	2.9	0.11	8.8	398.7	7.0	12.6
comp	32	163	28322.3	15.2	5.6	0.01	13.2	85.4	34.1	39.4
C1908	33	615	117029.6	57.2	2.5	0.02	31.0	66.0	6.4	33.7
my_adder	33	225	40842.1	21.0	2.0	0.02	31.1	32.1	8.9	36.0
term1	34	363	60460.5	37.3	11.7	0.02	15.4	160.0	23.9	27.0
count	35	144	22445.4	15.2	0.0	0.01	3.4	14.2	0.0	15.4
C432	36	200	38101.4	20.1	11.2	0.01	37.5	24.7	21.6	44.8
unreg	36	113	18188.4	12.7	4.6	0.01	17.3	84.4	20.1	30.7
too_large	38	582	107888.1	61.4	12.5	0.05	37.1	80.1	24.5	45.7
b9	41	111	16100.3	12.8	8.6	0.01	19.7	68.0	30.1	38.5
C1355	41	517	91739.0	50.7	4.5	0.02	19.1	95.0	12.1	25.4
C499	41	532	95292.0	48.3	5.0	0.05	18.2	84.5	16.8	28.4
cht	47	232	38560.8	25.3	4.5	0.02	14.7	22.8	18.4	27.1
apex7	49	239	41955.1	26.0	19.3	0.02	30.3	25.6	26.9	36.9
C3540	50	1136	218977.1	115.0	2.9	0.08	21.3	133.8	11.5	28.2
x1	51	295	45351.2	32.8	17.7	0.02	25.0	105.9	32.1	38.2
C880	60	354	61978.8	35.8	12.6	0.04	25.8	39.9	21.7	33.5
dalu	75	1865	349299.8	187.5	3.8	0.15	23.2	194.9	29.1	43.5
example2	85	286	51036.6	32.6	4.3	0.02	41.5	28.9	11.3	45.7
i9	88	510	88469.6	63.9	0.0	0.04	17.3	156.0	0.0	50.1
x4	94	378	61336.3	46.4	28.2	0.03	33.6	206.5	40.1	44.7
i3	132	92	16166.9	14.9	0.0	*	18.5	*	0.0	27.2
i5	133	269	44848.1	34.3	19.9	0.02	42.0	45.6	35.8	53.5
i8	133	1898	305924.5	224.4	9.1	0.15	39.4	7591.3	43.5	62.3
apex6	135	710	126523.6	86.1	3.9	0.06	26.8	399.5	11.4	32.6
rot	135	601	109944.1	67.1	17.5	0.06	23.1	403.3	23.5	28.7
x3	135	742	116641.0	89.5	15.6	0.07	20.4	384.4	29.7	33.7
i6	138	340	47021.1	47.3	46.4	0.03	59.0	89.8	68.9	76.2
frg2	143	1030	165090.4	136.0	12.9	0.11	28.4	176.5	28.0	40.8
pair	173	1538	270729.8	160.9	7.6	0.14	17.5	366.0	14.9	24.0
C5315	178	1777	343295.9	188.3	6.0	0.15	11.5	534.5	11.6	16.8
i4	192	136	22699.8	22.8	3.1	0.01	27.8	34.6	28.3	46.6
i7	199	405	58431.5	58.4	1.2	0.04	13.5	117.9	37.7	45.5
i2	201	109	13174.8	22.1	19.7	0.01	36.8	36.1	36.1	49.7
C7552	207	2801	515320.2	293.3	0.6	0.18	5.9	726.0	20.6	24.8
C2670	233	807	155992.3	94.5	0.8	0.09	11.9	98.6	5.4	16.0
des	256	3995	931447.4	471.2	7.2	0.24	45.7	8502.6	17.6	51.8
i10	257	2281	440552.2	261.6	6.7	0.2	14.3	162.8	11.7	18.8
Average				80.9	10%	0.05	24%	510.2	23%	37%

the 10K random search has 17% gates at WLS(orig, wls); the proposed fast gate replacement and divide-and-conquer techniques reduce this ratio to 11%(G.R. wls) and 9%(D.C. wls), respectively.

More detailed results for these 43 circuits are shown in Table 5.2. Columns 4-5 list the leakage current and runtime when the best MLV from 10,000 random vectors is applied to each circuit. The next two columns show the results when the fast gate replacement algorithm is applied to such best MLV. The average leakage reduction is 10% and the run time is only 0.05 seconds and increases linearly to the number of gates in the circuit.

The next two columns show results by the divide-and-conquer approach where we set a 5% maximum delay constraint. In the genetic algorithm, we start with a population size of $N = 150$ and it converges after 50 generations. We are able to achieve, over the best MLV from 10,000 random vectors, 24% leakage saving. Although the average run time is 6X of the random search, we mention that this is mainly caused by the two circuits, *i8* and *des*. They have a couple of large tree circuits and therefore the frequently called dynamic programming takes considerably long time. Excluding these two circuits, the average run time for random search and the divide-and-conquer algorithm drop to 64.7s and 143s, respectively. More importantly, we see clearly the run time for random search increases exponentially to the number of primary input and linearly to the number of gates (columns 2,3,5). However, the run time for the divide-and-conquer approach grows at a much slower pace (column 9).

The last two columns compare our results with those reported in [1]. Because

Table 5.3: The percent of WLS gates in 43 circuits and the area increase with different input vector control algorithms.

circuit	pi	gate	random 10k	gate replace (G.R.)		div & conq (D.C.)			
	#	#	wls(%)	wls(%)	area inc(%)	wls(%)	# tree	# cg	area inc (%)
cordic	23	102	21.6	11.8	5.7	7.8	52	7	9.3
ttt2	24	207	18.4	17.4	4.4	14.5	43	13	9.6
i1	25	39	7.7	0.0	4.3	0.0	16	1	5.1
pcler8	27	90	16.7	11.1	4.0	10.3	31	0	4.0
c8	28	164	19.5	4.3	8.4	0.0	38	8	6.9
C6288	32	2400	29.0	27.7	1.9	11.7	1424	700	27.3
comp	32	163	22.1	11.7	2.4	9.7	77	4	5.4
C1908	33	615	20.5	17.1	0.9	13.4	218	63	10.1
my_adder	33	225	22.2	20.0	1.5	18.2	95	15	6.4
term1	34	363	18.5	9.6	4.0	8.8	75	18	8.8
count	35	144	17.4	17.4	0.0	16.7	37	3	2.4
C432	36	200	15.0	9.0	3.3	8.0	79	12	8.9
unreg	36	113	19.5	5.3	3.1	5.3	18	2	4.9
too_large	38	582	17.4	9.6	2.2	9.6	113	43	10.9
b9	41	111	11.7	8.1	2.0	7.9	34	4	8.7
C1355	41	517	22.1	13.0	1.4	6.9	265	80	13.1
C499	41	532	20.3	13.3	2.2	8.8	197	39	5.8
cht	47	232	16.8	11.6	3.7	10.1	66	5	3.3
apex7	49	239	20.1	8.4	5.8	7.4	82	8	11.1
C3540	50	1136	18.2	15.3	1.3	7.7	381	174	2.1
x1	51	295	16.3	4.7	4.8	4.0	61	21	11.9
C880	60	354	18.9	11.6	4.1	11.6	115	45	10.8
dalu	75	1865	25.6	23.2	1.4	17.9	321	157	14.2
example2	85	286	17.5	15.0	1.4	13.2	110	7	9.8
i9	88	510	1.0	1.0	0.0	1.0	113	14	2.1
x4	94	378	18.3	4.5	5.3	4.5	110	41	8.6
i3	132	92	21.7	21.7	0.0	20.7	6	0	0.0
i5	133	269	12.6	4.8	2.9	4.0	68	5	7.8
i8	133	1898	14.2	11.4	0.8	4.0	259	110	6.3
apex6	135	710	20.8	5.9	2.1	3.0	215	71	5.7
rot	135	601	20.0	13.8	5.5	12.0	208	58	12.7
x3	135	742	14.3	9.0	3.2	5.6	192	57	10.0
i6	138	340	9.1	0.6	2.1	0.0	71	4	3.0
frg2	143	1030	16.1	7.4	3.2	6.8	244	55	7.4
pair	173	1538	18.9	13.2	2.4	5.4	434	185	12.0
C5315	178	1777	18.7	15.0	2.0	9.9	532	216	15.1
i4	192	136	8.8	8.8	0.4	8.8	6	0	4.6
i7	199	405	6.2	5.7	0.2	5.7	76	7	1.1
i2	201	109	4.6	0.0	2.2	0.0	12	4	3.6
C7552	207	2801	20.8	15.3	0.3	6.9	908	409	16.1
C2670	233	807	18.1	17.8	0.2	14.6	235	89	9.9
des	256	3995	23.6	18.5	2.5	7.3	847	450	14.2
i10	257	2281	20.4	19.2	1.9	4.5	695	319	6.1

their detailed results are not available, we can only compare the average performance. In their experimental setup, the leakage reduction is compared with the average value among 1,000 random vectors. For a fair comparison, we also report in the last two columns the improvement of our approaches over the same baseline.

As we mentioned earlier, most of the leakage currents are contributed by gates in their worst leakage state (WLS). After gate replacement and inserting internal control points, the number of WLS gates will be reduced, however, the area of the circuits may change. Hence, we report the results for these circuits in Table 5.3.

Table 5.4: Average performance comparison with algorithm in [1].

	algorithm in [1]	gate replacement	divide-and-conquer
leakage reduction	25%	23%	37%
delay penalty	$\leq 5\%$	0%	$\leq 5\%$
area penalty	$\leq 15\%$	2%	7%

Finally, Table 5.4 summarizes the performance improvement in the control point insertion approach [1], our gate replacement algorithm, and the divide-and-conquer approach.

5.6 Summary

We study the MLV+ problem which seeks to modify a given circuit and determine an input vector such that the correct functionality is maintained when the circuit is active and the leakage is minimized under the determined input vector when the circuit is at stand-by mode. The relaxation of circuit modification with changing its functionality enlarges the solution space of the IVC method. We show

that MLV (and hence MLV+) problem is a hard problem and propose low-complexity heuristics to solve the MLV+ problem. The proposed algorithms are practical and effective in the sense that we do not need to change the design flow and re-do place-and-route. The experimental results show that this technique improves significantly the performance of IVC in leakage reduction at gate level with little area and delay overhead.

Chapter 6

Energy Efficient Detection Scheme for Wireless Sensor Network Design

6.1 Introduction

Wireless sensor networks (WSNs) are an emerging class of systems with a variety of applications. The advent of small, low-cost, low-power micro-electromechanical sensor technology and low-power RF design has made it possible to conceive of large sensor networks which can perform a comprehensive set of functions. In particular, with the ability of sensor nodes to sense, process, and transmit data, WSNs are well suited to perform event detection mission [99, 100, 15, 110, 67, 84, 107, 13].

In an event detection scenario, sensor nodes are deployed into a target field to collect data. The sensor nodes can process the observed data if needed before transmitting the data to a fusion center, where a final decision is made about whether an event occurs or not. Traditionally, there are two types of detection schemes: a *centralized scheme* requires sensor nodes to forward all the information contained in the observations to the fusion center; while a *distributed scheme*, on the other hand, allows each sensor node to make its own decision and then send out only its 1-bit decision to the fusion center. For both schemes, a final decision will be made at the fusion center based on the information provided by all the sensor nodes.

It is crucial for the sensor network to detect the occurrence of the target event accurately. We define *detection accuracy* as the probability that the fusion center makes the correct final decision, or equivalently, the probability of error in the final decision. Apparently, the more information the fusion center has, the higher the detection accuracy is. Therefore, for the same system parameters, the centralized scheme will achieve the highest detection accuracy and the distributed scheme will have low detection accuracy.

Energy efficiency is another important design concern for WSNs. An energy efficient detection scheme will extend the system's *life time* as sensor nodes usually must rely on small and non-renewable batteries. A sensor node consumes energy in collecting, processing, transmitting, and receiving data. With the communication power dominating in many applications, the distributed scheme is more energy efficient than the centralized scheme as it reduces each sensor node's data transmission to the minimal level (only a 1-bit decision is transmitted). These two schemes' inherent tradeoff between detection accuracy and energy consumption has been investigated in [99].

However, neither of the centralized and distributed detection schemes provides flexibility for WSN designers to choose between detection accuracy and energy consumption. In this chapter, we propose an *energy-driven hybrid* detection scheme to fully exploit the energy-accuracy tradeoff. More specifically, we study how to achieve the required detection accuracy with the least energy consumption. According to the proposed hybrid scheme, each sensor node sends out its 1-bit decision (like the distributed scheme) if that decision exceeds a pre-determined detection accuracy

threshold, and sends out all its observations otherwise (like the centralized scheme). Note that in the former case, a sensor node can stop collecting observations to further reduce energy consumption once it makes its 1-bit decision. The detection accuracy threshold at individual sensor node is selected such that the fusion center is *guaranteed* to achieve the required detection accuracy probabilistically. That is, the probability that the fusion center will make the correct final decision is higher than the required detection accuracy.

Our hybrid scheme is similar to the traditional sequential detection scheme, which implements sequential probability ratio test (SPRT) at the fusion center or sensors [100], in that both schemes operate data processing at sensors adaptively to the collected observations, and hence the number of observations at sensors will be a random variable instead of a fixed value. However, the proposed hybrid scheme sets a restriction for the maximum number of observations collected by each sensor, which avoids the potential delay at sensors and the consequent problem of asynchronism caused by arbitrary large number of observations in the case of sequential detection.

We survey the related WSN work on detection and energy efficiency and explain the novelty of our work in Section II. We describe the WSN model and the two traditional detection schemes in Section III. We introduce the energy-driven hybrid scheme and analyze it in Section IV. Section V presents the WSN's energy consumption model, which includes energy on sensing, processing, transmitting and receiving data. The simulation results reported in Section VI confirm that the proposed hybrid scheme is the most energy efficient to achieve the same detection accuracy. We summarize the chapter in Section VII.

6.2 Related Work

6.2.1 On Detection in Wireless Sensor Networks

For a WSN that performs an event detection function, most of the previous work focus on developing optimal decision rules or investigating the statistical properties for the distributed detection mechanisms. For example, the structure of an optimal sensor configuration has been studied for the scenario where the WSN is constrained by the capacity of the wireless channel over which the sensors are transmitting [15]. Optimum distributed detection system design has been studied in [110] for cases with statistically dependent observations from sensor to sensor. The work in [67] has focused on a WSN with a large number of sensors which is based on a specific signal attenuation model, and the problem of designing an optimum local decision rule has been investigated. In [84] and [107] the problem of binary hypothesis testing using binary decisions from independently and identically distributed sensors is studied, and the optimal fusion rules are obtained.

In this chapter we study the tradeoff between different metrics, detection accuracy and energy efficiency in particular, for the detection scheme's performance. Our main goal is to develop a detection scheme that consumes minimum energy to provide a given detection accuracy. This problem is orthogonal to several other approaches that are designed for objectives other than optimizing decision rules. Therefore, they can be integrated with our proposed detection scheme. For instance, [13] presents node sleeping scheduling protocol to maximize WSN's lifetime with guaranteed detection delay for rare-event detection. [75] studies the minimal

number of sensors required to monitor an environment with a desired sensing accuracy. [61] proposes a random sensor deployment with minimum energy consumption under the constraints of quality of monitoring and WSN lifetime. [46] proposes a tracking method in a WSN of binary proximity sensors.

6.2.2 On Energy Efficiency in Sensor Network Design

As we have mentioned, energy consumption has always been a key concern for WSN design. Many energy-efficient techniques have been proposed in the past, mainly at three design levels: sensor node level, communication level, and network level. Low power design techniques for digital circuits have been used to build low power microsensor nodes [70, 81]. For example, dynamic power management is introduced at system level to reduce energy consumption by turning off idle components in the node [85]. In addition, dynamic voltage scaling technique is used to further reduce the dynamic energy consumption by adjusting the supply voltage at runtime based on the processing workloads of sensors [98, 101].

A lot of work have been reported at communication and network levels to improve WSN's energy efficiency. These include clustering mechanisms [22], routing algorithms [50, 51], energy dissipation schemes [62], sleeping schedules [80] and so on. The energy reduction is normally achieved at the cost of other system performance metrics such as delay [80, 50], robustness [51], or network density [80, 50]. [76] summarizes several energy optimization and management techniques, to enhance the energy awareness of WSNs.

However, performance metrics associated with specific applications have not been adequately studied. As an example, [11] considers the energy-accuracy tradeoff for aggregation applications of a sensor network that performs distributed estimation. Previously we have investigated the energy-accuracy tradeoff for the two traditional detection schemes [99], as well as a sequential detection scheme [100]. The hybrid detection scheme proposed in this chapter improves the energy efficiency of the former two traditional schemes while providing the same detection accuracy. More importantly, this scheme provides WSN designers with the flexibility to trade off accuracy and energy, as well as sensor density.

6.3 System Model

A typical wireless sensor network that performs an event detection mission is shown in Figure 6.1. It consists of a number of sensor nodes and a fusion center. Each sensor node will collect observation data from its neighborhood, process the data if needed, then route the processed data to the fusion center, where a final decision on whether the event occurs or not will be made.

In such a sensor network, usually spatial and temporal correlations exist among observation data within or across sensor nodes; data aggregation occurs along the route from sensor nodes to the fusion center, where information can be partially lost due to compression; and interference is always a problem for the wireless channel. However, to focus our attention on the key issues of detection accuracy and energy efficiency, we assume that each sensor node independently observes, processes

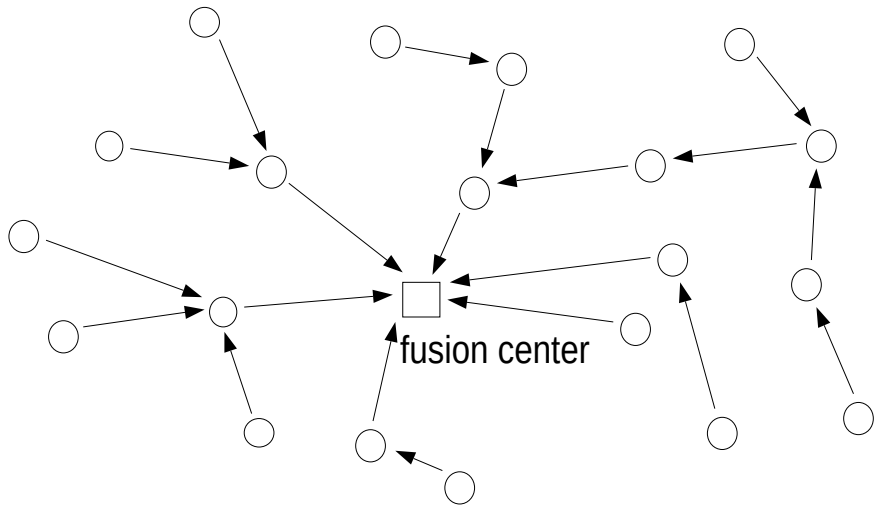


Figure 6.1: Wireless Sensor Network for Detection

and transmits data; given a certain hypothesis, observations are independently and identically distributed (*i.i.d.*) at each single node and across sensor nodes; data is transmitted via multi-hop routing, however on-route sensor nodes simply forward the transmitted data without doing any compression; and there is no noise or any other interference.

We start with the investigation of the binary hypothesis testing. Let H indicate whether an event happens ($H = H_1$) or not ($H = H_0$), with the prior probabilities $P[H = H_1] = p$ and $P[H = H_0] = 1 - p$, $0 < p < 1$. We have K sensor nodes, each collecting T observations. The observations follow Bernoulli distribution conditioned on each hypothesis, with the conditional pmf of $P[1|H_0] = p_0$ and $P[1|H_1] = p_1$. The event can be effectually detected as long as $p_0 \neq p_1$. Hence, WLOG we assume $0 < p_0 < p_1 < 1$. A final decision $\hat{H}$ will be made at the fusion center. Decision errors are penalized through a decision cost function of C_{ij}, which denotes the cost of choosing $\hat{H} = H_i$ when H_j is true, for $i, j \in \{0, 1\}$. For simplicity we assume

uniform costs as $C_{ij} = 0$ for $i = j$ and $C_{ij} = 1$ for $i \neq j$.

Minimizing the overall probability of error, i.e., $P_e = P[\hat{H} \neq H]$, over all admissible decision rules at the fusion center and each sensor node is a simple detection problem that belongs to the classical binary hypothesis testing. Based on the above model, we have studied the following two traditional detection schemes in [1]:

- Centralized Scheme

  At each sensor node, the observation data is transmitted to the fusion center without any loss of information. Since observations are assumed to be conditional *i.i.d.* binary random variables, It is obvious that the number of 1's in the T binary observations at each sensor node (denoted as n_i for the i^{th} node, $i = 1, \ldots, K$) is a sufficient statistic for the detection. The optimal final decision rule at the fusion center is actually the maximum *a posteriori* detector for the binary hypothesis testing [71], which is given by

$$\hat{H} = \begin{cases} H_1 & \text{if } n \geq \gamma_c; \\ H_0 & \text{if } n < \gamma_c. \end{cases} \tag{6.1}$$

  where $n = \sum_{i=1}^{K} n_i$, and the threshold

$$\gamma_c = \frac{ln\frac{1-p}{p} + KTln\frac{1-p_0}{1-p_1}}{ln\frac{p_1(1-p_0)}{p_0(1-p_1)}}. \tag{6.2}$$

  The overall probability of error is given by

$$P_e = p(1 - P[n \geq \gamma_c|H_1]) + (1-p)P[n \geq \gamma_c|H_0], \tag{6.3}$$

  where $P[n \geq \gamma_c|H_\alpha]$ (for $\alpha = 0, 1$) are the detection probability ($\alpha = 1$) and

the false alarm ($\alpha = 0$) that can be computed by

$$P[n \geq \gamma_c | H_\alpha] = \sum_{n=\lceil \gamma_c \rceil}^{KT} \binom{KT}{n} p_\alpha^n (1 - p_\alpha)^{KT-n}. \tag{6.4}$$

- Distributed Scheme

  Each sensor node makes a local decision and transmits the binary result (b_i for the i^{th} node) to the fusion center indicating its decision. We assume that sensor nodes are homogeneous, thus each sensor node will adopt the same detector, and the local decision rule does not depend on the total number of sensor nodes, which is considered as a global information. Therefore similar to the centralized scheme, both the local detector at each sensor node and the final detector at the fusion center will be maximum *a posteriori* detector. The detailed results can be found in [99], we do not elaborate them here.

The centralized and distributed schemes are easy to implement and can be conveniently analyzed theoretically. However, they do not provide any flexibility in detection accuracy and energy consumption. That is, for fixed system parameters the performance of the schemes, including detection accuracy and energy consumption, is fixed for the application users. In the following section we propose a hybrid detection scheme that can achieve flexibility between energy and accuracy. When we fix the detection accuracy, the energy-driven hybrid detection scheme can achieve significantly higher energy efficiency than the above traditional schemes.

6.4 Hybrid Detection Scheme

6.4.1 Intuition

When a single node is considered, the number of 1's (denoted by n_i) in the T binary observations indicates the confidence in the decision. That is,

$$P[H_1|n_i] = \frac{P[n_i|H_1]P[H_1]}{\sum_{\alpha=0}^{1} P[n_i|H_\alpha]P[H_\alpha]} = \frac{p\lambda_i}{p\lambda_i + 1 - p}, \tag{6.5}$$

where the *likelihood ratio* at the i^{th} node is given by

$$\lambda_i = \frac{P[n_i|H_1]}{P[n_i|H_0]} = [\frac{p_1(1-p_0)}{p_0(1-p_1)}]^{n_i}(\frac{1-p_1}{1-p_0})^T. \tag{6.6}$$

As we assume $0 < p_0 < p_1 < 1$, the *likelihood ratio* increases monotonically with n_i, so does $P[H_1|n_i]$, which means the more number of 1's a sensor node observes, the more confident it is to decide $\hat{H} = H_1$; and vice versa. In other words, the accuracy of decision $\hat{H} = H_1$ increases with the increase of n_i; or equivalently, the accuracy of decision $\hat{H} = H_0$ increases with the decrease of n_i.

Therefore, to achieve a certain level of accuracy at a sensor node, we only need to collect a minimum number of 1's (N_1) or a minimum number of 0's ($T - N_0$), where $0 \leq N_0, N_1 \leq T$. From the point view of energy efficiency to guarantee such a detection accuracy at the sensor node, it is sufficient to stop collecting new data once N_1 1's or $T - N_0$ 0's have been observed. This has the potential in saving energy from data collecting and processing. If neither N_1 1's or $T - N_0$ 0's are accumulated in all the T observations, the sensor node simply transmits all the data to the fusion center in order not to lose any information.

6.4.2 Detection mechanism

The hybrid detection scheme allows each sensor node to compare the number of 1's in its T binary observations (n_i for the i^{th} node) with two threshold N_0 and N_1 ($N_0 < N_1$). If $n_i \leq N_0$ the sensor node will send a 0 to the fusion center as the local result; or if $n_i \geq N_1$ a 1 will be sent; otherwise the original information from observations will be sent. In other words, if a sensor node observes enough 1's or 0's, which can guarantee a certain accuracy, it will make a local decision and send it to the fusion center, similar to the distributed scheme; otherwise it will send all the information to the fusion center, similar to the centralized scheme. A final decision will be made at the fusion center based on all the information provided from the sensor nodes. Therefore the new scheme can be considered as a hybrid of the two traditional schemes.

Obviously if $N_0 < 0$ and $N_1 > T$, sensor nodes will always perform the centralized scheme; if $N_1 - N_0 = 1$, sensor nodes will always perform the distributed scheme. Therefore these two traditional schemes can be considered as two extreme cases of the hybrid scheme.

Notice that for the hybrid scheme, each sensor node does not necessarily observe all of the T observations before they can make a local decision. As soon as the number of 1's reaches N_1, it can decide to send a 1; or on the other hand, as soon as the number of 0's reaches $T - N_0$, it can decide to send a 0. In either case, the data observing process will possibly be terminated before the total number of observations reaches T. Therefore, the hybrid scheme can potentially save energy

in data observing and processing. We will validate this intuition by simulations.

The mechanism of the hybrid scheme at each sensor node can be generalized as follows:

$$Y_1^j \rightarrow \begin{cases} \#(1) \geq N_1 & \rightarrow \text{send } b = 1 \\ \#(0) \geq T - N_0 & \rightarrow \text{send } b = 0 \\ \text{otherwise} & \rightarrow \text{take } Y_{j+1} \ldots \text{until } Y_T, \text{ send } n \end{cases} \tag{6.7}$$

where Y_j is the j^{th} observation collected at this sensor node, $j = 1, 2, \ldots, T$; Y_1^j stands for $\{Y_1, Y_2, \ldots, Y_j\}$; b is the binary local decision; and n is the number of 1's out of the T observations.

6.4.3 Decision rules

Similar to the distributed scheme, we adopt identical detectors, which means the thresholds of $\{N_0, N_1\}$ are the same for all sensor nodes. This is reasonable since we study homogeneous sensor networks, and it will significantly reduce the computation complexity. We first consider local decision rule, then address the approach to determine optimal final decision rule.

- Local decision rule

  Our object is to determine the local thresholds, i.e., $\{N_0, N_1\}$, that can guarantee a certain detection accuracy for the final decision at the fusion center, e.g., $P_e \leq \delta$. Here we introduce an approach that finds an upper bound for P_e.

  First, we loose P_e to P_{e-dis}, which represents the overall probability of error

of the case that all sensor nodes send a 1-bit decision. We have:

$$P_{e-dis} = (1-p)P[\hat{H} = H_1|H_0] + pP[\hat{H} = H_0|H_1].$$

Then to ensure $P_{e-dis} \leq \delta$, we enforce $P[\hat{H} = H_1|H_0] \leq \delta$ and $P[\hat{H} = H_0|H_1] \leq \delta$, such that

$$\begin{aligned} P[\hat{H} = H_1|H_0] &= \sum_{k=\lceil \Gamma_D \rceil}^{K} \binom{K}{k} P[b=1|H_0]^k \\ &\quad (1 - P[b=1|H_0])^{K-k} \leq \delta, \end{aligned} \tag{6.8}$$

where b is the local decision at each node, and Γ_D is the threshold that can be computed. Because we assume $P[b=1|H_0] < 1/2$, Equation (6.8) can be further simplified as

$$K\binom{K}{\lfloor K/2 \rfloor} \frac{P[b=1|H_0]}{1 - P[b=1|H_0]} (1 - P[b=1|H_0])^K \leq \delta.$$

Next considering $(1 - P[b=1|H_0])^K < 1$, we have

$$K\binom{K}{\lfloor K/2 \rfloor} \frac{P[b=1|H_0]}{1 - P[b=1|H_0]} \leq \delta.$$

Finally we obtain an upper bound on the accuracy of the local decision, given by

$$P[b=1|H_0] \leq \frac{K\binom{K}{\lfloor K/2 \rfloor}}{K\binom{K}{\lfloor K/2 \rfloor} + \delta}. \tag{6.9}$$

Similarly we can derive the upper bound for $P[b=0|H_1]$.

As we know

$$P[b=0|H_\alpha] = \sum_{i=T-N_0}^{T} \binom{i-1}{T-N_0-1} (1-p_\alpha)^{T-N_0} p_\alpha^{i-T+N_0}; \tag{6.10}$$

$$P[b=1|H_\alpha] = \sum_{i=N_1}^{T} \binom{i-1}{N_1-1}(1-p_\alpha)^{i-N_1} p_\alpha^{N_1} \tag{6.11}$$

for $\alpha = 0, 1$. From Equations (6.9), (6.10), (6.11), we will be able to determine $\{N_0, N_1\}$ as a function of $\{\delta, K, T, p_0, p_1\}$.

Although the upper bound can be loose, it demonstrates the idea that an overall accuracy of the system is dependent on the local thresholds at each sensor node.

- Final decision rule

  Suppose among the K sensor nodes, s of them send a '0' and another t of them send a '1', which leaves the other $K-s-t$ sensor nodes sending their n_i's, where $s, t \geq 0$ and $s + t \leq K$. WLOG let $\Omega = \{n_1, \ldots, n_{K-s-t}; 0, \ldots, 0; 1, \ldots, 1\}$ denote the set of data transmitted to the fusion center from sensor nodes.

  Then for a given Ω, the optimal final decision rule is to choose $\hat{H} = H_1$ if

$$P[H_1|\Omega] \geq P[H_0|\Omega]. \tag{6.12}$$

  Equation (6.12) can be derived to

$$\frac{P[\Omega|H_1]}{P[\Omega|H_0]} = \prod_{k=1}^{K-s-t} \frac{P[n_k|H_1]}{P[n_k|H_0]} \times B_0^s \times B_1^t \geq \frac{1-p}{p}, \tag{6.13}$$

  where

$$B_i = \frac{P[b=i|H_1]}{P[b=i|H_0]}; i = 0, 1.$$

  $P[b=i|H_\alpha]$ are given by Equations (6.10), (6.11).

  We can also compute

$$P[n_k|H_\alpha] = \binom{T}{n_k} p_\alpha^{n_k} (1-p_\alpha)^{T-n_k} \tag{6.14}$$

for $\alpha = 0, 1$.

The overall probability of error can be computed by Equation (6.3), where the detection probability (P_d) and false alarm (P_f) can be expressed as

$$\begin{aligned} P_d &= \sum_{\Omega:\hat{H}(\Omega)=H_1} \binom{K}{s}\binom{K-s}{t}(P[b=0|H_1])^s \\ &\quad \times (P[b=1|H_1])^t \times \prod_{k=1}^{K-s-t} P[n_k|H_1]; \end{aligned} \tag{6.15}$$

$$\begin{aligned} P_f &= \sum_{\Omega:\hat{H}(\Omega)=H_1} \binom{K}{s}\binom{K-s}{t}(P[b=0|H_0])^s \\ &\quad \times (P[b=1|H_0])^t \times \prod_{k=1}^{K-s-t} P[n_k|H_0]. \end{aligned} \tag{6.16}$$

Figure 6.2 presents the detection performance of the hybrid scheme with optimal decision rule applied, and it is compared with the centralized and distributed schemes. The system parameters are set as: $p = 0.5, p_0 = 0.2, p_1 = 0.7, T = 5$; K is varied from 5 to 10; and hybrid scheme of $\{N_0 = 1, N_1 = 3\}, \{N_0 = 1, N_1 = 4\}, \{N_0 = 0, N_1 = 4\}$ are examined. As we can see, for given values of parameters, the hybrid scheme performs significantly better than the distributed scheme, and it is comparable to the centralized scheme. Also, it is improved with the increase of the distance between N_0 and N_1. As long as the desired detection accuracy falls into the range between the results of the centralized and distributed schemes, the hybrid scheme can always guarantee to achieve the same accuracy by setting $\{N_0, N_1\}$ accordingly. Furthermore, we will see from the simulation results that the hybrid scheme consumes much less energy than the other two schemes to achieve the same detection accuracy.

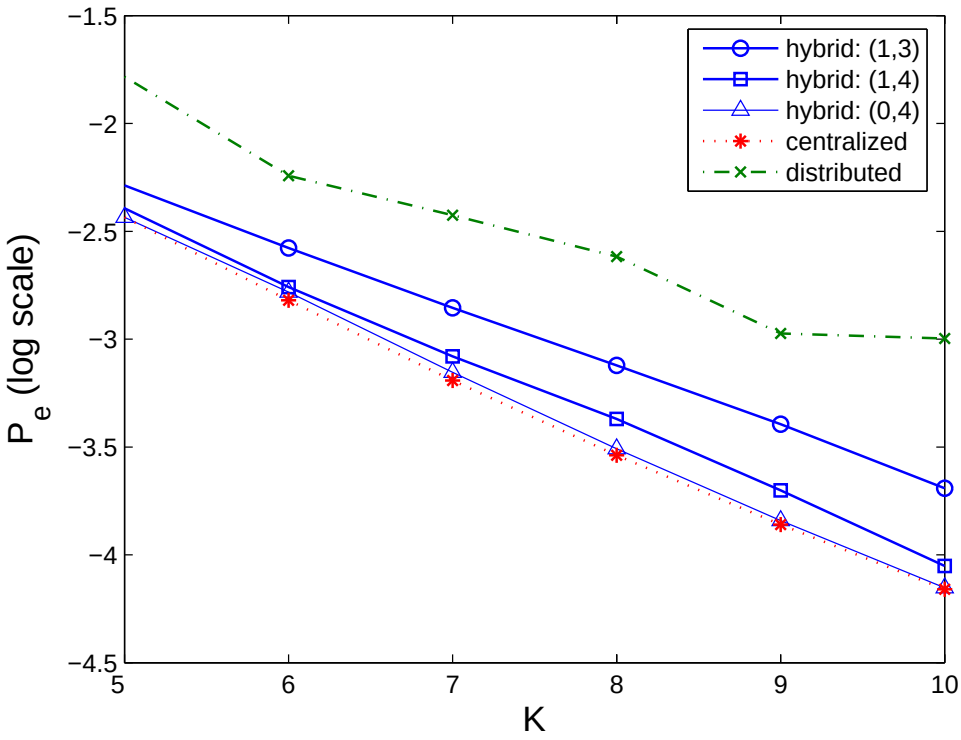


Figure 6.2: Comparison of Three Schemes in Detection Accuracy

6.4.4 Suboptimal algorithm

Determining the optimal detection performance by exhaustive search in Equations (6.15), (6.16) may not be practical due to the excessive computations, especially for large K and T. Hence we develop a suboptimal algorithm to compute an approximate result for the detection performance.

In the suboptimal algorithm, instead of computing $P[n|H_\alpha]$ for each different $n \in \{N_0+1, \ldots, N_1-1\}$, we replace it with its average value, which is given by

$$P_{avg}[n|H_\alpha] = \frac{\sum_{i=N_0+1}^{N_1-1}(P[n=i|H_\alpha])^2}{\sum_{i=N_0+1}^{N_1-1} P[n=i|H_\alpha]}; \alpha = 0,1 \tag{6.17}$$

where $P[n=i|H_\alpha]$ can be computed by Equation (6.14). Thus Equation (6.13) can

be simplified to

$$(\frac{P_{avg}[n_k|H_1]}{P_{avg}[n_k|H_0]})^{K-s-t} \times B_0^s \times B_1^t \geq \frac{1-p}{p},$$

and Equations (6.15), (6.16) can be simplified in the same way.

6.5 Energy Consumption Model

Energy is mainly dissipated on three parts in our network: data acquisition, data processing and communication.

6.5.1 Data acquisition

Data acquisition includes sensing observations from the field and converting the observed information into digital format. Each sensor node collects observations via a sensing device. The power consumption for sensing is dependent on the types of sensing devices. For example, the MICA Mote microsensors [35] have three different sensing devices: photoresister, accelerometer and temperature meter. Their power consumption ranges from 0.4mW to 13.5mW. After an observation is collected, the analog-digital converter (ADC) in the sensor node will convert the information into a binary value and store it into a local memory.

Because the observations in our scenario are discrete, we can assume the sensing device will be turned on and off periodically to sample the data and store it locally. For one observation, we measure the total energy consumed in sensing, converting, and storing, and denote this unit energy consumption as e_s. We also assume, after the sensing device is turned off, no energy will be dissipated.

For the T observations collected from the sensing field, both the centralized and the distributed detection schemes will sample T times. Therefore, the sensing energy in each sensor node will be Te_s. For the hybrid scheme, sensor nodes will keep collecting observations until they can make a decision or all the T observations are collected. As we have described in Section IV, given the local thresholds N_0 and N_1, the number of observations collected at each node will be between $\min\{N_1, T - N_0\}$ and T. Therefore, the sensing energy at a node for the hybrid scheme will be $e_s\tau$, where $\tau \in [\min\{N_1, T - N_0\}, T]$.

6.5.2 Data processing

Two major contributors to energy consumption in data processing are dynamic power and leakage power in the sensor node's microprocessor (or microcontroller). Dynamic power is caused by the capacitance charging and discharging and is proportional to $C_s V_{dd}^2$, where C_s is the total switching capacitance in the microprocessor and V_{dd} is the supply voltage. Leakage power is caused by the leakage current in the CMOS circuits and is dependent on the threshold voltage V_{th} and thermal voltage V_T of the circuits. During the active mode, both dynamic and leakage power are present; while in the idle mode, the leakage power is the dominant part. As the leakage power increases dramatically in the past few years and is projected to be more and more significant in the future, most microprocessors have incorporated one or more *sleep* modes, in which the leakage power consumption is tiny. However, when the microprocessor switches from *sleep* mode back to active mode, an additional

time and energy is required for waking up. Such wake-up time in microprocessors is usually in a few to hundreds of cycles and the energy is between 2 and $45nJ$ [72].

In our system model, we assume that the observations are sparse in the time domain and the inter-arrival time is much longer than the wake-up time plus data processing time. Therefore, for energy efficiency, we put the microprocessor into *sleep* mode as soon as it finishes data processing. Assuming that the energy overhead for wake-up is e_w and the microprocessor needs n cycles to process an observation, with e_{cyc} as the energy consumption for each cycle, the energy consumed to process each observation will be $e_p = e_w + ne_{cyc}$. For the centralized scheme, since no data processing is needed, the data processing energy is virtually zero. For the distributed scheme, each sensor node needs to spend T cycles to process data and one additional cycle to make a final decision. In this case, the total energy is $(T+1)e_p$. For the hybrid scheme, the processor will process each incoming observation until it can make a decision, otherwise all the T observations will be processed. Similar to the sensing energy, the amount of processing energy will depend on the number of observations it processes, thus it will be between $\min\{N_1, T - N_0\}e_p$ and Te_p.

6.5.3 Communication

We consider multi-hop communications in our network model, where a greedy perimeter stateless routing (GPSR) algorithm is implemented. Each sensor node will receive data from and send data to its neighbors within a communication radius, denoted by R. The energy for transmitting and receiving one bit information is

modeled as:

$$E_{tx} = e_t d^{\beta}; E_{rx} = e_r, \tag{6.18}$$

where e_t is the energy of transmitting one bit data over a unit distance; e_r is the energy for receiving one bit information from a neighbor; d is the distance between two neighboring nodes; and β is the path loss exponent, which is an environment-dependent constant, usually between 2 and 4. We adopt $\beta = 2$ in our simulation.

When a sensor node sends packets to the fusion center via a routing path, all the nodes along that path will receive and forward the packets. Assuming that the i^{th} node sends S_i bits of data to the fusion center, we can calculate the communication energy consumed on this path as

$$E_{tx-i} = \sum_{j=1}^{l_i} e_t d_j^2 S_i; E_{rx-i} = \sum_{j=1}^{l_i} e_r S_i, \tag{6.19}$$

where l_i is the length of the path, in number of hops, from the i^{th} node to the fusion center, and d_j is the distance between the $j-1^{th}$ node and the j^{th} node on the path.

For the centralized scheme, every sensor node sends the number of 1's of the collected data to the fusion center. Therefore $S_i = log_2(T+1)$ for all the nodes. For the distributed scheme, all the sensor nodes send only their one bit decision to the fusion center, thus $S_i = 1$ in this case. For the hybrid scheme, if a node has made a decision locally, it only needs to send one bit data out; otherwise, it will need to send out the $log_2(T+1)$ bit data to the fusion center.

6.6 Simulation Results

We have already demonstrated that the proposed hybrid scheme's detection accuracy is significantly higher than the distributed scheme and can be comparable to the centralized scheme. In this section, we conduct simulation to validate the energy efficiency of the hybrid scheme.

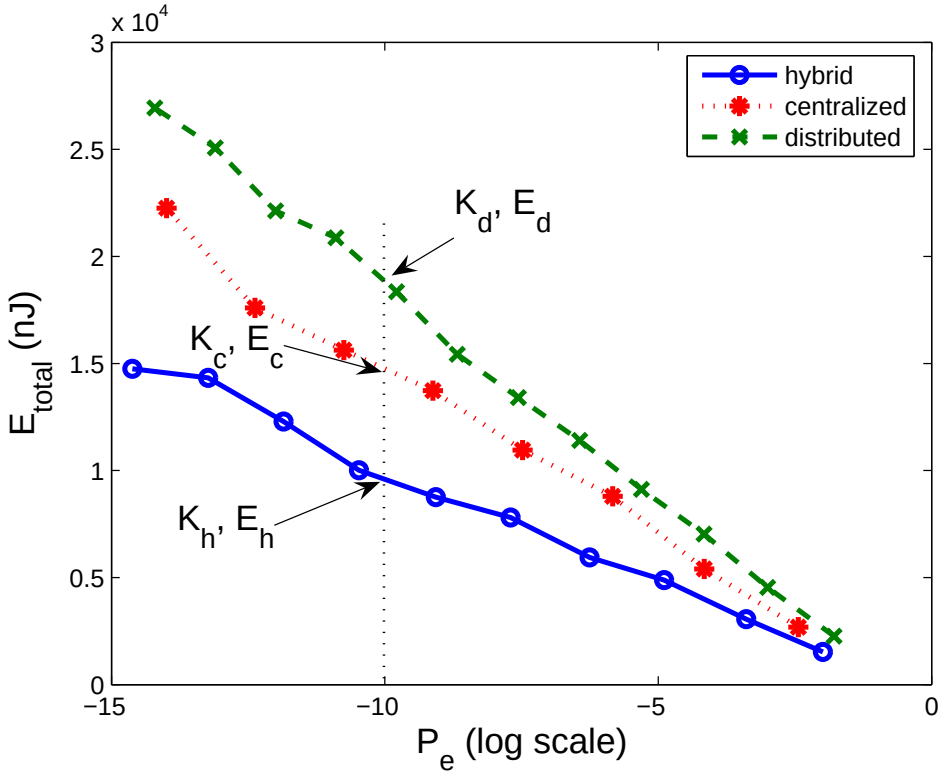


Figure 6.3: Dense Network: 1×1 Field

In our simulation, the probability p that the event happens changes from 0.1 to 0.9 with a step of 0.2, the conditional probability that 1 is observed when the event does not occur (p_0) and when the event does occur (p_1) varies from 0.1 to 0.3 and 0.7 to 0.9, respectively, both with the step of 0.1. Here we report the results under the representative parameter setting: $p = 0.5, p_0 = 0.2$ and $p_1 = 0.7$. It is

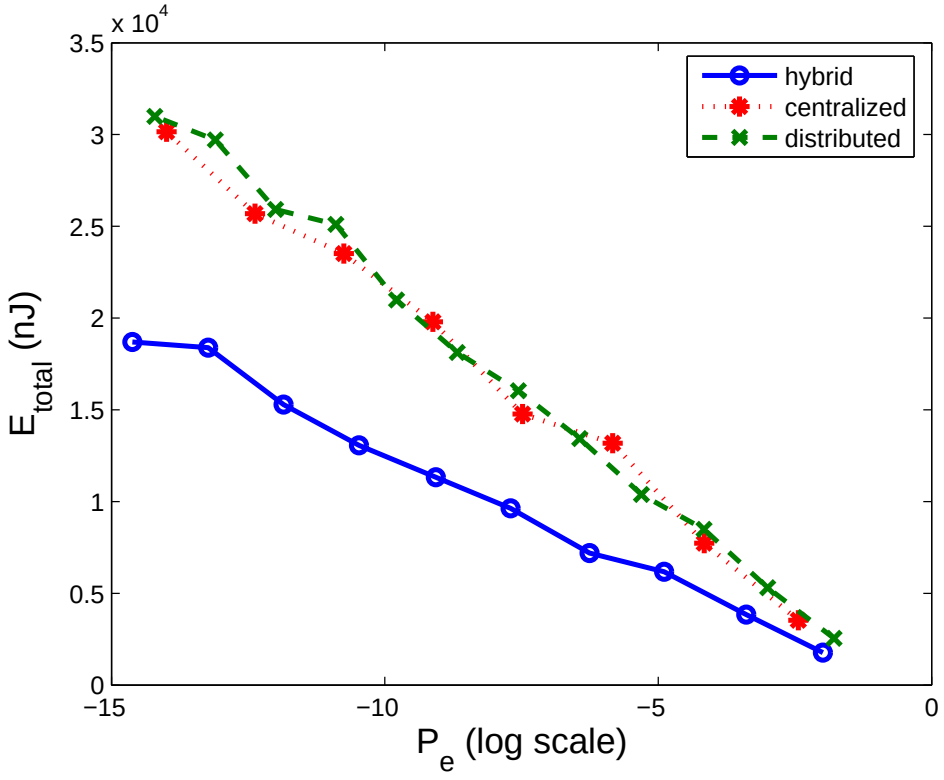


Figure 6.4: Intermediate Network: $\sqrt{2} \times \sqrt{2}$ Field

easy to see that as p_0 decreases (and/or p_1 increases), each observation becomes more accurate. So we can set a small distance between the two thresholds in the hybrid scheme to reach the given detection accuracy. As a result, the sensor node has a better chance to make its own decision and transmit only this 1-bit decision. Consequently the hybrid scheme becomes more energy efficient.

We adopt the following energy parameters: transmitting 1 bit over a unit distance needs $e_t = 400nJ$, receiving 1 bit from a neighbor node needs $e_r = 50nJ$, taking 1 bit observation requires $e_s = 10nJ$, and processing one observation consumes $e_p = 40nJ$. We report the results when each sensor node takes 5 observations (i.e., $T = 5$) and the local thresholds in the hybrid scheme are set as $N_0 = 1$ and

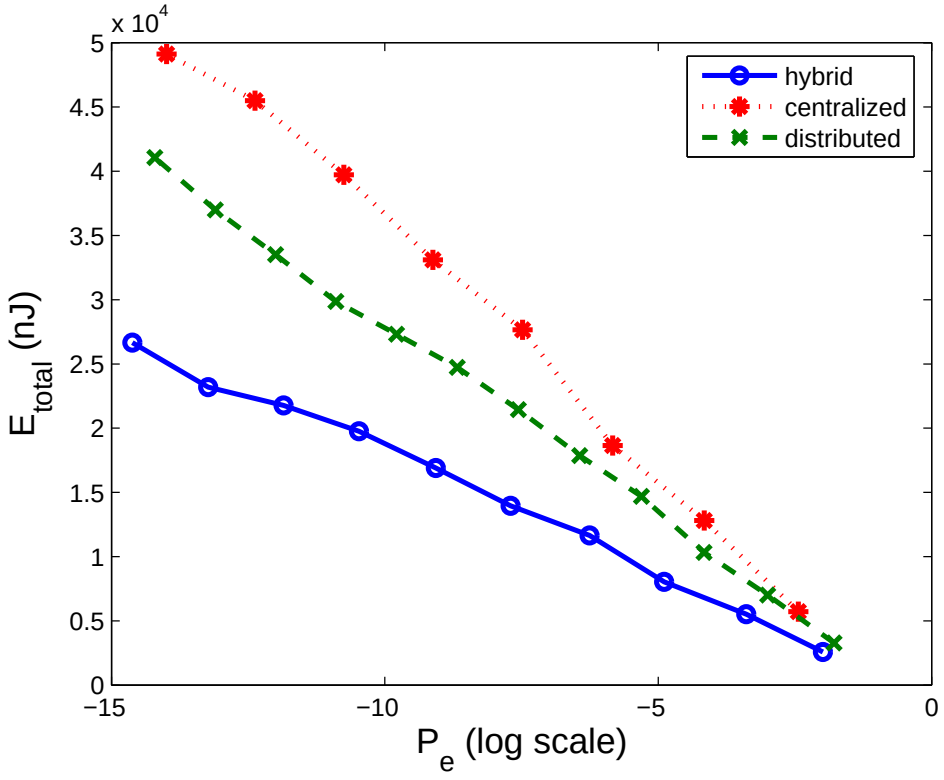


Figure 6.5: Sparse Network: 2×2 Field

$N_1 = 3$.

As we have discussed in Section IV, with the same number of sensor nodes, the three different detection schemes achieve different detection accuracy. In our simulation, we measure the detection accuracy by the overall probability of error P_e and let it vary from 10^{-2} (lowest accuracy) to 10^{-14} (highest accuracy). The minimal number of sensors required to reach P_e varies accordingly from 4 to 41, 6 to 48, and 6 to 59 for centralized, hybrid, and distributed detection schemes, respectively. For instance, when $P_e = 10^{-10}$, these three schemes need $K_c = 28$, $K_h = 34$, and $K_d = 41$ sensors, respectively. For a given detection accuracy, we randomly deploy the minimal number of sensors required by each detection scheme

in a unit square (1×1) field. We set the communication radius $R = 0.45$ and use the GPSR [44] protocol to route the data from sensor nodes to the fusion center.

Now, with the energy model presented in Section V, we are able to compute the energy consumption for each detection scheme as depicted in Figure 6.3. Clearly, the proposed hybrid scheme is more energy efficient than the centralized and distributed schemes. For example, when $P_e = 10^{-10}$, the sensor network with hybrid scheme consumes a total energy of $E_h = 10292.9nJ$, while centralized and distributed schemes require $E_c = 16714.3nJ$ and $E_d = 18988.3nJ$, or **62.4%** and **84.5%** more energy than E_h, respectively. Comparing with the centralized scheme, our hybrid scheme saves a large amount of energy on communications because it can make local decision and convert a lot of data into the 1-bit decisions. On the other hand, the distributed scheme's poor detection performance forces it to deploy more sensors ($K_d = 41$) which results in more energy cost on sensing and local data processing. This energy breakdown is shown in Figure 6.6.

To study the impact of the communication cost on the energy performance of these detection schemes, we scale the size of the sensor field to change the energy per bit per communication hop. Figure 6.4 depicts the result when the sensor field is doubled, and R is changed to 0.64. Figure 6.5 presents the result when the edge of the field is doubled, i.e., field is four times large, and R is changed to 0.9. As one can see, with communication becomes more expensive, the total energy cost for all schemes increases. However, the centralized scheme has the fastest increase as it has the most communication activity. Thus its performance is deteriorated most and it becomes less energy efficient than the distributed scheme. On the other hand,

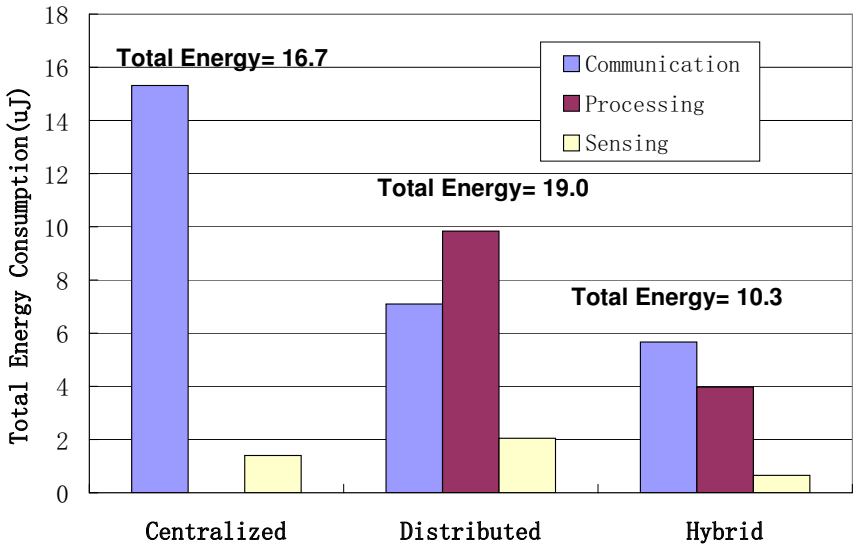


Figure 6.6: Breakdown of Energy Consumption for Dense Network

the proposed hybrid scheme still has the best performance because its increased cost in communications is less than that of the other two schemes when the same detection accuracy is achieved. In sum, the hybrid scheme can reduce total energy consumption significantly over both traditional schemes.

Finally, we compare the energy consumption per node to provide the desired detection accuracy. This is another important metric to evaluate WSN's energy efficiency, particularly when sensor's energy source is not renewable or rechargeable. As one can see from Figure 6.7, the centralized scheme has the highest energy per node cost. This implies that under the current simulation setting, a WSN with centralized scheme, despite consuming less total energy than a WSN with distributed scheme (see Figure 6.3), has a shorter lifetime. Figure 6.7 shows that the proposed

hybrid scheme performs distinctly better than the two traditional schemes in terms of energy per node, which results in a much longer lifetime for the WSN.

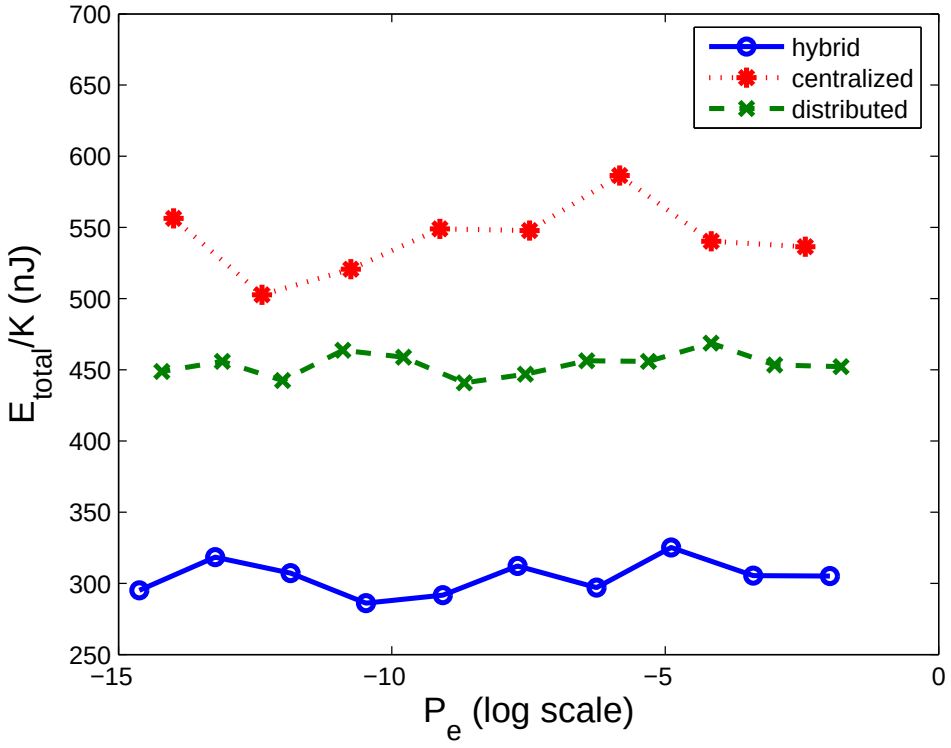


Figure 6.7: Comparison of Energy per Node for Dense Network

6.7 Summary

In this chapter we propose an energy-driven hybrid detection scheme that can reduce energy consumption while providing a guaranteed detection accuracy. We terminate the detection process on individual sensor node when a local decision can be made with accuracy higher than a pre-determined threshold. Such threshold is calculated to ensure that the overall detection accuracy will be achieved at the fusion center. We have developed the optimal decision rule for the proposed hybrid

detection scheme, and assessed its detection performance. We have also constructed an energy consumption model to estimate the energy consumption for the system. Simulation results confirmed that the hybrid scheme can significantly reduce the energy consumption than the traditional schemes to achieve the same detection accuracy.

The proposed hybrid scheme provides WSN designers with the flexibility in balancing different performance metrics. The inherent relation between the overall accuracy and local thresholds needs further investigations. Moreover, we are currently studying a more general WSN model to investigate the energy efficiency and the detection accuracy of the hybrid scheme. Many factors such as non-binary data, spatial and temporal correlation among observations, and data aggregation along routes are being considered in this general model. Our long term goal is to build a framework to study, across all WSN design levels, the minimum energy required to perform a given function.

Chapter 7

Conclusions

In this thesis, we address the increasing concern of power consumption in modern embedded system design. We propose a *re-engineering* design methodology to explore the ever-increasing design space of energy efficient system more efficiently and effectively.

This methodology is based on the observation that existing design space exploration algorithms may exclude good solutions in the early stage of the serial design flow. By re-constructing the design space and re-exploration, the solution quality can be improved.

A general design framework using this methodology is given. To keep the discussion concrete, we also apply this framework to four power minimization problems at different levels of embedded system design flow.

In the sequential circuit synthesis problem, we re-construct the FSM by duplicating states. State encoding in the enlarged FSMs can provide solutions with smaller switching activity, i.e., reduced dynamic power consumption. Our experiments on MCNC benchmarks have shown a 12% power reduction.

In the dual-V_{th} assignment problem, we combined the procedure of input vector control followed by V_{th} assignment in leakage power minimization. We developed an iterative algorithm to simultaneous assign input vectors and V_{th}. The performance

of dual-V_{ht} assignment is improved by over 30% with the same run-time.

In the input vector control for static power minimization problem, we re-engineer the technology mapping solution. Basically, we replace gates that are in the worst leakage states by other library gates to reduce the worst case leakage. A sleep signal is used to control the correct functionality at active mode. Our approach can achieve 50% more leakage reduction with less overhead.

In the energy efficient wireless sensor network design, this re-engineering idea is used to design the detection scheme at system level. A hybrid scheme is proposed to trade off detection accuracy for energy efficiency. We run simulation on networks with a specific application. Our scheme consumes the least energy with guaranteed detection accuracy.

Through these problems, we show that the re-engineering design framework can be practically integrated into today's embedded system design flow and improve design's energy efficiency. It will be a promising approach to solving other low power problems. Moreover, it is possible to apply this methodology to other optimization problems than power minimization. It can be another research direction in the future.

Appendix A

List of Publications

1. Lin Yuan and Gang Qu, Energy Efficient Design for Distributed Sensor Networks, Handbook of Sensor Network, Chapter 38, CRC Press, Oct. 2004.

2. Lin Yuan and Gang Qu, Analysis of Energy Reduction on Dynamic Voltage Scaling-Enabled Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 24, No. 12, pp. 1827-1837, Dec. 2005.

3. Lin Yuan and Gang Qu, A Combined Gate Replacement and Input Vector Control Approach for Leakage Current Reduction, IEEE Transactions on VLSI, Vol. 14, No. 2, Feb. 2006.

4. Lige Yu, Lin Yuan, Gang Qu and Anthony Ephremides, Energy-Driven Detection Scheme with Guaranteed Accuracy, to appear in International Conference on Information Processing in Sensor Networks (IPSN), 2006.

5. Sean Leventhal, Lin Yuan, Neal Bambha, Shuvra Bhattacharyya and Gang Qu, DSP Address Optimization Using Evolutionary Algorithm, 9th International Workshop on Software and Compilers for Embedded Systems (SCOPES), 2005.

6. Lin Yuan and Gang Qu, Enhanced Leakage Reduction Techniques by Gate Replacement, 42nd IEEE/ACM Design Automation Conference (DAC), 2005.

7. Lin Yuan, Gang Qu and Ankur Srivastava, VLSI CAD Tool Protection by Birthmarking Design Solutions, IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI), 2005.

8. Lin Yuan, Gang Qu, Tiziano Villa, and Alberto Sangiovanni-Vincentelli, FSM Re-Engineering and Its Application in Low Power State Encoding, Asia South Pacific Design Automation Conference (ASPDAC), pp. 254-259, 2005.

9. Lin Yuan and Gang Qu, FSM Re-Engineering for Low Power State Encoding, International Workshop on Logic and Synthesis (IWLS), pp. 257-264, 2004.

10. Lin Yuan, Pushkin Pari and Gang Qu, Finding Redundant Constraints for FSM Minimization, National Conference on Artificial Intelligence (AAAI), pp. 976-977, 2004.

11. Pushkin Pari, Lin Yuan, Jane Lin and Gang Qu, Generating "Random" 3-SAT Instances with Specific Solution Space Structure, National Conference on Artificial Intelligence (AAAI), pp. 960-961, 2004.

12. Lin Yuan and Gang Qu, Information Hiding in Finite State Machine, 6th Information Hiding Workshop (IHW), pp. 340-354, Lecture Notes in Computer Science, Springer-Verlag, 2004.

13. Lin Yuan, Pushkin Pari and Gang Qu, Soft IP Protection: Verilog HDL Watermarking Technique, 6th Information Hiding Workshop (IHW), pp. 224-238, LNCS, Springer-Verlag, 2004.

14. Pushkin Pari, Lin Yuan and Gang Qu, How Many Solutions Does a SAT Instance Have?, IEEE International Symposium on Circuits And Systems (ISCAS), pp. 209-212, 2004.

15. Adarsh Jain, Lin Yuan, Pushkin Pari and Gang Qu, Zero Overhead Watermarking Technique for FPGA Designs, IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI), pp.147-152, 2003.

16. Lin Yuan and Gang Qu, Design Space Exploration for Energy-Efficient Secure Sensor Network, IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 88-92, 2002.

BIBLIOGRAPHY

[1] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage Current Reduction in CMOS VLSI Circuits by Input Vector Control", *IEEE Trans. VLSI*, vol. 12, pp. 140-154, Feb. 2004.

[2] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power", in *IC-CAD* 1994, pp. 74-81.

[3] F. Aloul, S. Hassoun, K. Sakallah, D. Blaauw, "Robust SAT-Based Search Algorithm for Leakage Power Reduction", *International Workshop on Integrated Circuit Design*, pp. 167-177, 2002.

[4] Mohab Anis, Mohamed Elmasry "Multi-Threshold CMOS Digital Circuits : Managing Leakage Power", *Springer*, October 2003.

[5] F. Assaderaghi, D. Sinitsky, S.A. Parke, J. Bokor, P.K. Ko, and C. Hu, "Dynamic Threshold-Voltage MOSFET(DTMOS) for ultra-low voltage VLSI", *IEEE Transaction on Electron Devices*, vol. 44, pp. 414-422, 1997.

[6] M.J. Avedillo, J.M. Quintana, and J.L. Huertas, "SMAS: a program for concurrent state reduction and state assignment of finite state machines," *IEEE International Symposium on Circuits and Systems*, pp. 1781-1784, 1991.

[7] H. Aydin, R. Melhem, D. Mosse, and P.M. Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems", in *Proc. of RTSS*, 2001, pp. 95-105.

[8] L. Benini and G. D. Micheli, "State Assignment for Low Power Dissipation," *IEEE Journal of Solid-State Circuits,* Vol.30, pp.258-268, March 1995.

[9] S. Bobba and I.N. Hajj, "Maximum Leakage Power Estimation for CMOS Circuits", *IEEE Alessandro Volta Memorial Workshop on Low Power Design*, pp. 116, 1999.

[10] M. Borah, R.M. Owens, and M.J. Irwin, "Transistor sizing for low power CMOS circuits", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 15, pp. 665–671, June, 1996.

[11] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in Sensor Networks: An Energy-Accuracy Trade-off," *Elsevier Ad-hoc Networks Journal (special issue on sensor network protocols and applications)*, 2003.

[12] B.H. Calhoun, F.A. Honore, and A. Chandrakasan, "Design Methodology for Fine-Grained Leakage Control in MTCMOS", *International Symposium on Low Power Electronics and Design*, pp. 104-109, 2003.

[13] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection," *Proc. of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April, 2005.

[14] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits,* Vol.27, pp. 473-484, April 1992.

[15] J. -F. Chamberland and V. V. Veeravalli, "Decentralized Detection in Sensor Networks," *IEEE Trans. on Signal Processing*, 51(2):407-416, February 2003.

[16] Z. Chen, M. Johnson, L. Wei, and K. Roy, "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks", in *Proc. ISLPED*,1998, 1998, pp. 239-244.

[17] C. Chen, A. Srivastava, and M. Sarrafzadeh, "On Gate Level Power Optimization using dual-Supply Voltages", *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 9, pp. 616-629, Oct. 2001.

[18] K. Chopra and S.B.K. Vrudhula, "Implicit Pseudo Boolean Enumeration Algorithms for Input Vector Control", in *Proc. DAC*, 2004, pp. 767-772.

[19] A. Dasgupta and R. Karri, "Simultaneous Scheduling and Biding for Power Minimization During Microarchitecture Synthesis", in *Proc. of ISLPED*, 1995, pp. 69-74.

[20] S. Devadas, H-T. Ma, R. Newton, and A. Sangiovanni-Vincentelli, "MUSTANG: State Assignment of Finite State Machines Targeting Multi-level Logic Implementations," *IEEE Transactions on Computer-Aided Design,* pp. 1290-1300, December 1988.

[21] D. Duarte, Y. Tsai, N. Vijaykrishnan, and M. Irwin, "Evaluating Run-Time Techniques for Leakage Power Reduction", in *Proc. VLSI Design*, 2002, pp. 31-38.

[22] E. J. Duarte-Melo and M. Liu, "Analysis of Energy Consumption and Lifetime of Heterogeneous Wireless Sensor Networks," *Proc. of IEEE Globecom*, Taipei, Taiwan, November 2002.

[23] G.Lakshminarayana, A. Raghunathan, N.K. Jha, and S. Dey, "Power Management in High-Level Synthesis", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 7, No. 1, pp. 7-15, March 1999.

[24] F. Gao and J.P. Hayes, "ILP-Based Optimization of Sequential Circuits for Low Power", in *Proc. of ISLPED* 2003, pp. 140-145.]

[25] F. Gao and J.P. Hayes, "Exact and Heuristic Approaches to Input Vector Control for Leakage Power Reduction", in *Proc. ICCAD*, 2004, pp. 527-532.

[26] M.R. Garey and D.S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness", Freeman Company, 2001.

[27] L. Goodby, A. Orailoglu, and P.M. Chau, "Microarchitectural Synthesis of Performance Constrained Low Power VLSI Designs", in *Proc. of ICCD*, 1994, pp. 323-326.

[28] G. D. Hachtel, M. Hermida, A. Pardo, M. Poncino, and F. Somenzi, "Re-Encoding Sequential Circuits to Reduce Power Dissipation," *International Workshop on Low-Power Design,* Napa, April 1994.

[29] G. D. Hachtel, B. Macii, A. Pardo, and F. Somenzi, "Probabilistic Analysis of Large Finite State Machines," *Proceedings of the ACM Design Automation Conferences,* San Diego, CA, June 1994.

[30] G.D. Hachtel and F. Somenzi, "Logic Synthesis and Verification Algorithms", Kluwer Academic Publishers, 1996.

[31] G. Hallbauer, "Procedures of state reduction and assignment in one step in synthesis of asynchronous sequential circuits," *International IFAC Symposium on Discrete Systems,* pp. 272-282, 1974.

[32] J. Halter, and F. Najm, "A Gate-Level Leakage Power Reduction Method for Ultra Low Power CMOS Circuits", in *Proc. CICC*, 1997, pp 475-478.

[33] J. Hartmanis and R.E. Stearns, "Some dangers in state reduction of sequential machines", *Information and Control,* pp252-260, Sept, 1962.

[34] M.J. Heijligers, L.J. Cluitmans and J.A. Jess, "High-level synthesis scheduling and allocation using genetic algorithms", in *Proc. of ASPDAC* 1995, pp. 61-66.

[35] J. Hill, R. Szewcyk, a. Woo, D. Culler, S. Hollar, and K. Pister, "System Architecture Directions for Networked Sensors," *8th Intl' Conf. on Architectural Support for Programming Languages and Operating Systems*, pp. 93-104, 2000.

[36] I. Hong, G. Qu, M. Potknojak, and M.B. Srivastava, "Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processor", in *Proc. of RTSS*, 1998, pp. 178-187.

[37] S. Hua and G. Qu, "Voltage Set-up Problem for Embedded Systems with Multiple Voltages", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 13, No. 7, July 2005.

[38] S. Hua, G. Qu, and S.S. Bhattacharyya, "Energy-Efficient Multi-Processor Implementation of Embedded Software", *ACM Transactions in Embedded Computing Systems (Special Issue on Concurrent Hardware-Software Design Method for Multi-Processor System-On-Chip)*, 2005.

[39] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits", in *Proc. of ISCAS* 1985, pp. 695-698.

[40] S. Iman, and M. Pedram, "POSE: Power Optimization and Synthesis Environment", *Proceedings of the 33rd Design Automation Conferences,* pp. 21-26, Las Vegas, NV, June 1996.

[41] R. Jejurikar, C. Pereira and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems", in *Proc. of DAC*, 2004, pp. 275-280.

[42] M.C. Johnson, D. Somasekhar, and K. Roy, "Models and Algorithms for Bounds on Leakage in CMOS Circuits", *IEEE Tran. Computer-Aided Design of Integrated Circuits and Systems*, vol.18, pp. 714-725, 1999.

[43] J. Kao, S. Narendra, A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques", in *Proc. ICCAD*, 2002, pp.141-148.

[44] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. of ACM MOBICOM*, August 2000.

[45] N.S. Kim, D. Blaauw, and T. Mudge, "Leakage Power Optimization Techniques for Ultra Deep Sub-Micron Multi-Level Caches", in *Proc. of ICCAD*, 2003, pp. 627-632.

[46] W. Y. Kim, K. Mechitov, J. Y. Choi, and S. Ham, "On Target Tracking with Binary Proximity Sensors," *Proc. of The Fourth International Symposium on*

Information Processing in Sensor Networks (IPSN'05), Los Angeles, CA, April 2005.

[47] M. Koegst, G. Franke, and K. Feske, "State Assignments for FSM Low Power Design", *Proceedings of the Conference on European Design Automation,* pp. 28-33, 1996.

[48] M. Koegst, S. Rulke, G. Franke, and M. Avedillo, "Two-Criterial Constraint-Driven FSM State Encoding for Low Power", *Euromicro Symposium on Digital Systems Design,* Warsaw, Poland, September 2001.

[49] C.M. Krishna and Y.H. Lee, "Voltage Clock Scaling Adaptive Scheduling Techniques for Low Power in Hard Real-Time Systems", in *Proc. of RTAS*, 2000, pp.156-165.

[50] B. Krishnamachari, D. Estrin and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," *Proc. of ICDCSW'02*, Vienna, Austria, July 2002.

[51] B. Krishnamachari, Y. Mourtada, and S. Wicker. "The Energy-Robustness Tradeoff for Routing in Wireless Sensor Networks," *IEEE International Conference on Communications*, Anchorage, Alaska, May 2003.

[52] A. Kumar and M. Anis, "Dual-V_t Design of FPGAs for Subthreshold Leakage Tolerance", in *Proc. of ISQED*, pp. 27-29, 2006.

[53] T. Kuroda, et al, "A 0.9V 150MHz 10mW 4mm2 2-D Discrete Cosine Transform Core Processor with Variable Threshold-Voltage(VT) Schemen", *IEEE Journal of Solid-State Circuits*, pp. 1770-1779, Nov. 1996.

[54] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn, "Managing Power and Performance for System-on-Chip Designs using Voltage Islands", in *Proc. of ICCAD* 2002, pp. 195-202.

[55] E.B. Lee and M. Perkowski, "Concurrent minimization and state assignment of finite state machines", *International Conference on Systems Man and Cybernetics,* pp. 248-260, 1984.

[56] Y-H. Lee, K.P. Reddy, and C.M. Krishna, "Scheduling Techniques for Reducing Leakage Power in Hard Real-Time Systems' ', *Euromicro Conference on Real-Time Systems*, pp. 140-148, 2003.

[57] D. Lee, W. Kwong, D. Blaauw, and D. Sylvester, "Analysis and Minimization Techniques for Total Leakage Considering Gate Oxide Leakage", in *Proc. DAC*, 2003, pp. 175-180.

[58] D. Lee and D. Blaauw, "Static Leakage Reduction through Simultaneous V_t/T_ox and State Assignment", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1014-1029, Vol. 24, No. 7, Jul. 2005.

[59] I. Lemberski, M. Koegst, S. Cotofana, and B. Juurlink, "FSM Non-Minimal State Encoding for Low Power," *Proceedings of the 23rd International Conference on Microelectronics,* Yugoslavia, May 2002.

[60] B. Lin, and A. R. Newton, "Synthesis of multiple-level logic from symbolic high-level description languages," *Proceedings of the IFIP TC 10/WG 10.5 International Conference on Very Large Scale Integration,* pp. 187-196, Federal Republic of Germany, August 1989.

[61] M. Maleki and M. Pedram, "QoM and Lifetime-constrained Random Deployment of Sensor Networks for Minimum Energy Consumption," *Proc. of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.

[62] D. Maniezzo, K. Yao and G. Mazzini, "Energetic Trade-off between Computing and Communication Resource in Multimedia Surveillance Sensor Network," *IEEE MWCN2002*, Stockholm, Sweden, September 2002.

[63] G. D. Micheli, R. Brayton, and A. Sangiovanni-Vincentelli, "Optimal State Assignment for Finite State Machines," *IEEE Transactions on Computer-Aided Design,* pp. 269-285, July 1985 .

[64] G. D. Micheli, "Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-level Logic Macros," *IEEE Transactions on Computer-Aided Design,* pp. 597-616, October 1986.

[65] S. Mutoh, T. Douskei, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold Voltage CMOS", *IEEE Journal of Solid-State Circuits*, pp. 847-854, Aug. 1995.

[66] C. Neau and K. Roy, "Optimal Body Bias Selection for Leakage Improvement and Process Compensation over Different Technology Generations", *International Symposium on Low Power Electronics and Design*, pp. 116-121, 2003.

[67] R. Niu, P. Varshney, M. H. Moore, and D. Klamer, "Decision Fusion in a Wireless Sensor Network with a Large Number of Sensors," *Proc. of the Seventh International Conference on Information Fusion*, Stockholm, Sweden, June 2004.

[68] W. Noth, and R. Kolla, "Spanning Tree Based State Encoding for Low Power Dissipation," *Proceedings of the Design Automation and Test in Europe*, pp. 168, Munich, Germany, March 1999.

[69] E. Olson and S.M.Kang, "State assignment for low-power FSM synthesis using genetic local search," *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference,* pp.140-143, San Diego, CA, May 1994.

[70] C. Park, J. Liu, and P. H. Chou, "Eco: an Ultra-Compact Low-Power Wireless Sensor Node for Real-Time Motion Monitoring," *Proc. of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.

[71] H. Vincent Poor, "An introduction to Signal Detection and Estimation," Second Edition, Springer.

[72] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research", in *Proc. of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.

[73] G. Qu, "What is the limit of Energy Saving by Dynamic Voltage Scaling?", in *Proc. of ICCAD*, 2001, pp. 560-563.

[74] G. Quan and X. Hu, "Minimum Energy Fixed-Priority Scheduling for Variable Voltage Processors", in *Proc. of DATE*, 2002, pp. 782-787.

[75] Y. Rachlin, R. Negi, and P. Khosla, "Sensing Capacity for Discrete Sensor Network Applications," *Proc. of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.

[76] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-Aware Wireless Sensor Networks," *IEEE Signal Processing,* vol. 19, no. 2, pp. 40-50, March 2002.

[77] R.M. Rao, F. Liu, J.L. Burns, and R.B. Brown, "A Heuristic to Determine Low Leakage Sleep State Vectors for CMOS Combinational Circuits", in *Proc. ICCAD*,2003 pp.689-692.

[78] K. Roy and S. C. Prasad, "SYCLOP: Synthesis of CMOS logic for low power application," *Proceedings of the International Conference on Computer Design,* pp. 464-467, October 1992.

[79] Robert R. Schaller, "Moore's law: past, present, and future", textitIEEE Spectrum, Vol. 34, pp. 52-59, June 1997.

[80] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," *IEEE Trans. on Mobile Computing*, vol. 1, no. 1, January-March 2002.

[81] B. Schott and M. Bajura, "Power-Aware Microsensor Design," *Intl' Conference on Computer-Aided Design*, San Jose, CA, November 2005.

[82] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw and V. Zolotov, "Discrete Vt Assignment and Gate Sizing Using a Self-Snapping Continuous Formulation", in *Proc. of ICCAD*, 2005, pp. 705-711.

[83] D. Shin, J.Kim and S. Lee, "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications", *IEEE Design and Test of Computers*, pp. 20-30, Mar. 2001.

[84] W. Shi, T. W. Sun, and R. D. Wesel, "Quasiconvexity and Optimal Binary Fusion for Distributed Detection with Identical Sensors in Generalized Gaussian Noise," *IEEE Trans. Inform. Theory*, vol. 47, pp. 446-450, January 2001.

[85] A. Sinha and A. P. Chandrakasan, "Dynamic Power Management in Wireless Sensor Network," *IEEE Design & Test of Computers*, Vol. 18, No. 2, pp. 62-74, April 2001.

[86] E. Sentovich, et al., "SIS: A System for Sequential Circuit Synthesis,"University of California, Berkeley, Electronics Research Laboratory Memorandum, No. UCB/ERL M92/41, May 1992.

[87] A. Srivastava, D. Sylvester, and D. Blaauw, "Power Minimization using Simultaneous Gate Sizing, Dual-Vdd and Dual-Vth Assignment", in *Proc. of DAC*, 2004, pp. 783-787.

[88] P. Surti, L. F. Chao and A. Tyagi, "Low Power FSM Design Using Huffman-Style Encoding," *Proceedings of IEEE European Design and Test Conference,* pp. 521-525, Paris, France 1997.

[89] C. Tsui, M. Pedram, A.M. Despain, "Technology decomposition and mapping targeting low power dissipation", in *DAC*, 1993, pp. 68-73.

[90] C. Tsui, M. Pedram, C. Chen, and A. M. Despain, "Low Power State Assignment Targeting Two- and Multi-level Logic Implementations," in *Proc. of ICCAD*, 1994, pp. 82-87.

[91] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design", in *Proc. of ISLPED*, 1995, pp. 3-8.

[92] V. Veeramachaneni, A. Tyagi, and S. Rajgopal, "Re-encoding for Low Power State Assignment of FSMs," *International Symposium on Low Power Design,* pp. 173-178, Dana Point, CA, April 1995.

[93] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 905-924, September 1990.

[94] Q. Wang and S.B.K. Vrudhula, "Algorithms for Minimizing Standby Power in Deep Submicronmeter Dual-V_t CMOS Circuits", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 306-318, Vol. 21, No. 3, Mar. 2002.

[95] S. Washabaugh, P. Franzon, and H. Nagle, "SABSA: Switching Activity Based State Assignment," *Proceedings of IEEE Solid State Circuits and Technology Committee Workshop on Low Power Electronics,* 1993.

[96] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits", in *Proc. of DAC*, pp. 489-494, 1998.

[97] L. Yan, J. Luo and N.K. Jha, "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-time Embedded Systems", in *Proc. of ICCAD*, 2003, pp. 30-38.

[98] L. Yuan and Gang Qu, "Design Space Exploration for Energy-Efficient Secure Sensor Network", *IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pp. 88-92, 2002.

[99] L. Yu and A. Ephremides, "Detection Performance and Energy Efficiency Trade-off in a Sensor Network," *Proc. of 2003 Allerton Conference*, Allerton, IL, October 2003.

[100] L. Yu and A. Ephremides, "Detection Performance and Energy Efficiency of Sequential Detection in a Sensor Network," *Proc. of HICSS'06*, Hawaii, January 2006.

[101] L. Yuan and G. Qu, "Energy Efficient Design for Distributed Sensor Networks," *Handbook of Sensor Network*, Chapter 38, CRC Press, October 2004.

[102] L. Yuan and G. Qu, "FSM Re-Engineering for Low Power State Encoding", *International Workshop on Logic and Synthesis (IWLS)*, pp. 257-264, June 2004.

[103] L. Yuan and G. Qu, "A Combined Gate Replacement and Input Vector Control Approaches for Leakage Current Reduction", Institute for Advanced Computer Studies (UMIACS), University of Maryland, MD, Tech. Rep. TR 2005-63, Nov. 2005.

[104] L. Yuan and G. Qu, "Analysis of Energy Reduction on Dynamic Voltage Scaling-Enabled Systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 12, pp. 1827-1837, Dec. 2005.

[105] L. Yuan, G. Qu, T. Villa, and A. Sangiovanni-Vincentelli, "FSM Re-Engineering and Its Application in Low Power State Encoding", in *Proc. of ASP-DAC*, 2005, pp. 254-259.

[106] L. Yuan and G. Qu, "A Combined Gate Replacement and Input Vector Control Approach for Leakage Current Reduction", *IEEE Transactions on Very Large Scale Integration Systems*, pp. 173-182, Vol. 14, No. 2, Feb. 2006.

[107] Q. Zhang, P. K. Varshney, and R. D. Wesel, "Optimal Bi-level Quantization of I.I.D. Sensor Observations for Binary Hypothesis Testing," *IEEE Trans. Inform. Theory*, July 2002.

[108] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling", in *Proc. of DAC*, 2004, pp. 868-873.

[109] L. Zhong, J. Luo, Y. Fei, and N.K. Jha, "Register Binding Based Power Management for High-level Synthesis of Control-Flow Intensive Behaviors", in *Proc. of ICCD*, 2002, pp. 391-394.

[110] Y. Zhu, R. S. Blum, Z. Q. Luo, and K. M. Wong, "Unexpected Properties and Optimum-Distributed Sensor Detectors for Dependent Observation Cases," *IEEE Trans. on Automatic Control*, vol. 45, no. 1, January 2000.

[111] Berkely Smart Dust Wireless Embedded Sensors, Available: http://robotics.eecs.berkeley.edu/ pister/SmartDust

[112] U.C.Berkeley BSIM3v3.1 SPICE MOS Device Models, 1997. Available: http://www-device.EECS.Berkeley.edu/bsim3/.

[113] Freescale Semiconductor Inc., Available: http://www.embedded.com/showArticle.jhtml?artic

[114] Transmeta Corporation, Available: http://www.transmeta.com/crusoe/specs.html.

[115] Saeyang Yang, "Synthesis and Optimization Benchmarks User Guide", 2002, Available: ftp://mcnc.mcnc.org.