ABSTRACT

Title of dissertation: ENHANCING POWER EFFICIENT DESIGN TECHNIQUES IN DEEP SUBMICRON ERA

Junjun Gu, Doctor of Philosophy, 2011

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

Excessive power dissipation has been one of the major bottlenecks for design and manufacture in the past couple of decades. Power efficient design has become more and more challenging when technology scales down to the deep submicron era that features the dominance of leakage, the manufacture variation, the on-chip temperature variation and higher reliability requirements, among others. Most of the computer aided design (CAD) tools and algorithms currently used in industry were developed in the pre deep submicron era and did not consider the new features explicitly and adequately.

Recent research advances in deep submicron design, such as the mechanisms of leakage, the source and characterization of manufacture variation, the cause and models of on-chip temperature variation, provide us the opportunity to incorporate these important issues in power efficient design. We explore this opportunity in this dissertation by demonstrating that significant power reduction can be achieved with only minor modification to the existing CAD tools and algorithms.

First, we consider peak current, which has become critical for circuit's reliability in deep submicron design. Traditional low power design techniques focus on the reduction of average power. We propose to reduce peak current while keeping the overhead on average power as small as possible. Second, dual V_t technique and gate sizing have been used simultaneously for leakage savings. However, this approach becomes less effective in deep submicron design. We propose to use the newly developed process-induced mechanical stress to enhance its performance. Finally, in deep submicron design, the impact of on-chip temperature variation on leakage and performance becomes more and more significant. We propose a temperature-aware dual V_t approach to alleviate hot spots and achieve further leakage reduction. We also consider this leakage-temperature dependency in the dynamic voltage scaling approach and discover that a commonly accepted result is incorrect for the current technology.

We conduct extensive experiments with popular design benchmarks, using the latest industry CAD tools and design libraries. The results show that our proposed enhancements are promising in power saving and are practical to solve the low power design challenges in deep submicron era.

ENHANCING POWER EFFICIENT DESIGN TECHNIQUES IN DEEP SUBMICRON ERA

by

Junjun Gu

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
2011

Advisory Committee:
Professor Gang Qu, Chair/Advisor
Professor Manoj Franklin
Professor Kazuo Nakajima
Professor Martin Peckerar
Professor Lourdes G. Salamanca-Riba

Dedication

To my parents and friends

Acknowledgments

First and foremost I'd like to thank my advisor, Professor Gang Qu for guiding me into a qualified researcher and influencing me into a nicer person. As an advisor, he is both strict to push for a more solid work, and liberal to allow student's self-innovation. As a collaborator, he teaches me how to better communicate with partners and always recognize other people's efforts. He also makes me realize that a patient listener and a timely offer of help is what constitutes a good friend, as what he is to me. I will always cherish my four-year experience with him, which I believe will impact me life long.

I would like to thank Professor Kazuo Nakajima, with whom I took three courses with, and Professor Martin Peckerar for serving on my proposal committee and giving me valuable comments for my work. I would also like to thank Professor Manoj Franklin and Professor Lourdes G. Salamanca-Riba for agreeing to serve on my thesis committee and spending their time reviewing my manuscript.

I would like to thank Dr. Lin Yuan, a former student of Professor Gang Qu. I've worked with him on several projects and those are very pleasant collaboration. He introduces practical problems from industrial perspective and gets me more familiar with standard industry flow and simulation tools. I enjoy lots of inspiring discussions with him. I would also like to thank Dr. Yi-Min Jiang, my supervisor when I was doing my summer intern. Through the interactive meetings with him during the three months, I was able to apply my parallel computing knowledge learned from graduate seminar and speed up the core engine of their tool. He also encouraged me to take the initiative to resort others for help instead of struggling with everything by myself.

My PhD years would not have been so memorable without my friends. Jing Wu, previously my labmate for over one year, is someone I can always trust and talk to. Besides, she is such a lovely person to enjoy off-work time with. We went to the gym together, ate together and hung out together. I will always remember the happy moments with her. Sangkyo Han, my former labmate, is also a fun person. We enjoyed Chinese buffet or Korean BBQ together from time to time. I also respect his optimism about difficulties and life.

I would also like to acknowledge help and supports from Ms. Tracy Chung, Ms. Melanie Prange, Ms. Maria Hoo and Ms. Vivian Lu and the staff members at ECE Help Desk. 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

List of Figures

List of Abbreviations

IC	Integrated Circuit
EDA	Electronic Design Automation
DFM	Design For Manufacturability
DVS	Dynamic Voltage Scaling
SCE	Short Channel Effect
DIBL	Drain-Induced Barrier Lowering
CMOS	Complementary metal-oxide-semiconductor
MTCMOS	Multithreshold-voltage CMOS
VTCMOS	Variable Threshold CMOS
DTMOS	Dynamic Threshold CMOS
DGDT-MOS	Double-gate Dynamic Threshold SOI CMOS
DVS	Dynamic Voltage Scaling
SRAM	Static Random Access Memory
DPM	Dynamic Power Management
FSM	Finite State Machine
TSV	through-silicon-via
TTSV	Thermal-through-silicon-via
MFC	micro-fluidic channel

Chapter 1

Introduction

Over the past several decades, the continuous downsizing of transistors and related frabrication processes have been the key driving force for the blooming of semiconductor and integrated circuit (IC) industry. The number of transistors integrated on a single die roughly doubles in every 18 months, enabling the implementation of much faster and more sophisticated systems that permeate daily life, from portable computing devices and wireless communication systems to high-end products used in scientific computing and large data centers.

However, the delivery of advanced silicon solutions has to rely on the availability of electronic design automation (EDA), design technologies, as well as effective verification methodologies at all levels of abstraction [1]. With semiconductor industry entering deep submicron era, many challenges are imposed on EDA and IC design community. This thesis focuses on the challenges related to power and energy efficiency. Specifically, we investigate how to enhance several state-of-the-art power effcient design techniques so that further power/energy savings can be achieved and other critical issues can be alleviated at the same time. The keyword in this statement is to "enhance". That is, we are not developing new techniques. Instead, we will incorporate new requirements and information in the deep submicron design to improve the performance of existing power optimization techniques without introducing major changes to these EDA tools.

1.1 Challenges in Deep Submicron Era

The design challenges in deep submicron domain can be generally classified as follows [2]:

- **Signal Integrity:** The much higher operating frequency along with the scaling and integration of mixed-signal and RF components bring ever larger noise and interference to the circuit signals. Emerging issues include noise headroom, large numbers of capacitively and inductively coupled interconnects, supply voltage IR drop and ground bounce and substrate coupling. These issues may cause false switching, delay variation or timing failures [6, 7].

- **Power Management:** High power densities worsen the thermal impact on circuit performance and reliability, while decreasing supply voltages worsen noise margin and leakage current. Under 45nm technology and beyond, leakage power starts to dominate dynamic power. As a result, the focus of research and EDA tool development has shifted to the reduction of leakage and total power consumption, particularly subthreshold leakage.

- **Reliability and Resilience:** Due to the much lower supply voltage, the gate oxide thickness has to be reduced to maintain circuit performance. However, with the continuous scaling, gate oxide breakdown is expected to arise. In addition, the much higher direct current densities in deep submicrometer design can lead to Joule heating and electromigration effect. Other issues such as single-event upset can also become a serious reliability concern. Evidently, automatic insertion of robustness into design is becoming a new trend.

- **Manufacturing Variability:** Advanced ICs with high functionality, low power consumption and extreme reliability are pushing the manufacturing process to the limit. Slight manufacturing variations can significantly affect performance and yield. Interconnecting manufacturing-related disciplines with design process can provide a possible solution. A popular research topic known as design for manufacturability (DFM) has emerged.

- **Design Productivity:** To avoid the exponential increase of design complexity and thus design cost, design productivity has to be improved with technology scaling. The implied needs for design productivity reside in verification, embedded software design, and automated methods for analog/mixed-signal design and test. For instance, design verification is one of the major bottlenecks in the design process. One current practice, the partial verification process provides only a small fraction of coverage. A potential breakthrough relies on the shift from such ad hoc verification method to more structured and formal processes.

Each of the design challenges demands substantial research efforts. This thesis focuses on the challenge of power management. In the remaining of this chapter, we first review the state-of-the-art power efficient design techniques. Next we explore the new challenges and opportunities facing low power design in deep submicron domain. Finally we conclude with the overview of the thesis and the key contributions.

1.2 Existing Work on Power Efficient Design

In this section, we briefly describe the most relevant work on power efficient design. Detailed survey of the current state-of-the-art power efficient design methodologies can be found in Chapter 2.

Extensive researches on power efficient design have been performed at all design levels. To reduce dynamic power, various dynamic power management techniques [33] such as clock gating and dynamic voltage scaling can be applied at system level. Clock gating uses a hierarchical clocking scheme with conditional clocks that turn off sections of the chip that are not needed on a cycle-by-cycle basis [34]. Dynamic voltage scaling scales down the supply voltage when the workload is not intensive [32]. At logic synthesis level, precomputation method is proposed which adds combinational logic in front of the original circuit to precompute the output logic values of a subset of input cases so that part of the circuit could be turned off in the succeeding clock cycle [36]. Path equalization makes all signal paths from input to output of the same length to minimize spurious switching activities [37]. Local transformations like re-factoring, re-mapping, phase assignment and pin swapping seek to have gates with high switching activity fan out to nets with small capacitance [37]. Finite state machine encoding also matters because the hamming distance between two codes determines the number of switchings [39]. At circuit level, static voltage scaling can be applied, where critical units and non-critical units are powered by higher and lower supply voltages, respectively [29, 30]. Transistor sizing is also effective in that it reduces load capacitance [38].

To minimize leakage power, varieties of techniques are also proposed at different design levels. At system level, dynamic V_t scaling technique will automatically adjust threshold voltage depending on the current workload [24, 25]. Less leakage will be consumed by increasing V_t when workload is small. At logic synthesis level, approaches like input vector control takes advantage of transistor stacking effect and selects an optimal input vector to minimize subthreshold leakage in standby mode [16, 17]. At circuit level, multithreshold-voltage CMOS (MTCMOS) reduces standby leakage power by inserting high V_t devices in between power/ground line and low V_t circuitry. The low V_t circuitry will be cut off from power/ground during standby mode by the high V_t devices to reduce leakage [18, 19]. Dual V_t technique assigns high V_t to transistors on non-critical paths, while maintaining the performance by preserving low V_t to transistors on critical paths [112]. Variable threshold CMOS (VTCMOS) employs self-substrate bias circuit and applies a reverse body bias in standby mode to increase threshold voltage and cut off leakage current [41, 20]. Similarly, dynamic threshold CMOS is also able to alter threshold voltage by connecting the gate and body together [21, 22].

1.3 New Challenges and Opportunities for Power Efficient Design

All the challenges in deep submicron design we describe in Section 1.1 have impact on the power and energy efficient design methodologies developed in the past couple of decades. Among others, the dominance of leakage, the manufacture variation, the on-chip temperature variation, and higher reliability requirement highlight the new challenges. These are the features that most of the low power design methods developed in the pre deep submicron era did not consider explicitly and adequately.

The recent research advances in deep submicron design, such as the mechanisms of leakage, the source and characterization of manufacture variation, the cause and models of on-chip temperature variation, provide us the opportunity to incorporate these important issues in power efficient design. Since the late 1990's, not only have we seen many new low power techniques targeting deep submicron design, but also the growing trend of combining two or more techniques simultaneously for further power efficiency.

Meanwhile, there is another design trend known as *physical synthesis* that integrates physical design with high level synthesis. In deep submicron domain, design productivity requires system level specifications to sign off into reliable and predictable handoffs at physical design level. However, the growing silicon complexity makes it difficult to estimate the effects of an optimization factor on eventual design quality (speed, power, signal integrity, reliability and manufacturing variability). To avoid excessive guardbanding, the design in logic synthesis level and even system level must become more closely related to physical design. In some cases, it is accomplished by characterizing the target physical design specifications at logic or system level. In some other cases, it involves repeated iterations of synthesis stage and placement and/or routing stage to achieve design spec closure.

Finally, deep submicron design does bring many challenges and there is the need to re-visit many of the existing CAD tools and algorithms. However, from practical point of view, it is not desirable for EDA and semiconductor industry to re-design and re-development these tools. The goal of this dissertation is to demonstrate that it is feasible to modify the current CAD tools and algorithms to incorporate the new features and requirements of deep submicron design. Specifically, we will elaborate our idea and approach through the following four examples.

Peak current is one of the vital physical parameters for circuit reliability. Short-lived high current pulses can bring large voltage drops to the power line or ground bounce, both of which can impact circuit timing [57, 54, 69] or lead to logic errors[54, 69]. As the feature size continues shrinking, the metal conductors are much finer than before. This imposes an even greater challenge for designers to control the peak current in order to avoid electromigration effect [56, 64, 58]. It is known that peak current in synchronous circuits usually occurs at clock transition, when the clock tree, flip-flops, and the combinational circuits directly driven by the flip-flops switch simultaneously. A recent report based on SPICE simulation shows that peak current in a synchronous circuit is dominated by the current in state registers, which can be characterized at logic synthesis level by the maximum number of state bits switching in the same direction in the finite state machine (FSM) model. **Therefore, it becomes interesting to study how to incorporate peak current reduction into the traditional average power minimization technique at logic synthesis stage.**

It has been a common practice in today's industry design to combine the multiple power optimization techniques while achieving timing closure. For example, dual V_t technique and gate sizing have been used simultaneously to leverage timing slack for leakage and dynamic power savings [79, 112, 78, 38]. However, in deep submicron design, the room for changing threshold voltage and sizing gates becomes much smaller. In addition, the timing budget becomes tighter and tighter. These significantly limit the effectiveness of low power design techniques that rely on the power versus timing tradeoff. Process-induced mechanical stress is a novel technique induced into CMOS channel that enhances carrier mobility and reduces transistor delay [87, 88, 89]. **With the pre-characterized stress-enhanced cell into the library [90], we will be able to integrate mechanical stress optimization into the power optimization process.**

Excessive power dissipation in a certain chip region can result in local heat accumulation and therefore hot spots. For a high-performance microprocessor chip, the hot spot temperature can be 50^oC higher than other regions, which can potentially become a reliability issue. Meanwhile, since leakage current has an exponential dependency on temperature, the leakage power consumption can be much larger in high temperature region, which will in turn exacerbate heat accumulation. Traditional dual V_t technique assumes a uniform on-chip thermal distribution, which can either lead to too pessimistic or too optimistic design. Therefore it would be more accurate and effective if thermal profile can be taken into consideration during the dual V_t design procedure. Because of the interdependency between leakage and temperature, it is necessary to iteratively update thermal and power profile each time dual V_t assignment is performed. However, thermal profile can not be obtained until the placement is performed and physical location of each cell is known. **As a result, further investigation needs to be conducted for thermal-aware dual V_t techniques.**

On-chip temperature does not only affect leakage, it also has impact on power optimization techniques at system level. Dynamic voltage scaling (DVS) is one of the most popular power management techniques at system level. Generally the supply voltage is scaled down so that scheduled tasks can be finished exactly at its deadline to save dynamic power and energy [124, 132]. When leakage starts to dominate dynamic power, it is believed that more total energy can be saved if tasks can be finished earlier and the system can be shut down to avoid leakage consumption [125, 128, 137, 116, 119, 120, 138]. However, the DVS policies proposed in existing works either ignore leakage power or treat leakage as a temperature-independent constant. **It is therefore of first importance to propose an accurate transient temperature model that fully considers temperature-leakage interdependency in order to enhance the DVS technique.**

1.4 Key Contributions

As illustrated in the previous section, multiple techniques should be combined to achieve further power efficiency and higher level designs must be more closely linked to physical design to improve design productivity. This thesis proposes several enhancement to traditional power optimization techniques, regarding issues of peak current, on-chip temperature and layout-dependent mechanical stress in deep submicron domain.

- **Enhancing FSM Power Efficient Design by Simultaneous State Replication and State Re-encoding:** Traditional FSM power efficient design targets minimization of average power consumption, particularly, dynamic power consumption. It is well known that dynamic power relates to the total switching activities (TSA) [53]. However, peak current can remain high even when average power is minimized, affecting circuit's reliability. Huang et al. [70] show a strong correlation between the peak current in a sequential circuit and the maximum number of state registers switching in the same direction upon state transitions, referred to as peak switching value (PSV). In the work presented in Chapter 3, we propose an enhanced FSM synthesis framework that takes both average power minimization and peak power reduction into consideration. The enhanced framework applies power optimization techniques *state replication* [74] and *state re-encoding* simultaneously. Starting with an already synthesized FSM with, for example, minimal average power as the synthesis target, we identify all the transitions that reach maximum PSV and put them into a working set S. Then we construct the corresponding solution pool SP that includes all the feasible codes for reducing the PSV by state replication or state re-encoding. We pick the least constraining solution from SP for the most constrained transition in S and perform an update over S and SP. We continue this procedure until either S or SP becomes empty. In the former case, it means that PSV has been reduced. In the latter, it means that we fail to reduce PSV. Our experiments show that out of the 52 MCNC FSM benchmarks encoded by power-driven encoding algorithm *POW3* [55], 39 of them are not optimal in terms of PSV. Our approach can improve 34 of them with an average 39.2% reduction, while [70] can improve 27 benchmarks with an average 24.5% reduction. Meanwhile, our approach only incur 3% overhead in TSA compared to *POW3* and [70].

- **Improving Dual V_t Technology by Simultaneous Gate Sizing and Mechanical Stress Optimization:** Process-induced mechanical stress is used to enhance carrier mobility and drive current in contemporary CMOS technologies. Stressed cells have reduced delay but larger leakage consumption. Its efficient power/delay trading ratio makes mechanical stress an enticing alternative to other power optimization techniques. The work in Chapter 4 re-evaluates the concept of leakage minimization with multiple techniques simultaneously (dual V_t, gate sizing, and mechanical stress in our discussion). More specifically, we first balance the circuit paths as close to the timing constraint as possible (apply dual V_t in our case) and identify paths that do not have sufficient slack for further leakage reduction with high V_t. Then we repeat the following 2-step procedure: using gate sizing and/or mechanical stress to create new slacks at the cost of power overhead; using dual V_t to trade these new slacks for power saving. We introduce the concept of *urgent paths* to help us locate the cells that would relax as many paths as possible in the first step. The use of mechanical stress is shown to achieve 9.77%

leakage and 2.79% total power savings over combined gate sizing and dual V_t approach. Moreover, the employment of *urgentpath* also plays a crucial role in power optimization, achieving 13.5% leakage and 5.00% total power reduction compared to the sensitivity-based approach in the existing literature.

- **Enhancing Dual V_t Leakage Minimization by Considering On-Chip Temperature Variation:** In Chapter 5, we aim to use on-chip temperature variation and coupling effects between leakage and temperature to guide further leakage reduction. The temperature profile can be obtained by 3-D mesh model [96] given the placement information and V_t assignment. The proposed temperature-aware dual V_t algorithm is a three-phase procedure. First, we obtain the temperature profile based on an initial V_t assignment. Then we repetitively adjust this initial assignment by changing as many low V_t cells as possible in hot region to high V_t (for leakage reduction), and changing as few high V_t cells as possible in cool regions to low V_t (for timing closure). Temperature profile is updated at the end of each iteration. This second phase will stop when there is little leakage savings or temperature variation becomes trivial. After the second phase, slacks tend to be distributed to cells in hot region where the change of low V_t to high V_t gives more leakage saving. In the last phase, we take advantage of slacks created from timing closure step and search for further leakage reduction. The experiment is performed on five large circuits from OpenCores [115], ranging from a few thousands to over 50K cells. We use a TSMC 65nm low-power dual-V_t library with 17 base cells.

The circuits are synthesized and placed by *Synopsys Design Compiler* and *IC Compiler* respectively. *Synopsys Power Compiler* provides leakage power evaluation and the intial V_t assignment (using dual-V_t algorithm described in [111]). The results show that we are able to achieve an average of 11.2% in leakage saving and 39% reduction of cells in hot regions without timing failure.

- **Enhancing DVS Technique for Real-Time Systems by Incorporating Temperature-Leakage Interdependency:** In the work presented in Chapter 6, temperature-aware design approach is applied to system level. Dynamic voltage scaling (DVS) is one of the most effective techniques to reduce dynamic power for real-time systems. We first study the interdependency of temperature and leakage and how it influences DVS. We derive a temperature model in analytic form that considers the interdependency between leakage and temperature so that the transient behavior of temperature can be captured. We then re-visit DVS approaches and find that, for a single task that starts execution from time zero, scaling down the voltage to the lowest level without missing execution deadline gives the most total energy saving with the parameters in the current technology. With the presence of idle time, allocating all of it before executing the task helps cooling down the system, and thus achieves the most energy saving. However, for a set of non-preemptive real-time tasks, the same idle time distribution strategy may result in high starting temperature for the subsequent task. Based on the observation that an idle time of $1.5 \times R \times C$ will be sufficient for the system cooling procedure, we propose an efficient online DVS scheduling policy for total energy minimization. Experimental results show that the traditional DVS and CS-DVS approaches consume 6% and 9% more total energy, respectively.

Chapter 2

Preliminary

2.1 Power Fundamentals

There are three major sources of power dissipation in a CMOS circuit [12]:

$$P_{total} = P_{dynamic} + P_{leakage} + P_{SC} \tag{2.1}$$

$P_{dynamic}$ is the dynamic power, $P_{leakage}$ is the leakage power, and P_{SC} is the short circuit power.

The dynamic power is caused by charging and discharging the parasitic capacitance in the circuit. It can be modeled as follow:

$$P_{dynamic} = \alpha C V_{dd}^2 f_{CLK} \tag{2.2}$$

where α is the switching activity, which is the effective voltage transitions per clock cycle. C is the load capacitance, V_{dd} is the supply voltage, and f_{CLK} is the clock frequency.

High leakage current in deep submicron era is becoming a significant contributor to power dissipation of CMOS circuits as threshold voltage, channel length, and gate oxide thickness are reduced. As indicated by the left chart in Figure 2.1, leakage power consumption can take up to 50% of the total power for 45nm technology node [5].

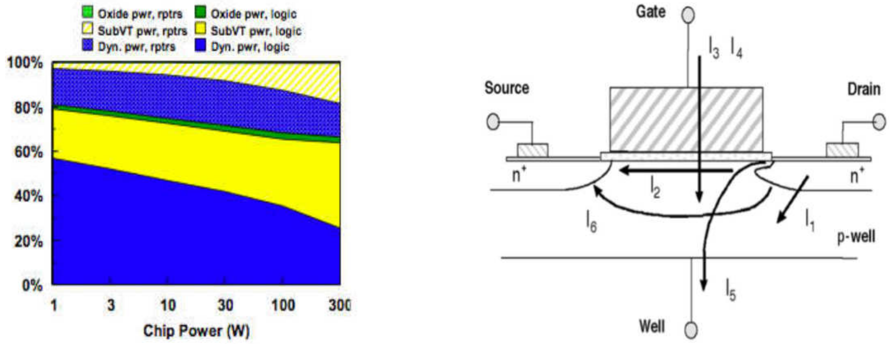

Figure 2.1: Left: Cumulative power allocation fractions for the logic in processor cores, for 45nm node [5]. Right: Leakage current mechanisms of deep-submicrometer transistors [105].

There are six leakage mechanisms [4], shown in the right subfigure in Figure 2.1: the reverse bias pn junction leakage (I_1), subthreshold leakage (I_2), oxide tunneling current (I_3), gate current due to hot-carrier injection (I_4), gate induced drain leakage (I_5), and channel punchthrough current (I_6). In long-channel devices, leakage is dominated by reverse bias pn junction leakage, while in deep submicron domain, subthreshold leakage takes the larger proportion in short-channel transistors. It is because short-channel transistors require lower power supply levels to reduce power consumption. The lower power supply levels force a reduction in threshold voltage to maintain a high drive current, which can cause substantial increase in leakage. This is commonly referred to as Short Channel Effects (SCE) [105]. Meanwhile, oxide thickness has to be reduced nearly in proportional to the channel length to maintain a reasonable SCE immunity. The decrease in oxide thickness results in increase in the electric field, which along with thinner oxide, can result in considerable leakage current flowing through the gate. As a result, the major leakage sources in deep submicron domain include subthreshold leakage and gate oxide leakage (contributed by three mechanisms of I_3, I_4 and I_5). In current technology, gate leakage current can be largely supressed by using high-k dielectric materials, while subtheshold leakage is still a serious concern. Moreover, subthreshold leakage has an exponential dependency on temperature according to BSIM model [140].

Besides dynamic power and leakage power, the short circuit power can be kept less than 15% of the dyanmic power with careful design [13] and therefore is no longer a big concern.

2.2 Power Efficient Design Techniques for CMOS Circuits

Extensive researches have been performed on power efficient design at different design levels. In the following, we provide a survey on various power efficient design techniques at various design levels [105, 15].

2.2.1 Dynamic power minimization

Dynamic power minimization can be achieved by decreasing any of the four factors that appear in Equation 2.2.

- **Dynamic Power Management:** Dynamic power management (DPM) is a design methodology that dynamically reconfigures a system to provide the requested services and performances with a minimum number of active components or a minimum load on such components [33]. Clock gating is a typical example of DPM. Since the power dissipation of the clocked components in a system is often the largest of the total power consumption, clock gating uses a hierarchical clocking scheme with conditional clocks that turn off sections of the chip that are not needed on a cycle-by-cycle basis [34]. Similarly, [35] propose to shut down the idle components of a system to save both dynamic and leakage power. However, both approaches suffer from latency penality and power overhead.

- **Dynamic Voltage Scaling:** Because of the quadractic dependence of the dynamic power on supply voltage, it is very effective to reduce the dynamic power if the supply voltage scales down. The operating system of a microprocessor can intelligently determine the processor speed. Then the power management unit generates the minimum voltage required for the desired speed. All devices of the microprocessor work at the same frequency under the same supply voltage. When it comes non-intensive task, the operating system scales down the supply voltage to save power [32]. Dynamic Voltage Scaling (DVS) can also fall into the category of DPM.

- **Static Voltage Scaling:** Static voltage scaling is also based on the quadratic dependence of the dynamic power on supply voltage [29, 30]. However, multiple supply voltages are provided. It is of the same flavor as dual V_t technique, where critical units and non-critical units are powered by higher and lower supply voltages, respectively. The differences are that the critical and non-critical units are clustered as much as possible and level converter is inserted at the interfaces of low V_{dd} to high V_{dd} units.

- **Precomputation method:** Precomputation method adds combinational logic in front of the original circuit. It can precompute the output logic values of a subset of input cases one clock cycle before they are required so that part of the circuit could be turned off in the succeeding clock cycle [36]. This method reduces both load capacitance and switching activity.

- **Path equalization:** Path equalization transforms the logic network so that all signal paths from input to output have the same length. This ensures that most gates can have aligned transitions at their inputs, thereby minimizing spurious switching activities [37]. This technique is very effective in arithmetic circuits as they have more regular structures. For those with irregular structures, gate resizing can be applied so that the delay of fast paths can be equalized to that of critical paths without affecting circuit timing.

- **Local transformations:** Techniques like re-factoring, re-mapping, phase assignment and pin swapping can all be classified as local transformations. They are applied on gate netlist and target the nets with large switching capacitance (αC). They seek to fan out nodes with high switching activity to the net that contributes small load capacitance [37].

- **Finite State Machine encoding:** State encoding is to assign a unique code to each state such that certain cost function of binary logic level can be minimized. As the 01 or 10 switching at binary logic level will result in the switchings in flip-flops, the hamming distance of two codes involved in each transition should be as small as possible to minimize dynamic power [39].

- **Transistor sizing:** Transistor sizing is exploited in both combinational cells and flip-flops to minimize dynamic power consumption [38]. By sizing down the transistors, load capacitances and thereby power consumption of the fan-in components can be largely reduced. Sizing down flip-flops can be especially effective in reducing power as flip-flops contribute to the load of clock distribution network, which constantly involves switching activities. However, sizing down transistors can increase transistor delay, so transistor sizing needs to be carefully designed to take advantage of power-delay tradeoff.

2.2.2 Leakage power minimization

As leakage power has become a major contributor to the total power dissipation in deep submicron technology, it is crucial to take leakage power minimization into consideration.

- **Transistor stacking effect:** The subthreshold leakage flowing through a stack of transistors is significantly smaller if more than one transistor in the stack is turned off. Because of this stacking effect, the magnitude of subthreshold leakage can be largely dependent on the applied input vector, which is propagated from primary input vector. Due to the exponential size of primary input vectors, an enumerative approach is impractical. [16] applied a random search-based approach, while [17] employed a genetic algorithm to speculate new search points with potentially improved performance. In [14], a low V_t transistor is inserted into the stack and is turned off in standby mode to reduce subthreshold leakage.

- **Multiple-threshold technologies:** The threshold voltage of a transistor can be adjusted by means of channel-doping densities, gate oxide thickness, channel length and body bias. Various techniques have been proposed that take advantage of different properties of different V_t transistors: low V_t transistors are used to achieve high performance, while high V_t transistors can suppress subthreshold leakage power. Multithreshold-voltage CMOS (MTCMOS) reduces standby leakage power by inserting high V_t devices in between power/ground line and low V_t circuitry. The low V_t circuitry will be cut off from power/ground during sleep mode by the high V_t devices to reduce leakage [18, 19]. MTCMOS can bring delay overhead in active mode and area overhead to the circuit. Dual V_t technique assigns high V_t to transistors in non-critical paths, while maintaining the performance by preserving low V_t to transistors on critical paths [112]. Dual V_t technique is able to suppress leakage in both active and standby modes. Variable threshold CMOS (VTCMOS) employs self-substrate bias circuit and applies a reverse body bias in standby mode to increase threshold voltage and cut off leakage current [41, 20]. However, the effectiveness of reverse body bias has been decreased as technology scales. Dynamic threshold CMOS (DTMOS) is able to alter threshold voltage dynamically by connecting the gate and body together [21]. DTMOS is only suitable for ultralow voltage circuits. Double-gate dynamic threshold SOI CMOS (DGDT-MOS) combines the advantage of DTMOS and double-gate

FD SOI MOSFETs without limitations on the supply voltage [22].

- **Super cutoff CMOS:** Super cutoff CMOS (SCCMOS) uses an on-chip boost voltage to provide reverse bias on the sleep transistor to cut off the leakage current in standby mode [23]. Different from MTCMOS, the sleep transistor is low V_t and therefore the circuit can work at lower supply voltages.

- **Dynamic V_t Scaling:** Dynamic V_t scaling will automatically adjust threshold voltage as well as operating frequency depending on the current workload [24, 25]. When workload is small, less leakage power is consumed by increasing V_t.

- **SRAM Cache memory leakage reduction:** Memory structures (instruction and data caches, prediction tables and translation look-aside buffers) in modern microprocessors have taken large fraction of chip areas. In deep submicron technology, caches account for a large component of leakage power consumption. Several techniques have been proposed to address this issue. Date retention gated-ground chache places an extra NMOS transistor in the leakage path from power to ground of SRAM cells. The unused portions of cache is changed into low leakage mode by turning of the extra NMOS transistor [26]. Drowsy cache [27] and Dynamic V_t SRAM [28] utilizes multiple supply voltages and dynamic threshold voltages by body biasing to reduce SRAM leakage.

2.3 Summary

In this chapter, we first address the three major sources of power consumption in CMOS circuits: dynamic power, leakage power and short circuit power. Dynamic power has already been a major component of power consumption. With continous scaling of CMOS devices, however, leakage power starts to dominate dynamic power and becomes a emerging challenge for power efficient design. On the other hand, short circuit power can be largely suppressed in today's technology. We then review the most popular and representative techniques that minimize dynamic power and leakage power respectively. The techniques apply to various design levels. Take the dynamic power minimization techniques for example. DPM and DVS apply to system level design, requiring software support. Precomputation, path equalization, FSM encoding and local transformations belong to logic synthesis level. Transistor sizing can be categorized into circuit level.

Chapter 3

Finite State Machine Synthesis Technique for Peak Current Reduction

Peak current reduction has attracted more and more attention in circuit design at deep sub-micron technology node. Peak current is vital for circuit reliability because short-lived high current pulses can cause a metal line to fail instantaneously and large voltage drops that impact circuit timing [57, 54, 69]. As the feature size continues shrinking, the metal conductors are much finer than before. This imposes a great challenge for designers to control the peak current in order to meet the electromigration specification when the peak current occurs [56, 64, 58]. In addition, the supply voltage in today's digital circuits has been consistently scaled down to reduce power consumption. Large voltage drops caused by peak current can have a serious impact on circuit timing and may lead to timing failures and logic errors [54, 69].

In synchronous circuits, peak current usually occurs at clock transition, when the clock tree, flip-flops, and the combinational circuits directly driven by the flip-flops switch simultaneously. Many approaches at different design levels have been proposed to mitigate the large current flows at clock transitions. At system level, various clock skew scheduling algorithms are proposed to achieve more even distribution of clock arrival times [58, 59, 60, 61, 62, 63]. At physical level, many works seek to change polarities of clock buffers so that current flows can be altered into opposite directions [64, 65, 66, 67]. Other techniques like MTCMOS [68] and flip-flop resynthesis [69] can also be applied at physical design stage to reduce peak current. Very few works happen at logic synthesis level. The only two works we are aware of are state register re-encoding approach [70] and SAT-based encoding [71].

Peak current reduction at FSM synthesis level is based on the two observations: (i) peak current in an FSM circuit is largely dependent on peak current in state registers. This is because the state registers often have large switching current due to their large load capacitance at the fanouts. Besides, all the state registers switch simultaneously at the arrival of the clock signal (assuming a very small clock skew). (ii) peak current in state registers are directly related to the maximum number of state bits switching in the same direction. This is because when state bits switch from one logic value to the other. The total currents in power or ground nets are the sum of the charging or discharging current in individual registers.

We focus on minimizing peak current in state registers during state transition in FSM. For a given encoded FSM, we first identify the state transitions that cause peak current, then we consider state replication and state re-encoding simultaneously to reduce the maximal number of charging and discharging on such state transitions. In state re-encoding, we aim to find a new code for a state that currently contributes to peak current. In state replication, we create a copy of a state and assign it a different code to reduce the charging and discharging on that state transition. Our proposed approach seamlessly combines these two basic techniques by the most constrained least constraining paradigm [72] and outperforms the state-of-the-art FSM level approach [70] in reducing peak current on most of the MCNC benchmark circuits.

We have made the following contributions in peak current reduction for FSM in this article:

- This is the first work of *re-encoding state* for peak current reduction. Although Huang et al. use the same term in [70], our method is completely different from theirs. In fact, the solution space they consider is only a very small subset of ours.

- We have applied the *state replication* technique proposed by Yuan et al. [74] for peak current reduction. By the nature of state replication, it gives us more freedom and options to reduce the maximal charging and discharging and therefore peak current.

- We propose the simultaneous state replication and re-encoding (*SSRR*) algorithm that combines these two techniques by the most constrained least constraining paradigm. The algorithm takes an encoded FSM as input and produces a new encoded FSM with less peak current. Its complexity is polynomial to the number of states and the number of transitions in the FSM.

- We have implemented the power-driven state encoding algorithm POW3 [55], the only existing peak current minimization approach at FSM level [70], and our *SSRR* and apply them to the entire 52 MCNC benchmark circuits. At FSM level, *SSRR* reduces *PSV* on 34 of the 36 circuits that POW3 is unable to minimize with solutions 18.4% better than those found by [70].

The rest of the article is organized as follows. In Section 3.1, a brief survey on recent works for peak current reduction is conducted. In Section 3.2, we define the necessary terms and review the two most relevant works. The two basic techniques, state replication and state re-encoding, are illustrated using an example in Section 3.3 to show how they can be improved or tuned for our purpose. Then we discuss in details how we can combine these two techniques seamlessly to reduce peak current and total dynamic power in Section 3.4. An illustrative example is given in Section 3.5 to help understand the proposed approach. Section 3.6 reports the experimental setup and results. Section 3.7 concludes.

3.1 Related Works

Different approaches have been proposed to reduce peak current. For a synchronous circuit, peak current can be mostly observed at the moment of clock transitions, because clock buffers at the same level in the clock tree, flip-flops and combinational cells driven by the flip-flops can switch at the same time window. As a result, lots of charging and discharging can happen to the load capacitances of the transistors involved with these units. Varieties of clock skew scheduling approaches have been proposed [58, 59, 60, 61, 62, 63]. These works aim at finding an optimal distribution of clock arrival time so that the current pulses from neighboring units will have less overlap and accumulation. Benini et al. [58] formulated the problem of finding optimal clock latencies as an NP-complete problem, and proposed a Genetic Algorithm (GA) to solve it. GA-based technique has long run time, and the objective function used requires pre-characterization of individual flip-flops for each circuit, which is also timing-consuming. Vittal et al. [59] formulated the problem into Integer Linear Programming (ILP) problem under the constraint of multi-domain clock skews. Both the complexity of ILP problem and extensive circuit simulations involved with the objective function also renders this approach impractical for large circuits. Huang et al. [60] accelerated this work by first applying ASAP and ALAP scheduling policies to prune the redundancies in ILP formulation, then further applying a zone-based scheduling algorithm to solve large circuits heuristically. Yu et al. [61] instead, seeks to achieve the statistically even distribution of clock skews by setting skew variance as the optimization objective, which is not linear. A heuristic that gradually tunes initial solution by peak trimming and valley filling is proposed to solve the problem. Rahimi [62] considers the load capacitances of flip-flops when spreading clock transitions. It creates a center-of-gravity constraint to help balance clock latencies around the center of the search interval. Mukherjee et al. [63] considers the impact on peak current from clock buffers, flip-flops and fanout combinational cells all at the same time by combining retiming and clock scheduling techniques.

Another group of works seek to change polarities of clock buffers by replacing them with inverters for peak current reduction and power/ground noise control [64, 65, 66, 67]. Nieh et al. [64] proposed an opposite-phase approach that changes clock buffer close to the root of clock tree into inverter, which successfully reduces peak current. However, the power/ground noises in a local area are not. Samanata et al. [65] observed that power/ground noises are sensitive to the locations of clock buffers and proposed to replace half of the buffers in some particular locations with inverters. However, it can render the clock skew out of control. Chen et al. [66] assigns signal polarities at the leaves of clock tree, so that peak current and power/ground noises can be reduced simultaneously, and clock skew can also be kept in control. Jang et al. [67] applies buffer sizing together with polarity assignment to the minimization of power/ground noises while satisfying the clock skew constraint.

Other techniques like MTCMOS and flip-flop resynthesis are also addressed in concern of peak current. Lu et al. [68] applies Mutiple Threshold CMOS (MTCMOS) technique by changing cells in non-critical paths to high V_t to reduce both peak current and leakage power in the circuit. Wu et al. [69] introduces long delay flip-flops to reduce peak current by flip-flop resynthesis. It is similar to traditional clock skew scheduling, however more advantageous in the sense that it is able to bypass hold time constraints. Moreover it can be applied to either logic synthesis stage or physical design stage.

However, very few peak current reduction works happen at FSM synthesis level. Since a sequential circuit can be represented by a set of finite state machines (FSM) at logic synthesis level, FSM state encoding (or re-encoding), which seeks to assign distinct binary codes to each state in the FSM, is a well know problem in sequential circuit optimization. Most early work (such as JEDI, MUSTANG, MUSE, etc.) focus on minimizing circuit area [73]. Many power-driven state encoding techniques that target at reducing the circuit switching activity have been proposed in the mid-90s. A representative work among them is the *POW3* encoding proposed by Benini and De Micheli [55]. A brief survey on power-driven state encoding can be found in a recently published work [74], where a novel approach to re-construct FSM for low power is proposed. [70] and [71] are the only two works we are aware of that target at minimizing peak current by FSM state encoding. Huang et al. [70] consider reducing peak current using FSM synthesis technique. They first conduct SPICE simulation to show that peak current in power line V_{dd} and V_{ss} are directly related to the maximal number of simultaneous charging and discharging, respectively, in the state registers. They then propose a state re-encoding technique to reduce peak current by reducing the maximum number of state registers that switch in the same direction. Lee et al. [71] first achieves the optimal solution that minimizes peak current by formulating the state encoding into SAT problem with pseudo-Boolean expressions, then further reduces the switching power without deteriorating the minimum peak current using an efficient SAT-based heuristic.

3.2 Preliminaries

In this section, we give the necessary background on FSM model and then describe the two most relevant works: peak current reduction by state re-encoding [70] and FSM re-engineering for switching activity and power reduction [74].

3.2.1 Finite State Machine Synthesis

FSM is a popular model for sequential circuit design. We use the state transition graph (STG) $G = (V, E, \{C_i\}, \{w_{ij}\})$ to represent an encoded FSM, where a node $v_i \in V$ represents a state s_i (or i for short); $C_i = c_{i0}c_{i1} \cdots c_{ik}$ is a $(k+1)$-bit code assigned to state s_i; a directed edge $(v_i, v_j) \in E$ between nodes v_i and v_j represents a *state transition* from state s_i to state s_j; and the non-negative value w_{ij} is its *transition probability.*

FSM synthesis consists of state minimization and state encoding. *State minimization* finds a functionally equivalent FSM that has the minimum number of states. *State encoding* assigns distinct codes to each state of the FSM such that the sequential circuit modeled by the FSM can be efficiently implemented in terms of area, performance, or power. *State re-encoding* is an approach to adjust the codes of certain states in order to optimize some objectives (e.g. power and performance) while the current coding scheme optimizes other objectives (e.g. area).

For a state transition $s_i \rightarrow s_j$, for convenience, we call s_i a *previous-state* of s_j, s_j a *next-state* of s_i, and they are *neighboring states* to each other. The transition's *switching activity* is defined as the Hamming distance $H(C_i, C_j)$ between their codes C_i and C_j. The *total switching activity* (TSA) of the encoded FSM is defined as the following weighted sum:

$$TSA = \sum_{(v_i,v_j)\in E} w_{ij} H(C_i, C_j) \tag{3.1}$$

Let $p_{ij}(0 \rightarrow 1)$ denote the number of bits switching from 0 to 1 during the state transition $s_i \rightarrow s_j$. That is, the number of bits that have value 0 in C_i but change to value 1 in C_j. Take the state transition from 101100 to 001010 for example, $p_{ij}(0 \rightarrow 1)$ equals one and is contributed by the second bit from the right. We can define $p_{ij}(1 \rightarrow 0)$ similarly, which equals two and are contributed by the first and fourth bits from the left. We denote the *peak switching value* on this transition $psv_{ij} = \max\{p_{ij}(0 \rightarrow 1), p_{ij}(1 \rightarrow 0)\}$. Obviously, $psv_{ij} = \max\{1, 2\} = 2$ in this example.

The *peak switching value* of the entire FSM can be defined as follows:

$$Peak \quad (0 \rightarrow 1) = \max_{(v_i,v_j)\in E} p_{ij}(0 \rightarrow 1) \tag{3.2}$$

$$Peak \quad (1 \rightarrow 0) = \max_{(v_i,v_j)\in E} p_{ij}(1 \rightarrow 0) \tag{3.3}$$

$$\begin{aligned} PSV &= \max_{(v_i,v_j)\in E} psv_{ij} \\ &= \max\{Peak(0 \rightarrow 1), Peak(1 \rightarrow 0)\} \end{aligned} \tag{3.4}$$

We say that a transition $s_i \rightarrow s_j$ is a *critical transition* (CT) if $p_{ij}(0 \rightarrow 1)$ or $p_{ij}(1 \rightarrow 0)$ reaches PSV.

The value of TSA is a good indicator of the sequential circuit's average dynamic power; the values of $Peak(0 \rightarrow 1)$ and $Peak(1 \rightarrow 0)$ have a strong correlation with peak current at the power line V_{dd} and V_{ss}, respectively [70], which is also validated by Figure 3.7 and Figure 3.8 in Section 3.6. Therefore, we choose PSV as the main optimization objective for peak current reduction. Meanwhile, we try to keep TSA as low as possible in order to minimize average dynamic power.

3.2.2 Peak Current Reduction by Re-Encoding

In [70], the authors propose to consider the switching directions of the state register (either go from logic 0 to 1 or go from logic 1 to 0) to reduce the number of state registers that switch in the same direction at the same time. That is, if they decide to change the bit (that is, the switching direction) on one state register, the code of every state in the FSM will flip at that bit position. Deciding which bit to be changed first follows greedy fashion: At each iteration, the unmarked bit that helps reduce the *peak swiching value* of the entire FSM most or reduce the number of *critical transitions* most with the same *peak switching value* is flipped and marked.

Table 3.1: Original codes from $POW3$ for circuit *ex5_n* (see Figure 3.1) and the new codes obtained by the state register re-encoding in [70].

state	0	1	2	3	4	5	6	7	8
$c_0c_1c_2c_3$	0000	0100	0011	0010	0001	0101	0111	0110	1110
$d_0d_1d_2d_3$	0101	0001	0110	0111	0100	0000	0010	0011	1011

For example, Table 3.2.2 gives the original codes $c_{i0}c_{i1}c_{i2}c_{i3}$ for each state i in the FSM shown in Figure 3.1 as well as the new codes $d_{i0}d_{i1}d_{i2}d_{i3}$ obtained by the re-encoding approach in [70], where we can see that the second and last state registers have been flipped (that is, $d_{i1} = c'_{i1}, d_{i3} = c'_{i3}$).

According to the original coding scheme, $PSV = 3$ on two CTs: 6 → 0 and 8 → 0, where three state registers change from logic 1 to 0 simultaneously. In the new coding scheme, the PSV is reduced to 2.

3.2.3 FSM Re-Engineering for Low Power

Yuan et al. [74] propose a novel re-engineering paradigm for FSM synthesis. Their idea is to construct a new FSM from the minimized FSM to relax the constraints on the most constrained portion of the original FSM. The basic technique they develop to achieve this is state replication, or state splitting as is called in their work. When it becomes hard to find a code for a state without causing large

Hamming distance to one or more of its neighboring states, they create a new state to take away some of the previous-states to make encoding easier and more efficient.

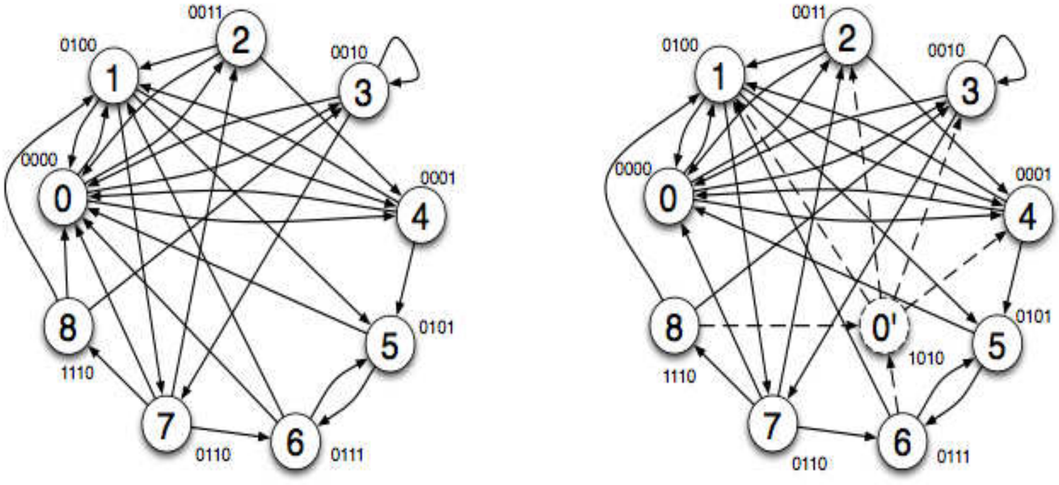

Figure 3.1: The 9-state STG representing MCNC benchmark circuit *ex5*. Left: the original STG with two CTs 6 → 0 and 8 → 0. Right: new 10-state STG where state 0' is a replicate of state 0 and PSV is reduced from 3 to 2.

For example, on right of Figure 3.1, state 0' is a replicate of state 0 with the two CTs being redirected to this new state. Originally, for transition 8 → 0, we have $H(1110, 0000) = 3$. Now we have the new transition $8 \rightarrow 0'$ with $H(1110, 1010) = 1$. So the TSA defined in Equation (1) would be reduced.

3.3 Two Basic Techniques for peak current reduction

Figure 3.1 depicts circuit *ex5* from the MCNC benchmark suite [77], where there are 9 states labeled from 0 to 8. The FSM is encoded by the power driven encoder POW3 [55] and the codes for each state can also be found in Table 1.We can see that on two CTs 6 → 0 and 8 → 0, there are three bits changing from 1 to

0. This means that three state registers will switch from logic level 1 to logic level 0 simultaneously, causing peak current at power line V_{ss}.

To reduce peak current, which corresponds to $PSV = 3$, it will be sufficient to make adjustments along each CT such that there will not be three or more state registers switching at the same direction at the same time. The two basic techniques we propose target directly the states that are involved in the critical transitions.

3.3.1 State Re-Encoding

Consider the CT 6 → 0, where the codes for the two states are 0111 and 0000, we can find a new code either for state 6 or state 0 to reduce the number of 1's changing to 0's and thus make this transition not critical. By checking the codes of previous-states of 6: {0101,0110} and its next-states: {0000,0100,0101}, we see that among the seven unused 4-bit codes, any one of the followings {1000,1001,1010,1100} can be used for state 6 such that all the state transitions that are involved with state 6 will not have more than two bits switching from 0 to 1 or from 1 to 0 at the same time. Similarly, we can achieve this by keeping the code of state 6 and re-encoding state 0 to be any of the followings: {1001,1010,1100}.

We refer to this technique as *state re-encoding* because of its nature of assigning a new code to the state. The new code to resolve a CT can be any valid code, not necessarily unused ones. We consider only unused codes in the above example for simplicity. If we use a code that is currently assigned to another state, this state will lose its code and we need to find a new code for it. We will elaborate this later.

If we have an n-state FSM where each state has a distinct k-bit code ($k \geq \lceil \log_2 n \rceil$), potentially each of the n states can be encoded by any of the 2^k k-bit codes. In another word, the state re-encoding space we can explore is as large as $\binom{2^k}{n} \cdot n!$.

Note that our approach is conceptually different from the term of state re-encoding used by Huang et al. in [70]. They indeed consider the switching direction of the state registers, not individual states. Consequently, in their solution $d_0 d_1 \cdots d_i \cdots d_{k-1}$, for any given bit position i, $d_i = c_i$ or $d_i = c_i'$ for all the states. They consider up to all the 2^k combinations of whether to flip each of the k state registers, which is a subset of our re-encoding solution space. Moreover, considering that 2^k is normally of the same order of n, 2^k is much less than $\binom{2^k}{n} \cdot n!$.

Another feature that distinguishes our state re-encoding and that in [70] is the impact on TSA. In [70], because they re-encode the state registers, the Hamming distance between any pair of states will not change. From Equation 3.1, the TSA will not change. In our approach, we re-encode individual states and may change the value of TSA. This brings the challenge of how to resolve the CT without increase or with the minimum increase on TSA. On the other hand, it gives us the opportunity to reduce TSA at the same time. Therefore, when there are multiple options of new code for state re-encoding, we will select the one that will reduce TSA or cause the minimum increase of TSA. We will discuss this in-depth in the next section.

3.3.2 State Replication

The FSM re-engineering technique proposed in [74] seeks to reduce the FSM's TSA. It consists of three steps, first, identify a most constrained state s whose code contributes the most to TSA; then add a new state s' to replicate state s and connect s' to all the next-states of s; finally assign a code to s' and split the previous-states of s such that each previous-state goes to either s or s'.

We modify this approach to resolve CT and reduce PSV. Firstly, the state to be replicated is set to be the ending state of as many CTs as possible. Consider the example in Figure 3.1, the target state is state 0 as the two CTs are $6 \rightarrow 0$ and $8 \rightarrow 0$. We add a new state $0'$ and connect $0'$ to all the next-states of 0. Then we redirect the CTs $6 \rightarrow 0$ and $8 \rightarrow 0$ to $6 \rightarrow 0'$ and $8 \rightarrow 0'$, respectively. Finally we let state $0'$ go into all the next-states of state 0. Now we only need to assign $0'$ a code such that on all the transitions involving $0'$, there will not be two or more bits switching from 1 to 0 (or from 0 to 1) at the same time. It is not hard to verify that codes 1010, 1100, or 1001 can all satisfy this requirement and there is no transition causing more than two bits switching at the same direction, which means that PSV is reduced. Note that by introducing a replicate of state 0, it enables us to split the previous-states of state 0 and makes both state 0 and its replicate less constrained. We refer to this technique as *state replication*.

For a CT $i \rightarrow j$, *state re-encoding* can be applied to both states i and j; *state replication* can only be applied to state j. This is because that a state and its replicate will have the same set of next-states, replicating state i alone will not make the transition $i \rightarrow j$ non-critical.

Finally, both techniques have its pros and cons and it is a non-trivial problem to combine the two techniques to achieve our optimization goal. Details are discussed in the following section.

When there are multiple codes available to make a CT non-critical, we can choose the one that contributes the least to TSA. This is the reason that we encode the replicate state $0'$ by 1010 in the above example. We can also seek further reduction in TSA by redirecting other transitions into state 0 into state $0'$ instead, if the hamming distance of latter is smaller. It suggests that state replication has the potential to reduce both peak current and average dynamic power. Normally state replication is applied when there are unused codes available, that is, the number of states in the FSM is not a power of 2. Because otherwise it will increase the number of encoding bits and introduce additional state registers. We restrict our discussion under this constraint for simplicity.

The solution space defined by state replication can be very large. If the FSM has n states and encoded with k-bit codes. Let p be the number of states that are the ending state of a CT, and q be the total number of previous-states that those p states have. The search space will be roughly $\binom{2^k-n}{p} \cdot p! \cdot q$. The first term is the ways to find unused code for the replicate states; term p! is the different ways to assign each replicate states a code; determining whether a previous-state should be re-directed to the replicate state can be done by checking whether the CT is resolved and TSA can be reduced. Both state re-encoding and state replication give large solution space, in the next section, we present how to explore such space efficiently.

3.4 Simultaneous State Replication and Re-Encoding

As mentioned earlier, our proposed approach aims at reducing an encoded FSM's PSV while keeping the increment of TSA, if any, as little as possible. We start with an encoding scheme that minimizes the TSA, then perform simultaneous state replication and re-encoding (*SSRR*) so that the psv_{ij} can be reduced along each of the CT $state_i \rightarrow state_j$. Since these adjustments can cause the increase of TSA, we make the adjustment local to minimize this impact. However, global adjustment of the FSM is often unavoidable due to the limited number of codes available for state replication and re-encoding. In this section, we describe how we seamlessly combine these two basic techniques to balance the local and global adjustments.

3.4.1 Overview of the Approach

The proposed *SSRR* approach is depicted in Figure 3.2. Recall that *Peak Switching Value* (PSV) is the maximal number of bits that switches from 1 to 0 (or from 0 to 1) during a single transition. We define the *working set* S as the set of transitions in the given encoded FSM that need to be adjusted for PSV reduction. It intially contains all the *critical transitions* (CTs). In the later phase, transitions whose two involved states have one or both codes missing are also included. We define the *solution pool* SP as the set of codes that can be used for the states such that one or more transitions can be removed from the working set S.

Starting with an FSM that has been encoded (for example, to minimize TSA), we first identify all the CTs. This can be done by checking each state transition

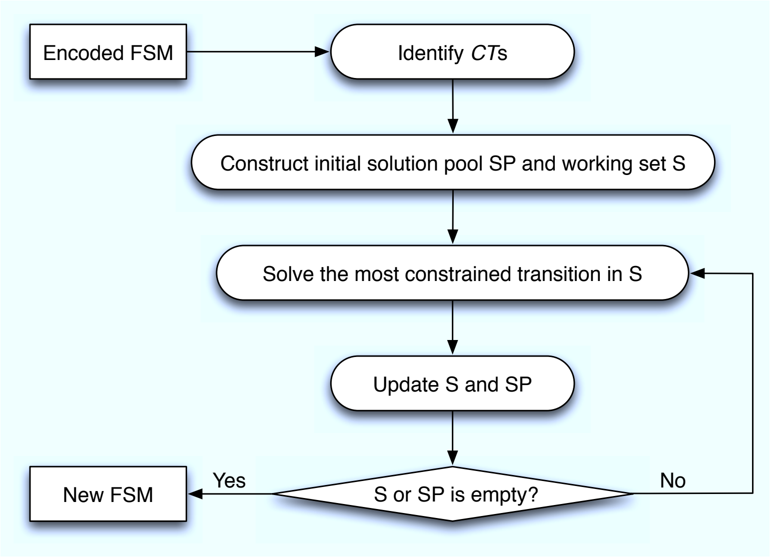
Encoded FSM
Identify *CT*s
Construct initial solution pool SP and working set S
Solve the most constrained transition in S
Update S and SP
S or SP is empty?
Yes
No
New FSM

Figure 3.2: Overview of the simultaneous state replication and re-encoding approach.

with a complexity of $O(k \cdot m)$, where m is the number of state transitions in the FSM and k is the number of bits in the code. These CTs form the initial working set S. Next we construct the corresponding solution pool SP based on how each CT in S can be resolved. The complexity in this step is bounded by $O(2^k \cdot m)$ when S has all the m transitions and each of them can be reduced by any of the k-bit code. Then we pick the most constrained transition from S and use the least constraining solution from SP to adjust the states involved in the transition. We will show that this has a complexity of $O(n+m)$, where n is the number of states in the FSM. We update both S and SP and continue this procedure until S becomes empty, which means that the PSV has been reduced, or SP becomes empty which means that we fail to reduce PSV. As we will show in Section 3.4.4, the complexity is bounded by $O(m^2)$ and there are at most $2m$ rounds, so the complexity of this step is $O(m^3)$. Note that normally code length k is close to $\lceil \log n \rceil$ and $m > n$, so the overall complexity of our approach is bounded by $O(m^3)$.

3.4.2 Initial Construction of S and SP

Consider an FSM with n states encoded with code of length $k \geq \lceil \log_2 n \rceil$, we say that there are states AVAILABLE for state replication without increasing code length if $n < 2^k$; for each of the 2^k k-bit code, we say that a code is USED if the code has been assigned to some state, otherwise the code is UNUSED.

For each transition $state_i \rightarrow state_j$ (denoted as T_{ij}) in S, we say that a code is a *proper code* for $state_j$ if $state_j$ will not be involved with any transitions in the working set S when it is assigned this code. The solution pool SP initially contains all the proper codes for states involved in CTs in S. They can be found by one of the following three actions:

Replicate $state_j$**.** If there is state AVAILABLE, we first make a copy of $state_j$ as $state'_j$; add a transition from $state'_j$ to all the next-states of $state_j$; let $state_i$ move to $state'_j$ instead of $state_j$ if $state_i \rightarrow state_j$ is a CT. Then we find all the UNUSED *proper codes* for $state'_j$ and denote this set of codes as $SP.rep(T_{ij})$. This state replication strategy ensures that it is a local adjustment and will not trigger further state adjustments of other states. The advantage is that it preserves the original coding scheme which is optimized for TSA for example.

Re-encode $state_j$**.** For each code c_k, we check whether it is a *proper code* for $state_j$. If so, we will add code c_k into either the set $SP.rec0(T_{ij})$ or the set $SP.rec1(T_{ij})$ based on c_k is UNUSED or USED. The key difference between $SP.rec0(T_{ij})$ and

$SP.rec1(T_{ij})$ is their impact on the rest of the state adjustment process. Re-encoding by an UNUSED code is a local adjustment while re-encoding with a USED code c_k in $SP.rec1(T_{ij})$ can be more complicated (we will elaborate this as R2 in Section 3.4.4).

Re-encode $\boldsymbol{state_i}$**.** The peak switching along T_{ij} can also be reduced by re-encoding $state_i$. Similar to re-encoding of $state_j$, we can add UNUSED proper codes for $state_i$ into $SP.rec0(T_{ij})$ and USED proper codes into $SP.rec1(T_{ij})$. Finally, as we have discussed earlier, replicating $state_i$ will not help to resolve T_{ij}.

At the end of this stage, working set S is constructed as the set of all the CTs found in the first step. For each T_{ij} in working set S, $SP.rep(T_{ij}) \cup SP.rec0(T_{ij}) \cup SP.rec1(T_{ij})$ gives all the possible ways to reduce the psv_{ij} along T_{ij}.

3.4.3 The Most Constrained Transition and the Least Constraining Solution

For each transition T_{ij} in the working set S, we define its *level of constrainess* as the number of codes contained in its corresponding SP. Since the transitions in initial S have $SP.rep$ as a superset of $SP.rec0$, their level of constrainess can be calculated as $|SP.rep| + |SP.rec1|$. For the transitions that are added into S later (see Section 3.4.4 for when a transition needs to be included into S), we set its $|SP.rep| = 0$ and the level of constrainess will be $|SP.rec0| + |SP.rec1|$. In sum, we can define the level of constrainess of T_{ij} in S as

$$loc(T_{ij}) = \max\{|SP.rep(T_{ij})|, |SP.rec0(T_{ij})|\} + |SP.rec1(T_{ij})| \qquad (3.5)$$

The most constrained transition is the one with the minimal number of solutions. By adjusting the most constrained transition first, we tend to allocate a code, if it can be used to adjust multiple transitions in S, to the one that needs it the most. This is an effective way to utilize the limited number of codes available.

For each code c_k in the solution pool SP, we define its *level of constrainess with respect to transition* T_{ij} as

$$\sum_{c_k \in SP.rep(T_{pq}) \cup SP.rec0(T_{pq}) \cup SP.rec1(T_{pq}), T_{pq} \neq T_{ij}} \frac{1}{loc(T_{pq}) - 1} \tag{3.6}$$

If $loc(T_{pq}) = 1$ and the code c_k is the only solution to adjust transition T_{pq}, we can replace the expression of $\frac{1}{loc(T_{pq})-1}$ by a large number, say the number of states in the FSM for example. Clearly, the value of $\frac{1}{loc(T_{pq})-1}$ measures how important a code is for the adjustment of transition T_{pq}. The level of constrainess w.r.t. T_{ij} defined above measures the impact on other states by assigning code c_k to T_{ij}. The least constraining code for T_{ij} is a code in SP that can be used to adjust T_{ij} and has the minimal value of *level of constrainess* w.r.t. T_{ij}.

Figure 3.3 illustrates how we resolve the most constrained CT in S by the least constraining solution from SP. This combined with the update of S and SP in Section 3.4.4 will be performed iteratively until we reduce the PSV or fail to do so. We resolve the most constrained T^*_{ij} found in line 1 by the least constraining solutions from $SP.rep(T^*_{ij})$, $SP.rec0(T^*_{ij})$, and $SP.rec1(T^*_{ij})$, in the order of priority. State replication is of the highest priority because by splitting the incoming transitions, there will be fewer constraints on the code selection for the replicated state, and it can be more likely to find a code that can reduce both TSA and PSV. However,

```
 1. Select the most constrained transition $T^*_{ij}$ from $S$;
//case I: Use state replication strategy
 2. if $(SP.rep(T^*_{ij}) \neq \phi)$
 3.       Replicate $state^*_j$ into $state^{*'}_j$;
 4.       Select the least constraining code $c^*_k$ from
            $SP.rep(T^*_{ij})$ for $state^{*'}_j$;
 5.       for each next-state $state_i$ of $state^*_j$
 6.           Add transition $state^{*'}_j \rightarrow state_i$;
 7.       for each previous-state $state_i$ of $state^*_j$
 8.           if $(H(i, j^{*'}) < H(i, j^*))$
 9.               Add transition $state_i \rightarrow state^{*'}_j$;
10.               Delete transition $state_i \rightarrow state^*_j$;
11.       Mark the code $c^*_k$ USED;
//case II: Use state re-encoding with UNUSED code
12. else if $(SP.rec0(T^*_{ij}) \neq \phi)$
13.       Select the least constraining code $c^*_k$ from
          $SP.rec0(T^*_{ij})$ for $state^*_j$;
14.       Mark the code $c^*_k$ USED;
//case III: Use state re-encoding with USED code
15. else if $(SP.rec1(T^*_{ij}) \neq \phi)$
16.       Select the least constraining code $c^*_k$ from
          $SP.rec1(T^*_{ij})$ for $state^*_j$;
17.       Mark the code $c^*_k$ DONE;
//for case II and III
18. if ($state^*_j$ has its original code $c_l$)
19.       Mark the code $c_l$ UNUSED;
```

Figure 3.3: Solving the most constrained CT with the least constraining solution.

since the number of extra states in a given FSM is limited, state replication is not always a feasible option. State re-encoding with USED code, however, can always be applied. But it is of the lowest priority because it can trigger subsequently many code adjustments. Starting from a TSA-minimized encoding scheme, more changes on the state's code may bring more overhead to TSA. Therefore in line 17, once a USED code is re-assigned to a different state, we mark the code DONE and will not include it in SP as a potential solution to adjust any CTs. This constraint guarantees a bounded complexity. The computation is elaborated at the end of Section 3.4.4. Figure 3.3 is easy to follow and its complexity is bounded by $O(n+m)$.

We mention that other techniques have also been adopted to keep the increase of TSA as little as possible when we select the least constraining solution. For example, always selecting code c_k^* that gives the minimal switching activity when there is a tie; shifting transitions from previous-state $state_i$ of $state_j$ to its replicate $state'_j$ if $H(i,j')$ is smaller than $H(i,j)$ during state replication (lines 7-10).

3.4.4 Update S and SP

After the most constrained transition T_{ij}^* is adjusted, we need to update both the working set S and the solution pool SP before we can proceed to the next iteration. Here we list the set of rules for the updating stage.

R1. In state replication, when there is no state AVAILABLE for replication. Change $SP.rep(T_{ij}) = \phi$ for all $T_{ij} \in S$.

R2. When a state is re-encoded with USED code c_p, which is currently used to encode $state_p$, identify all transitions T_{ij} with $i = p$ or $j = p$. There are following two cases we need to consider:

1) If $T_{ij} \notin S$, add T_{ij} into S, construct $SP(T_{ij})$.

2) If $T_{ij} \in S$, re-construct $SP(T_{ij})$.

R3. If $state_k$ is re-encoded, for all $T_{ij} \in S$, $i = k, j = l$ or $i = l, j = k$, there are following two cases:

1) If $state_l$ has its code, delete T_{ij} from S and $SP(T_{ij})$ from SP.

2) If $state_l$ has its code missing, re-construct $SP(T_{ij})$.

R4. For all the codes that have changed their status in Figure 3.3, the corresponding entries in SP have to be updated.

R2 adds new transitions into S. Different from the CTs added into S initially, the newly added transitions T_{ij} are those with either $state_i$ or $state_j$'s code missing. It is also possible that both $state_i$ and $state_j$'s codes are missing. It happens when $State_i$'s code is first deprived to re-encode some state, while $state_j$'s code is next deprived to re-encode some other state before $state_i$ has been assigned a code.

Constructing and re-constructing $SP(T_{ij})$ in R2 and R3 involves the following two cases:

1) If both $State_i$ and $State_j$ have their codes missing, include the union of USED or UNUSED *proper code*s for $State_i$ and those for $State_j$. Here transition T_{ij} is not examined when looking for *proper code*s.

2) If either $State_i$ or $State_j$ has its code missing, without loss of generality, we assume $State_i$, $SP(T_{ij})$ will include all the USED or UNUSED *proper code*s for $State_i$.

Figure 3.4 depicts the implementation of R4.

Lines 1-6 are the updates performed when a code is first assigned to a state. Lines 7-12 illustrate the updates performed when a code is released by one state. Lines 13-14 discuss the case when a code is transferred from one state to another.

```
//$c_k$ status change: UNUSED to USED
 1. for each $SP.rep(T_{ij}) \cup SP.rec0(T_{ij})$ that contains $c_k$
 4.       add $c_k$ to $SP.rec1(T_{ij})$;
 5.       delete $c_k$ from $SP.rep(T_{ij})$ if any;
 6.       delete $c_k$ from $SP.rec0(T_{ij})$ if any;
```

```
//$c_k$ status change: USED to UNUSED
 8. for each $SP.rec1(T_{ij})$ that contains $c_k$
 9.       add $c_k$ to $SP.rec0(T_{ij})$;
10. if (there is state AVAILABLE &&
      both $State_i$ and $State_j$ have their codes &&
      $T_{ij}$ is no longer critical if $state_j$ is encoded with $c_k$)
11.       add $c_k$ to $SP.rep(T_{ij})$;
12.       delete $c_k$ from $SP.rec1(T_{ij})$;
```

```
//$c_k$ status change: USED to DONE
14. delete $c_k$ from all the entries in $SP$;
```

Figure 3.4: Update code information.

Line 14 ensures that once one code is transferred, it is settled and won't be used in re-encoding again. This prevents the algorithm to enter an endless loop such as a cyclic state re-encoding operations, where two or more states need each other's code and form a circle.

All the update rules are bounded by $O(m)$ in the worst case, and there will at most m transitions to be updated. So the update process is bounded by $O(m^2)$ for each round of *SSRR* algorithm.

Since the program terminates when S becomes empty and we will not add the same transition to S more than twice, there will be $O(m)$ rounds to solve all the transitions that have been put into S. Therefore, overall complexity of the proposed *SSRR* algorithm will be $O(n \cdot m) + O(m \cdot (m + n)) + O(m \cdot m^2) = O(m^3)$. In practice, the runtime is much less than this theoretical bound and is almost trivial. Since *SSRR* algorithm aims at reducing PSV by one, we may perform several iterations of *SSRR* algorithm until no further reduction can be achieved.

3.5 Illustrative Example for SSRR

In this section, we present a small example to illustrate step by step how the proposed SSRR approach can reduce PSV of an encoded FSM.

Fig. 3.5 shows part of a 14-state FSM already encoded using four encoding bits. Therefore there are two states AVAILABLE for state replication without increasing number of encoding bits. The two UNUSED codes are 1000 and 1100. Currently *PSV* equals 2 and is expected to be reduced to 1. There are only two CTs in the FSM, which are both included in the figure indicated by dashed arrows. The transition probabilities are also given in the figure. It can be noted that the sum over all transition probabilites is less than 1, because the figure only includes part of the FSM.

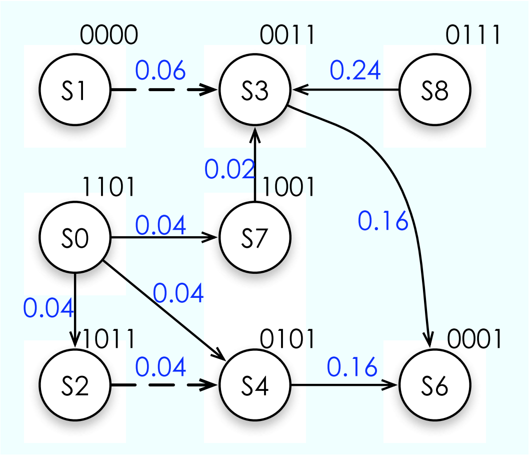

Figure 3.5: Part of a 14-state FSM already encoded.

3.5.1 Initial Construction of S and SP

Initially *Working Set* S contains the two CTs $T_{1,3}$, $T_{2,4}$. They are also marked as in C_{INI} for later usage. The initial *Solution Pool* SP for each CT can be constructed by exhausting all the solution options provided by state replication and state re-encoding. Table 3.2 lists $SP.rep$, $SP.rec0$ and $SP.rec1$ for $T_{1,3}$ and $T_{2,4}$ respectively.

Table 3.2: Initial *Working Set* and *Solution Pool* for FSM in Fig. 3.5

S	$T_{1,3}$	$T_{2,4}$
$SP.rep$	1000	–
$SP.rec0$	1000	1100
$SP.rec1$	0001, 0010, 0011 0101, 0110, 0111 1001, 1010, 1011	0101, 0111 1001, 1101

The second row indicates that state replication is infeasible for $T_{2,4}$, because neither of the two UNUSED codes can be assigned to replicated state of S_4 without reaching PSV of 2. The third and fourth rows list sets of eligible codes, UNUSED or USED, to re-encode either state involved with the CT.

3.5.2 The First Iteration

Each iteration involves two steps: Resolving the most constrained transition from S using the least constraining code from the corresponding SP; Updating S and SP.

Since the level of constrainess regarding a transition indicates the size of its corresponding SP, the most constrained transition refers to the one with the smallest SP size. In the illustrative example, $|SP|(T_{2,4}) = 5 < |SP|(T_{1,3}) = 10$, therefore $T_{2,4}$ is the most constrained and should be resolved first.

Meanwhile, the level of constrainess regarding a code measures the impact on resolving other transitions by assigning it to the target transition, therefore the least constraining code can be interpreted as the one least shared by transitions remaining in the S. Among the five solutions in SP of $T_{2,4}$, based on the fact that 1100 does not appear in SP of $T_{1,3}$ and is UNUSED, it will be employed to resolve $T_{2,4}$. So in the first iteration, S_2 is re-encoded by 1100 and CT of $T_{2,4}$ is resolved.

At the end of this iteration, $T_{2,4}$ is deleted from S, and its corresponding entry in SP is also deleted from SP. Code 1100 becomes USED, while the original code of S_2 1011 becomes UNUSED. The SP of $T_{1,3}$ is updated accordingly as shown in Table 3.3.

Table 3.3: S and SP at the end of the first iteration

S	$T_{1,3}$
$SP.rep$	1000
$SP.rec0$	$1000, 1011$
$SP.rec1$	$0001, 0010, 0011$ $0101, 0110, 0111$ $1001, 1010$

3.5.3 The Second Iteration

As $T_{1,3}$ is the only transition remaining in S, the entire procedure would stop if $T_{1,3}$ can be resolved in the second iteration. According to the solution priority in Fig. 3.3, state replication is applied by performing the following actions: Replicate S_3 into S_3'; Encode S_3' is by 1000 from $SP.rep$; Redirect S_1 and S_7 into S_3' instead. The resultant FSM is depicted in Fig. 3.6, with PSV successfully reduced to 1.

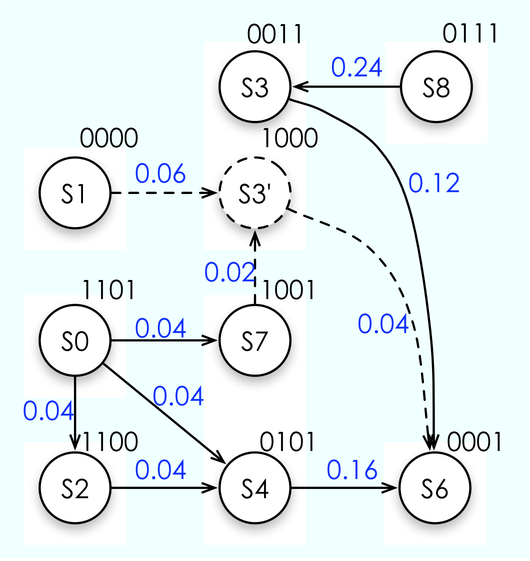

Figure 3.6: New FSM with reduced PSV.

3.5.4 Result Evaluation

It can be easily verified that the new FSM is functionally equivalent with the original one. Though the new FSM incur one extra state, the number of encoding bits remains as four, which indicates the same number of state registers in the synthesized circuit. Moreover, PSV has been reduced, so that peak current and peak power in the synthesized circuit can be expected to be reduced. TSA, which corresponds to average power in the circuit, can also be compared. In the original FSM, TSA with respect to all transitions that appear in Fig. 3.5 can be evaluated as: $0.04\times(1+1+2+3)+0.02\times2+0.16\times(1+1)+0.24\times1+0.06\times2 = 1.0$, while TSA in Fig. 3.6 equals: $0.04\times(1+1+1+2+2)+0.02\times1+0.16\times1+0.24\times1+0.12\times1+0.06\times1 = 0.88$, a 12% reduction.

3.6 Experimental Results

The main objective of the experiments is to validate (1) the observation that state registers are the major contributor to peak current in the circuit, and there is a strong correlation between peak current and the maximum number of state bits switching in the same direction at the moment of state transition; (2) the effectiveness of the proposed *SSRR* technique in reducing PSV while also considering TSA.

We demonstrate both (1) and (2) by comparing our approach with *POW3* encoding algorithm[55] and the state register re-encoding approach (*SR* for short) [70]. We first demonstrate (1) using the simulation results of *s512* from MCNC benchmarks. Since there is only one single operating voltage in our circuits, we measure the peak power to represent peak current using an industry gate-level transient power analysis tool Synopsys PrimeTime-PX [76].

Figure 3.7 is the transient power waveform of the circuit *s510* whose FSM is encoded using *POW3* [55] and *SR* [70]. They are the same because *SR* fails to improve the encoding scheme in the case of *s510*. The waveform on top is the

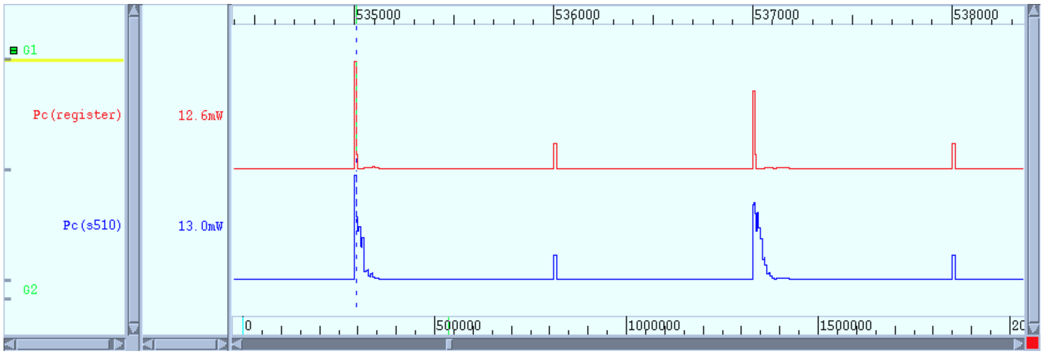

Figure 3.7: Power waveform in circuit s510 encoded by *POW3* and *SR*. Top waveform is the power in the circuit; bottom waveform is the power in the state registers only. Top x-axis is the simulation time. The peak power in the circuit is 13.1mW and the peak power in the state registers is 12.6mW. The maximum number of state registers switching in the same direction is **3**.

power of the entire circuit and the waveform at the bottom is the power of the state registers only. The top x-axis is the simulation clock time. We can clearly see that the peak power of the entire circuit (13.1mW, the 13.0mW reported on the figure is due to truncation) and the peak power of the state registers (12.6mW) both occur at the simulation time 535 *ns*, which is at clock rising edge. The peak power occurs when the FSM is switching from state '000010' to state '011110', which causes three state registers (the 3rd, 4th and 5th state registers) to switch from bit 0 to 1 at the same time.

Figure 3.8 shows the transient power waveform of the circuit *s510* whose FSM is encoded using our *SSRR* approach. The maximum number of state bits switching in the same direction is 2 when the FSM is transitioning from state '001000' to state code '011010' (the 2nd and 5th register are switching from 0 to 1). The peak power in the entire circuit (11.5mW, the 11.4mW reported on the figure is due to truncation) as well as the state registers (11.5mW, the 11.4mW reported on the

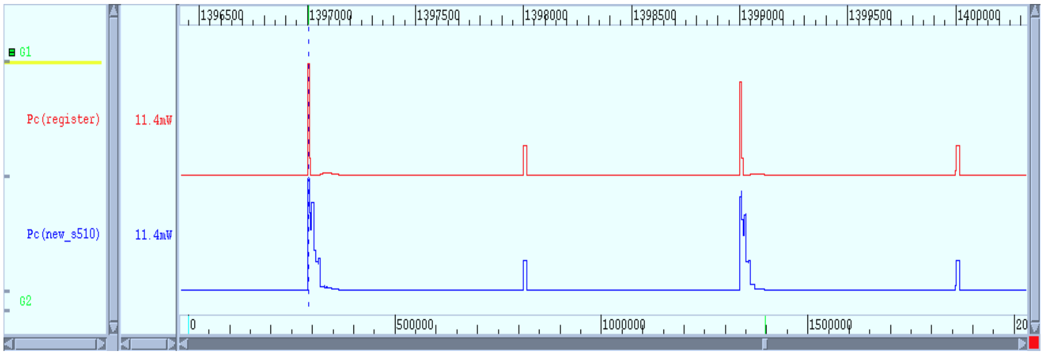

Figure 3.8: Power waveform in circuit s510 encoded by *SSRR*. Top waveform is the power in the circuit; bottom waveform is the power in the state registers only. Top x-axis is the simulation time. The peak power in the circuit and the state registers are both 11.5mW. The maximum number of state registers switching in the same direction is **2**.

figure is due to truncation) occur at simulation time 1397 *ns* [1]. The magnitude of circuit peak power is 12% smaller than that in Figure 3.7, which demonstrates the strong correlation between peak current and maximum number of state switching bits.

To validate the effectiveness of our approach, we apply all of the three encoding methods: *POW3*, *SR* [70] and *SSRR* to 52 MCNC FSM benchmarks[77]. We also implemented *SAT* method [71] to provide the optimal results. *SAT* method transforms the encoding problem into a Pseudo Boolean (*PB*) formulation, and employ *PB* solver to get the optimal results. However, it involves high computational effort, and therefore is limited to solve up to 8-bit encoding. We reported the result in Table 3.4. The first four columns list the name, the number of states (n), the

[1] Note that, in this case, the peak power in state registers and in the entire circuit are the same. This is because the input values for this state transition remain the same as the previous state transition, and therefore no combinational gate is switching before the current state registers get updated.

number of transitions (m), and the code length (k, in bits) for each *POW3* encoded FSM. These values are the same as in *SAT* and *SR* encoded FSMs. The fifth column (n') shows the number of states in *SSRR* encoded FSM, where the increased number of states is brought by state replication.

The *PSV* values in the encoded FSMs using *POW3*, *SAT*, *SR*, and *SSRR* respectively are reported in the next four columns. Though the *SAT*-based approach is of exponential complexity and cannot handle circuits with more than six state registers, its results are optimal in terms of *PSV* if the number of states are the same. It helps us study how far off our *SSRR* solution is from the optimal. Overall, *SR* approach can successfully reduce *PSV* in **27** FSMs with an average reduction of 24.5%; *SSRR* approach can reduce *PSV* in **34** FSMs with an average 39.2% reduction; Optimally, *SAT* approach reduces *PSV* in **39** FSMs with an average 46.3% reduction. However, in 6 out of 34 FSMs where *SSRR* can achieve smaller *PSV* than *POW3*, *SAT* is able to achieve further *PSV* reduction. It is worth noted that *SSRR* actually achieves smaller *PSV* than *SAT* for *s298*. It is a strong proof of the contribution made by state replication. The next column (*improv*) shows the improvement of *SSRR* over *SR*. We can see that *SSRR* is able to further reduce *PSV* in 18 FSMs over *SR* with an average improvement of 18.4%.

One advantage of *SR* algorithm is that it does not increase *TSA* in the *POW3* encoded FSM. We report the *TSA* in FSMs encoded by *SSRR* compared to FSMs encoded by *SR* in the last column. The result shows that *SSRR* on average has only 3% increase in *TSA* and in 13 FSMs it even has less *TSA* compared to *SR* and *POW3*. This again gives proof that state replication takes effect. Finally, the runtime of the *SSRR* algorithm is very small. It can finish all these benchmarks in a couple of seconds.

3.7 Summary

We focus on the problem of how to reduce peak current at the FSM level of the circuit by minimizing the peak switching value (PSV) of the FSM. For state transitions that reach PSV, we perform either state replication or state re-encoding to reduce its peak switching activity. We develop a most constrained least constraining heuristics to combine these two techniques seamlessly. The proposed algorithm is very effective in reducing FSM's PSV while keeping the increment on TSA under control. In 36 MCNC benchmarks whose PSV is not minimized by POW3, the only other known approach [70] reports better solutions on 27 circuits. Our algorithm outperforms this approach by finding better solutions on 34 circuits, out of which 21 are better than the solutions found by [70]. This is achieved with only an average of 0.5% increase in *TSA*.

Table 3.4: Comparison of different encoding techniques on the 39 circuits that *PSV* has not reached minimal by *POW3*.

circuit	n	m	k	n'	*PSV*					*TSA*	runtime
					POW3	*SAT*	*SR*	*SSRR*	improv.	ratio	/ *ms*
bbsse	16	56	4	16	4	2	3	2	33.3%	1.03	0.450
bbtas	6	24	3	7	2	1	1	1	0.0%	1.00	0.119
cse	16	91	4	16	4	2	3	2	33.3%	1.18	0.476
dk16	27	108	5	30	3	2	3	2	33.3%	0.89	0.821
dk27	7	14	3	8	3	1	2	2	0.0%	1.00	0.107
dk512	15	30	4	16	3	2	3	2	33.3%	0.99	0.177
donfile	24	96	5	29	4	2	3	2	33.3%	1.02	0.958
ex1	20	138	5	24	4	2	3	2	33.3%	1.03	1.134
ex2	19	74	5	23	4	2	3	2	33.3%	1.00	0.766
ex3	10	37	4	12	3	2	2	2	0.0%	0.99	0.171
ex4	14	21	4	24	2	1	2	2	0.0%	1.09	0.171
ex5	9	33	4	10	3	2	2	2	0.0%	1.00	0.175
ex6	8	34	3	8	3	2	2	2	0.0%	1.08	0.123
ex7	10	37	4	11	3	2	2	2	0.0%	1.00	0.215
keyb	19	170	5	20	4	2	3	2	33.3%	1.00	0.910
kirkman	16	367	4	16	2	1	1	1	0.0%	0.89	0.500
lion9	9	25	4	11	2	1	1	1	0.0%	0.83	0.227
mark1	15	22	4	15	3	2	3	2	33.3%	1.15	0.172
opus	10	22	4	10	2	1	2	2	0.0%	1.00	0.162
planet	48	115	6	48	2	1	2	2	0.0%	1.00	1.472
planet1	48	115	6	48	2	1	2	2	0.0%	1.00	1.471
pma	24	73	5	24	3	1	3	2	33.3%	0.90	0.506
s1	20	107	5	23	3	2	3	2	33.3%	0.97	0.445
s1488	48	251	6	51	5	2	3	2	33.3%	1.88	2.790
s1494	48	250	6	56	5	2	3	2	33.3%	1.03	5.479
s1a	20	107	5	23	3	2	3	2	33.3%	0.97	0.480
s208	18	153	5	18	3	2	2	2	0.0%	1.30	0.617
s298	218	1096	8	229	6	5	5	4	20.0%	0.89	41.20
s386	13	64	4	14	3	2	2	2	0.0%	1.01	0.203
s420	18	137	5	20	4	2	3	2	33.3%	1.00	0.897
s510	47	77	6	49	3	1	3	2	33.3%	0.99	1.578
s8	5	20	5	5	2	1	2	2	0.0%	1.00	0.114
s820	25	232	5	30	4	2	3	2	33.3%	0.99	1.794
s832	25	245	5	26	4	2	3	3	0.0%	1.00	0.651
sand	32	184	5	32	4	2	3	3	0.0%	0.98	0.619
sse	16	56	4	16	4	2	3	2	33.3%	1.03	0.422
styr	30	166	5	31	4	2	3	2	33.3%	1.00	0.609
tbk	32	1569	5	32	5	3	4	3	25.0%	0.92	3.845
tma	20	44	5	21	3	1	2	2	0.0%	1.00	0.352
average									17.4%	1.03	1.881

Chapter 4

Improving Dual V_t Technology by Simultaneous Gate Sizing and Mechanical Stress Optimization

When technology scales down to 90nm and beyond, leakage power starts to dominate dynamic power and dual V_t technique is projected to reduce leakage consumption. This is based on the fact that high V_t cells are less leaky compared to the low V_t cells. However, it can only be applied to off-critical path because high V_t cells have longer delay. This dual V_t technology has been well-researched in the past decade for leakage and total power minimization under timing constraint [79, 112]. In this work, we propose to explore performance enhancing techniques, gate sizing and mechanical stress used as examples, to improve dual V_t technology for further leakage reduction. The key idea is to apply gate sizing and mechanical stress to create delay slacks such that we will be able to assign high V_t to cells that would otherwise be assigned low V_t due to the timing constraint.

Gate sizing is a technique where we increase the channel widths of transistors and thus the current driving strength. It is very effective to improve circuit delay and dynamic power consumption [78, 38]. Many researches have been conducted to combine gate sizing with dual V_t technique for total power minimization. This problem has been known to be a Mixed-Integer Non-Linear Programs (MINLP) due to the discrete V_t levels and gate sizes, which is of high complexity. One category of works employ sensitivity-based heuristics to guide the selection of gates to be changed, which can be computationally efficient, but far from optimal due to its greedy nature [80, 81]. The methods focus on the isolated impact on delay and power of one cell, ignoring the overall impact on the circuit and other cells. Another category of works first approximate the discrete problem into continuous formulations, solve the continuous formulation using either optimization techniques like Lagrangian Relaxation [82, 83] or heuristics [84], and then snap the results. Although it shows in [84] that the results of most gates have snapped by themselves except the few due to sizing constraints or with fixed input drivers, significant descretization errors still cannot guarantee to be eliminated. There is also a category of works that employ combinatorial algorithms [85, 86], which would enumerate and propagate all possible cases while discarding impossible ones with certain pruning criterions. They are normally able to achieve very optimal results, however, at the cost of expensive runtime.

While gate sizing and dual V_t has already been commonly leveraged during physical synthesis stage (iterations of combined synthesis and physical design) as a tuning strategy for timing closure, it becomes more and more challenging to solely depend on these two techniques due to more and more tight timing budget. As device densities become much larger, there is not much space for sizing up the gates. Meanwhile, sizing up a certain gate contributes to larger load for all the gates in its fan-in cone, which may in turn require further sizing up the fan-in gates. This can make the timing optimization procedure very difficult to converge and fail. Similarly, as supply voltage level becomes much lower under 45nm technology, there is not much space for lowering threshold voltage to trade timing either. These facts necessitates the introduction of a third optimization technique.

Process-induced mechanical stress is a novel technique induced into CMOS channel that enhances carrier mobility [87, 88, 89]. On one hand, the enhanced carrier mobility leads to increased saturation drain current, which will in turn help reduce transistor delay; On the other hand, it also results in larger subthreshold drain current, which indicates larger leakage power consumption. This delay/power tradeoff makes mechanical stress a potential alternative to gate sizing and dual V_t. Compared to gate sizing, mechanical stress incurs little physical impact when tuning for timing closure. It has also been shown to have a more efficient power/delay trading ratio than dual V_t technique [90]. The work conducted in [90] fully explores the layout dependence of stress enhancement and provides stress-enhanced versions for cells in a 65nm industrial library (can be considered as "dual stress"). The characterization of cells into the library enables incorporating the use of mechanical stress into as early as synthesis stage. To the best of our knowledge, [90] is the only work so far that embodies techniques of dual V_t, gate sizing and mechanical stress together. However, gate sizing does not participate in leakage optimization in their algorithm. It is applied for the objective of iso-area.

In this work, we consider the following problem: *given the netlist of a combinational logic circuit with a timing constraint, a comprehensive leakage/performance driven cell library with options of dual V_t, dual sizing, and dual stress, choose the size, V_t, and stress level of each cell such that the circuit's leakage (or total) power is minimized while the timing constraint is met.* We re-evaluate the concept of leakage minimization with multiple techniques simultaneously (dual V_t, gate sizing, and mechanical stress in our discussion). More specifically, we propose to first balance the circuit paths as close to the timing constraint as possible (apply dual V_t in our case), then introduce the concept of *urgent paths* to guide the repeated procedure of: performance improvement (gate sizing and mechanical stress) to create new slacks at the cost of power overhead; and power optimization (dual V_t) to trade slacks for power saving. We validate this approach on 8 OpenCores benchmarks using the data from a 65nm industrial library. The results show that our approach can achieve an average of 13.5% more leakage saving than the standard sensitivity-based approach. The rest of the work is organized as follows. Section 4.1 reviews the components of power consumption in a CMOS circuit and the delay/power tradeoff property for each of the three techniques. The dual V_t combined with simultaneous gate sizing and mechanical stress is illustrated in Section 4.2 and experimental results are presented in Section 4.3. Finally Section 4.4 concludes.

4.1 Preliminaries

In this section, we first describe the power and delay models we use. Then we introduce the three techniques with emphasis on their difference in the power-delay tradeoffs. Small illustrative examples are included.

4.1.1 Power Consumption in CMOS Circuits

There are three major sources of power consumption in a digital CMOS gate [91] which are summarized in the following equation:

$$\begin{aligned} P_{avg} &= P_{dynamic} + P_{short-circuit} + P_{leakage} \\ &= \alpha C_L \cdot V_{dd}^2 \cdot f_{clk} + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} \end{aligned} \tag{4.1}$$

The first term represents the switching component of power, where C_L is the load capacitance, f_{clk} is the clock frequency, and α is the switching activity of the gate, which indicates the average number of times the gate makes a logic transition in one clock cycle ($0 \leq \alpha \leq 1$). The second term is due to the short circuit current I_{sc}, which occurs when both PMOS and NMOS transitors are turned on simultaneously and the current flows directly from the power line to the ground. Short circuit current can be kept less than 15% of dynamic power with careful design [92]. The last term is brought by leakage current $I_{leakage}$, which arises from various mechanisms, with subthreshold leakage as the major component. When technology scales down to deep submicron era, leakage power has contributed to substantial percentage of total power.

4.1.2 Delay Model

This work adopts a non-linear delay model based on the 2-D lookup tables provided by the 65nm technology library. Cell delay is typically measured from 50% input pin voltage to 50% output pin voltage. It is impacted by two factors: input pin transition delay and output capacitance of the cell. These two factors are used as the two indices of the lookup table for cell delay. Interpolation is applied for intermediate values. Output capacitance can be calculated as the sum of the capacitances of input pins the gate's output net connects to, while the input transition delay, typically measured from 10% to 50% input pin voltage, is by itself indexed from a separate 2-D lookup table. Parasitic capacitance of wires and wire delay is not considered at this phase since physical information is not yet attainable. In the following, we use inverters as an example to illustrate the concept and power-delay tradeoff of gate sizing, dual V_t and mechanical stress.

4.1.3 Gate Sizing, Dual V_t and Mechanical stress Techniques

Gate sizing adjusts the power-performance tradeoff by changing the (W/L) ratio of the gate. When sizing up a gate, its equivalent resistance becomes smaller, while its equivalent capacitances become larger. The overall effect is the reduction in gate delay and increase in leakage power. Meanwhile, since the dynamic power as well as the delay of a gate depend on the load capacitance it's driving, the gates in the fanin cone of the sizing gate will consume more dynamic power and have delay overhead.

Table 4.1 provides the input pin capacitances for different inverters under the 65nm library we use. Table 4.2 and Table 4.3 provides leakage power and cell delay information. The impact brought by gate sizing can be observed by making row-based comparisons among these tables. As indicated by Table 4.1 and 4.2, the input capacitance and the leakage power of a large inverter is much larger than that of the small inverter, both around three times, regardless of the V_t value. Meanwhile, the dynamic power of a cell will also go up with one or more of its fanout cells sized up. However, large inverter is fast. From Table 4.3, we can see that the speed-up of large inverter increases as the load capacitance becomes large.

Table 4.1: Comparison of input pin capacitances for different inverters. $V_{dd} = 1.2V$, temperature = 25^oC.

Gate	low V_t $/pf$	high V_t $/pf$
INV_{small}	0.003282	0.002893
INV_{large}	0.009204	0.008032

Table 4.2: Comparison of leakage power for different inverters. $V_{dd} = 1.2V$, temperature = 25^oC.

Gate	low V_t $/nW$	high V_t $/nW$
INV_{small}	0.253	0.013
INV_{large}	0.797	0.036

Table 4.3: Comparison of gate delay for different inverters. $V_{dd} = 1.2V$, temperature = 25^oC. Different number of fanout gates are considered, assuming all of them are of type INV_{small}. The input transition delay is assumes to be 0.24ns for simplicity.

Gate	#Fanouts	low V_t $/ns$	high V_t $/ns$
INV_{small}	1	0.0671	0.1037
	10	0.1302	0.1843
INV_{large}	1	0.0584	0.0915
	10	0.0890	0.1342

Example: Consider the 3-inverter circuit in Figure 4.1, where all three gates are low V_t inverters of type INV_{small}. By changing G_2 into type INV_{large}, the delay on path G_1 to G_2 will decrease by $0.0671 - 0.0584 = 0.0087ns$ (Table 4.3), the leakage of the circuit will increase by $0.797 - 0.253 = 0.544nW$ (Table Table 4.2), the input capacitance will increase by $0.009204 - 0.003282 = 0.005922pf$ (Table 4.1). The last change will increase the dynamic power of inverter G_1 as follows:

$$\begin{aligned} \Delta P_{dynamic} &= \frac{0.005922}{0.003282 + 0.003282} \cdot P_{dynamic,1} \\ &= 0.9 P_{dynamic,1} \end{aligned} \tag{4.2}$$

In general, the increase in dynamic power can be estimated by

$$\Delta P_{dynamic} = \frac{C'_{L,1} - C_{L,1}}{C_{L,1}} \cdot P_{dynamic,1} \tag{4.3}$$

where $C_{L,1} = C_{input,2} + C_{input,3}$

and $C'_{L,1}$ is the increased load capacitance after gate sizing.

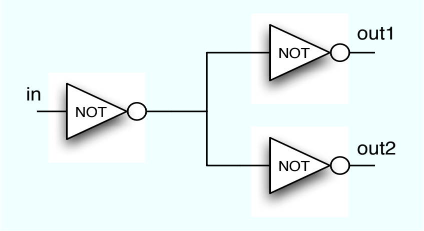

Figure 4.1: One inverter fans out to two inverters.

If the gate has low switching activity, the dynamic power increase can be trivial; If the gate is of high switching activity, however, the dynamic power increase can become numerically comparable to the leakage power increase.

Gate delay has a sub-linear dependency on threshold voltage while subthreshold leakage is exponentially dependent on threshold voltage. This makes threshold voltage a powerful optimization parameter. High V_t gates are generally slower, however consume less leakage power; while low V_t gates are faster, but consume more leakage power. From Table 4.3 and Table 4.2, we can see that by using high V_t, leakage can be reduced by around 20X while the delay increase is less than 2X.

Example: Combining gate sizing and dual-V_t techniques have been extensively researched. The advantage brought by the combination of the two techniques can be observed by comparing INV_{small} cell at low V_t ($0.253nW$ and $0.1302ns$) with INV_{large} cell at high V_t ($0.036nW$ and $0.1342ns$), both under the case of 10 fanouts. Although gate sizing and dual-V_t have opposite impacts on delay and leakage consumption, the huge leakage saving can be achieved without incurring much delay overhead, as indicated by the following estimation.

$$\frac{\Delta P_{leakage}}{P_{leakage}} = (0.253 - 0.036)/0.253 = 85.8\%$$
$$\frac{\Delta D}{D} = (0.1342 - 0.1302)/0.1302 = 3.0\%$$

Furthermore, when total power is concerned, the dynamic power increase caused by using INV_{large} can be estimated as in Equation 4.2. One can easily compare the leakage saving and the dynamic power overhead to evaluate the impact on total power. When the leakage dominates the total power, this will result in total power saving.

Mechanical stress is not able to adjust the gate delay as extensively as dual-V_t technique, but it is advantageous in that it provides a much more efficient power/delay tradeoff, and much finer tunability than gate sizing and dual V_t. We may evaluate the effectiveness of mechanical stress via the data in Table 4.4. More complete table for all gate types can be found in the previous work [90].

Example: Assuming the inverter has the same rise and fall time, and the same

Table 4.4: Drive current improvement and leakage power increase via mechanical stress. $V_{dd} = 1.2V$, temperature $= 25^oC$. In our approximation, the data is regarded to be universal to all inverter types in the library.

Gate	Percentage Drive Current Improvement by Stressing		Increase in LeakageCurrent after Stressing	
	NMOS	PMOS	NMOS	PMOS
INV	6%	13%	1.86X	3.88X

chance of 0 → 1 and 1 → 0 switching, we can convert the measurement into average delay reduction rate and leakage power increase rate by perform the following computation:

$$\frac{\Delta P_{leakage}}{P_{leakage}} = \frac{1}{2}(1.86 + 3.88) = 2.87$$

$$\frac{\Delta D}{D} = 1 - \frac{1}{2}(\frac{1}{1+6\%} + \frac{1}{1+13\%}) = 8.58\%$$

Dynamic power can also increase due to the increase in load capacitance. However, it happens after layout, which is beyond the control of synthesis stage, and is relatively trivial to consider.

The higher power/delay trading ratio can be taken advantage of for both power and delay optimization. If after the combined gate sizing and dual-V_t illustrated, mechanical stress is further applied, simultaneous leakage and delay reduction can be achieved.

$$P_{leakage}\_new = 0.036 \times (1 + 2.87) = 0.139 < 0.253$$

$$D\_new = 0.1342 \times (1 - 8.58\%) = 0.1227 < 0.1302$$

From these examples we see that dual V_t gives significant leakage reduction at the cost of delay increase, sizing up the cells and applying stress both help to speed up, and if we use them simultaneously, we are able to overcome the delay overhead from using high V_t cells. This motivates us to explore using gate sizing and mechanical stress as a vehicle to provide the delay slacks required for dual V_t technology.

4.2 Urgent Path Guided Power Optimization

In this section, we improve dual V_t technique by simultaneous gate sizing and mechanical stress. We introduce the concept of *urgent path*s, and propose an efficient yet more optimal approach than standard sensitivity-based method. Guided by *urgent path*s, gates are selected for gate sizing or mechanical stress based on not only its power/delay tradeoff, but also the effective slack [1] it contributes to low V_t gates in the circuit.

The *urgent path* for gate G_j (denoted as UP_j) is defined as the longest one among all paths that go through G_j. Obviously, the slack of the urgent path UP_j decides the effective slack of G_j. Consider a gate $G_i (i \neq j)$ on the path of UP_j (or G_i covers UP_j). Speeding up G_i brings slack to UP_j, which indicates that the effective slack of G_j can be increased and power savings can be traded in turn. It has to be noted that the urgent path of G_i (UP_i) can be different from UP_j, and G_i can cover multiple urgent paths, with UP_j as one of them. Obviously, UP_i is also covered by G_i itself.

[1] The slack of a gate is the minimum slack of every path going through it. To clarify, we call it effective slack in the work.

Figure 4.2 gives the overview of our flow.

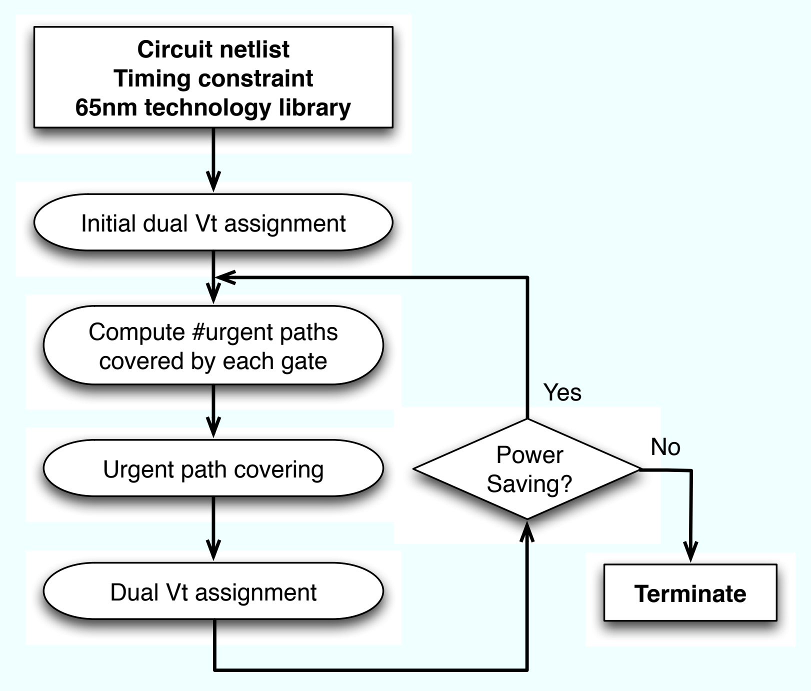

Figure 4.2: Overview of the proposed approach

The approach starts with all low V_t, small sized and non-stressed circuit. It first performs an initial dual V_t assignment. Then the approach enters a loop and performs the following three steps iteratively: 1. Compute the number of urgent paths each gate covers; 2. Select as few gates as possible to cover as many urgent paths as possible. Apply either gate upsizing or mechanical stress to the selected gates so that slacks are created on the urgent paths being covered; 3. Select as many low V_t gates as possible and change them into high V_t without timing violation. Power savings achieved in step 3 can normally exceed the power overhead in step 2 because the number of gates being changed in step 3 is normally much larger than that in step 2. The loop terminates when there is no more power savings achieved.

Performing initial dual V_t assignment provides a good starting point before entering the loop. After this phase, the urgent paths for each gate are exposed and approach timing constraint. By speeding up the gates that cover as many urgent paths as possible, which is done in step 2 of the loop, effective slacks can be created for many low V_t gates and large power savings can be achieved in return. The starting point in our approach is fundamentally different from most of the sensitivity-based works, where they either start from the fastest circuit, or the most power efficient one. In the former case, most paths have sufficient timing slack; while in the latter case, most paths violate timing. In either case, it gives subtle hints on which gates should be accelerated.

The dual V_t assignment in both the initial phase and in step 3 of the loop adopts levelization algorithm, where gates are changed level by level, in the order of the magnitude of power savings [93]. In the remaining content of this section, we will elaborate the first two steps performed in the loop.

4.2.1 Compute number of urgent paths each gate covers

The urgent path of G_i can be obtained by the longest path among those that enter G_i (denoted by UP_i_in) combined with (denoted by $\leadsto$) G_i itself and the longest path among those that leave G_i (denoted by UP_i_out):

$$UP_i = UP_i\_in \leadsto G_i \leadsto UP_i\_out$$

Define the *previous* gate of G_i as the gate connecting to the incoming net with the latest arrival time (denoted by $G_{prev(i)}$), and the *next* gate as the gate connecting to the outgoing net with the earliest request time (denoted by $G_{next(i)}$). Since UP_i_in

should always go through $G_{prev(i)}$ before entering G_i, and UP_i_out should always go through $G_{next(i)}$ after leaving G_i, the urgent path of G_i can be retrieved by the following recursion:

$$UP_i\_in = UP_{prev(i)}\_in \rightsquigarrow G_{prev(i)}$$

$$UP_i\_out = G_{next(i)} \rightsquigarrow UP_{next(i)}\_out$$

As a result, if we record the *previous* gate and *next* gate for each gate in the circuit by one forward and one backward traversal, the urgent path for each gate can be traced. This procedure is of the same complexity as performing static timing analysis.

Gate G_i not only covers the urgent path of itself, it can cover multiple urgent paths of other gates. We define the *urgent path set* of gate G_i as the set of urgent paths G_i covers, denoted by $\mathcal{UP}_i$. We compute the number of urgent paths G_i covers ($|\mathcal{UP}_i|$) by performing another round of forward and backward traversal. The procedure is based on the following key observation:

Lemma 1: if a gate G_i covers the urgent path of another gate G_j, UP_j should either overlap with UP_i_in or with UP_i_out, or both.

Proof Suppose G_j is topologically prior to G_i and G_i covers UP_j (see Figure 4.3). Assuming UP_j does not overlap with either UP_i_in or UP_i_out, it will contradict the definition of UP_j as by replacing the subpath P_j with UP_i_out, the new path is even longer and still goes through G_j. The similar reasoning can be applied to G_k which is topologically after G_i. UP_k must overlap with UP_i_in.$*$

Based on the observation above, we can divide the paths in the set of $\mathcal{UP}_i$ into two categories: those that overlaps with UP_i_in and those that overlaps with

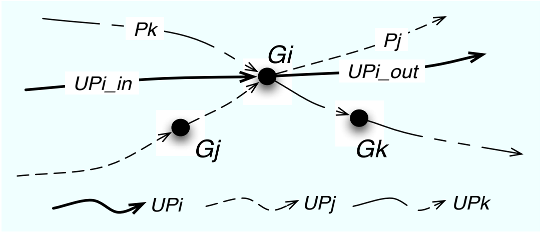

Figure 4.3: One gate covers the urgent paths of the other two gates.

UP_i_out. We denote the two sets of paths as $\mathcal{UP}_i\_in$ and $\mathcal{UP}_i\_out$, respectively. The only intersection of the two sets, if any, is UP_i.

Before we illustrate the procedure in Figure 4.4, two more lemmas are introduced below to show the correctness of the procedure.

Lemma 2: If G_i is the *next* gate of its fan-in gate G_{i_j}, $\mathcal{UP}_{i_j}\_out \subseteq \mathcal{UP}_i\_out$.

Proof Consider an urgent path $\tilde{UP} \in \mathcal{UP}_{i_j}\_out$, $\tilde{UP}$ overlaps with $UP_{i_j}\_out$ according to the definition of $\mathcal{UP}_{i_j}\_out$. Since G_i is the *next* gate of G_{i_j}, $UP_{i_j}\_out$ must contain UP_i_out as its subpath. As a result, $\tilde{UP}$ also overlaps with UP_i_out and according to the definition of $\mathcal{UP}_i\_out$, $\tilde{UP} \in \mathcal{UP}_i\_out$.$*$

Lemma 3: If G_i is the *previous* gate of its fan-out gate G_{i_k}, $\mathcal{UP}_{i_k}\_in \subseteq \mathcal{UP}_i\_in$.

Proof Similar to proof of *Lemma 2* and omitted here.

The procedure described in Figure 4.4 obtains the size of $|\mathcal{UP}_i|$ by computing $|\mathcal{UP}_i\_in|$ and $|\mathcal{UP}_i\_out|$ separately. Line 1-6 is the initialization stage. All gates, except those on PIs and POs, have the number of urgent paths they cover initialized to zero. In line 8-10, we examine each of the fan-in gates For a certain gate G_i. If G_i is the *next* gate of its fan-in gate G_{i_j}, Lemma 2 indicates that size of $\mathcal{UP}_{i_j}\_out$ will contribute to the size of $\mathcal{UP}_i\_out$ (line 10). Similarly, in the backward traversal described in line 14-16, if G_i is the *previous* gate of its fan-out gate G_{i_k}, size of

```
// Initialization
 1. for each gate $G_i$
 2.     $|\mathcal{UP}_i\_in| = 0$; $|\mathcal{UP}_i\_out| = 0$;
 3.     if all of $G_i$'s incoming nets are PIs
 4.         $|\mathcal{UP}_i\_out| = 1$;
 5.     else if all of $G_i$'s outgoing nets are POs
 6.         $|\mathcal{UP}_i\_in| = 1$;
// Forward traversal
 7. for each gate $G_i$ following topological order
 8.     for each fanin gate $G_{i_j}$ of $G_i$
 9.         if $next(G_{i_j}) == G_i$
10.             $|\mathcal{UP}_i\_out|$ += $|\mathcal{UP}_{i_j}\_out|$;
11.     if $prev(next(G_i))! = G_i$
12.         $|\mathcal{UP}_i\_out|$ += 1;
// Backward traversal
13. for each gate $G_i$ reversing topological order
14.     for each fanout gate $G_{i_k}$ of $G_i$
15.         if $prev(G_{i_k}) == G_i$
16.             $|\mathcal{UP}_i\_in|$ += $|\mathcal{UP}_{i_k}\_in|$;
17.     if $next(prev(G_i))! = G_i$
18.         $|\mathcal{UP}_i\_in|$ += 1;
// Combine
19. for each gate $G_i$ in the circuit
21.     $|\mathcal{UP}_i| = |\mathcal{UP}_i\_in| + |\mathcal{UP}_i\_out|$ - 1;
```

Figure 4.4: Compute #urgent paths each gate covers by forward and backward traveral.

$\mathcal{UP}_{i_k}\_in$ contributes to the size of $\mathcal{UP}_i\_in$ (line 16) based on Lemma 3. In line 11-12, the urgent path if G_i itself (UP_i) is added into $\mathcal{UP}_i\_out$ if it is not previously included in line 8-10. UP_i was not included only when G_i is not the *next* gate of its *previous* gate, therefore UP_i is not counted in line 10 when examining the *previous* gate. Similarly, line 17-18 include UP_i into $\mathcal{UP}_i\_in$ if it is not previously included in line 14-16. Line 20-21 combines the results of $|\mathcal{UP}_i\_in|$ and $|\mathcal{UP}_i\_out|$ into $|\mathcal{UP}_i|$ while eliminating the double count of UP_i.

If there are n gates in the circuit with m edges. The complexity of this step will be the same as circuit traveral, which is $O(n + m)$.

4.2.2 Urgent path covering

In the second step of the loop, we select as few gates as possible to cover as many urgent paths as possible. For each gate being selected, either gate upsizing or mechanical stress is applied to reduce the gate delay.

We use a heuristic approach to perform urgent path covering. The following metric is defined:

$$W_i = |\mathcal{UP}_i| \times max\{\frac{\delta D_{i_k}}{\delta P_{i_k}}|k = 1, 2\} \tag{4.4}$$

where i indicates different gate, k indicates two options of gate upsizing or mechanical stress. The larger W of a gate, the larger slack it brings and more urgent paths will benefit, while smaller power overhead is incurred. Both δD_{i_k} and δP_{i_k} $(k = 1, 2)$ are pre-estimated. Upsizing a gate does not necessarily reduce the path delay, because larger gate brings larger load to its fan-in gates and their delay will increase. If it results in timing violation, the option of gate upsizing is regarded as ineligible.

We find a set of gates as small as possible to cover urgent paths by iteratively selecting the gate with the largest W. After a gate is selected and changed, incremental timing analysis is performed. δD_{i_k} and δP_{i_k} $(k = 1, 2)$ for all the gates in the circuit are re-estimated because the input transition delay and/or output load capacitance can be different. We also need to update the number of urgent paths each gate covers, because some of them have been covered by the selected gate. The iteration stops when the number of urgent paths newly covered is smaller than a threshold. Figure 4.5 elaborates the update procedure.

```
 1. $|\mathcal{UP}_i\_out| = 0$; $|\mathcal{UP}_i\_in| = 0$;

update_fanin_urgent_path($G_i$)
 2. if $G_i \in PI$
 3.       return;
 4. for each fanin gate $G_{i_j}$ of $G_i$
 5.       if $next(G_{i_j}) == G_i$
 6.           $|\mathcal{UP}_{i_j}\_out| = 0$;
 7.           update_fanin_urgent_path($G_{i_j}$);

update_fanout_urgent_path($G_i$)
 8. if $G_i \in PO$
 9.       return;
10. for each fanout gate $G_{i_k}$ of $G_i$
11.       if $prev(G_{i_k}) == G_i$
12.           $|\mathcal{UP}_{i_k}\_in| = 0$;
13.           update_fanout_urgent_path($G_{i_k}$);

Repeat line 7-21 in Figure 4.4
```

Figure 4.5: Update #urgent paths each gate covers

When gate G_i is selected, all the urgent paths in the set of $\mathcal{UP}_i$ are covered. Line 1 therefore updates the size of $\mathcal{UP}_i$ to zero. In line 2-7, transitive fan-ins of G_i are examined. According to Lemma 2, $\mathcal{UP}_{i_j}\_out \subseteq \mathcal{UP}_i\_out$ and therefore $|\mathcal{UP}_{i_j}\_out| \leq |\mathcal{UP}_i\_out|$ if G_i is the *next* gate of its fan-in gate G_{i_j}. Since $|\mathcal{UP}_i|$ is updated to zero in line 1, $|\mathcal{UP}_{i_j}\_out| \leq |\mathcal{UP}_i\_out| \leq |\mathcal{UP}_i| = 0$ (line 6). With the help of *next* gate, all the transitive fan-in gates getting affected can be recursively backtraced. Similarly, line 8-13 updates $\mathcal{UP}_{i_k}\_in$ to zero if transitive fan-out gate G_{i_k} is the transitive *previous* gate of G_i. The update in $|\mathcal{UP}\_out|$ of the transitive fan-in gates of G_i will also affect $|\mathcal{UP}\_out|$ of other gates. We perform the same forward traversal from Figure 4.4 to update $|\mathcal{UP}\_out|$ of un-updated gates. Similarly, the same backward traversal from Figure 4.4 is applied to update $|\mathcal{UP}\_in|$ due to the change in $|\mathcal{UP}\_in|$ of the transitive fan-out gates of G_i.

If we evaluate the complexity of this step, selecting the gate with largest weight takes O(n) complexity. Re-estimating δD_{i_k} and δP_{i_k} takes another O(n) complexity. Performing the update in Figure 4.5 won't be of higher complexity than four passes of circuit traversal, which is bounded by $O(m + n)$. Incremental timing analysis can also be bounded by $O(m + n)$. So the total complexity of selecting one gate is $O(m + n)$. Meanwhile each gate in the circuit won't be selected into urgent path covering for more than twice because there are only options of either gate sizing or mechanical stress. The accumulative complexity of urgent path covering over all loops can therefore be bounded by $O(n(m + n))$.

The complexity of performing dual V_t assignment can vary by different algorithms. Suppose it is of $O(c(n, m))$ accumulative complexity over all loops, the overall complexity of our approach can therefore be evaluated as $O(n(m + n) + n(m + n) + c(n, m)) = O(nm + c(n, m))$.

4.3 Experimental Results

We use 9 different types of cells from the same 65nm low-power dual-V_t industrial library as in [90], where each cell has both low V_t and high V_t implementations. The library also provides implementations with larger driving strength for each type of cells. These implementations characterize the behavior of gate upsizing, which has increased leakage but reduced delay. In our current work, only one larger version of each cell type is included for ease of analysis. However, our approach can be easily extended to more general cases, and given more options, better results can be expected. The behavior of mechanical stress is characterized by the data provided in [90]. So the library used in our work can be regarded as the one containing 9 cell types, each with three options: low/high V_t, small/large size, and unstressed/stressed, which are eight combinations in total. The switching activity and dynamic power is calculated by propagating the user-specified toggle rate and static probability at the primary inputs. In this work, we specify static probability as 50% and toggle rate as 10%.

We apply our approach on 8 benchmarks from OpenCores [115]. The number of cells in the design is listed in Table 4.5, ranging from a few thousands to 38 thousands. We obtain the netlist of these designs using *Synopsys Design Compiler*. The ambient temperature is assumed to be 25^oC, and supply voltage is assumed to be $1.2V$.

Table 4.5: Number of cells for each benchmark circuit after synthesis.

Benchmark	des3	tv80s	mc_top	systemcaes
#cells	3758	6991	8140	10498
Benchmark	ac97	pci_bridge32	aes_cipher_top	sha512
#cells	11362	20043	21249	38002

As mechanical stress is the newly introduced optimization technique, our first set of experiments is designed to show the effectiveness of mechanical stress. Table 4.6 lists different leakage and dynamic power consumption under the following three cases: Column 2-4 in the first tabular shows the results when only dual V_t (*D*) is applied based on levelization algorithm [93]; Column 5-9 in the first tabular shows the results when combined dual V_t and gate sizing (*D+S*) is applied; The second tabular gives the results when mechanical stress is also incorporated (*D+S+Str*). We apply our urgent path guided approach to both *D+S* and *D+S+Str* cases. In column 5-7 of the first tabular, the leakage, dynamic and total power savings/overhead achieved by *D+S* are compared against dual V_t only; while the power savings in the second tabular achieved by *D+S+Str* are the further savings compared with *D+S*. On average, leakage power is reduced by 7.31% when gate sizing is combined, and is further reduced by 9.77% when mechanical stress is added. The straightforward reason is that gate upsizing and mechanical stress helps create timing slacks in the circuit, enabling more low V_t cells to be changed into high V_t. This can be observed from column 4, 8 of the first tabular, and column 5 of the second tabular, where the number of high V_t cells grows larger and larger. Mechanical stress is sometimes more efficient in trading power for speed than gate sizing, which partially contributes to the further leakage saving. It is also interesting to notice that the number of large cells also increases after mechanical stress is introduced (column 9 of the first and column 6 of the second). It is because the slack created by mechanical stress helps offset the delay increase in the fan-in cells brought by gate upsizing, which would otherwise result in timing violation. Gate sizing thereby can be more thoroughly exploited for potentially more power savings. To summarize, mechanical stress not only functions as an alternative for gate sizing, it is also a supplement to apply gate sizing under tight timing budget. The average power consumption saving is much smaller, with 1.57% by *D+S* and 2.79% more by *D+S+Str*. It is due to the slight increase in dynamic power (around 0.86%) caused by gate sizing, and the fact that dynamic power is still relatively larger than leakage power.

In our second set of experiments, we investigate the crucial impact of employing urgent paths into the power optimization procedure. We compare our approach with the one without the guidance of urgent paths simply by defining the metric in section 4.2.2 as $W_i = max\{\frac{\delta D_{i_k}}{\delta P_{i_k}}|k = 1, 2\}$ instead. The results are reported in Table 4.7. For ease of comparison, we still list the results in the second tabular of Table 4.6 into the second one of Table 4.7, though the absolute power consumption is the same as in Table 4.6 after conversion. In all but one circuit, either the number of large cells or the number of stressed cells is much smaller if urgent path is not employed. It is because without the guidance of urgent paths, the cells selected in step 2 of the loop in our algorithm may reside in paths not under the most tight timing budget. Changing these cells into large or stressed version won't be able to contribute effective slack to the low V_t cells in the circuit. Therefore the low V_t cells won't be eligible to be changed into high V_t and power consumption cannot be saved or even retrieved. As a result, rather than keep on changing cells into large or stressed ones and incurring more power overhead, the procedure would stop at an early stage. In *ac97* however, there are more large cells and stressed cells when urgent paths is not employed, but the number of high V_t cells is still smaller. This on the other hand also indicates that the location of large and stressed cells matters more than their quantity. Figure 4.6 compares the different number of high V_t cells we can afford to have in the circuit when every ten more cells are upsized or stressed.

We take *des3* and *pci_bridge32* as examples. It is obvious to see that more high V_t cells can be afforded at each sample point and the number also grows faster if guided by urgent paths. On average, the urgent path guided approach consumes 12% less leakage power and 3.49% less total power.

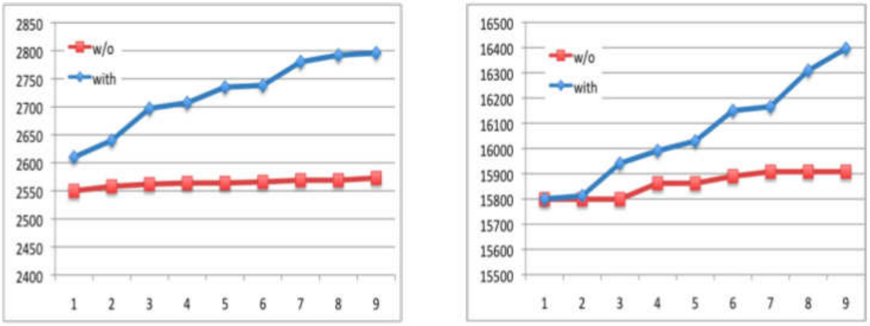

Figure 4.6: X axis: sample points when every ten more cells are upsized or stressed. Y axis: number of high V_t cells in the circuit. Left: des3. Right: pci_bridge32.

Last but not the least, we extend the sensitivity-based approach in [81] to deal with combined dual V_t with gate sizing and mechanical stress. The approach starts from the all high V_t and small sized circuit, which corresponds to minimal dynamic and leakage power but maximal delay. In each iteration, the approach finds among all the cells with negative slacks, the option of which with the largest sensitivity. It modifies the cell by either upsizing, reducing V_t or applying mechanical stress. Then it updates the circuit using static timing analysis. The iteration terminates when it achieves timing closure. There can still exist slacks at the end of the procedure. To make fair comparison, we apply levelization in the end to balance the circuit and obtain more power savings. The results are reported in Table 4.8. Again the results of urgent path based approach are the same as those in Table 4.6 and Table 4.7.

Similar to what happened in our second set of experiments, sensitivity-based approach in [81] select cells purely based on power/delay tradeoff to make up timing. Without considering the actual impact it can bring to the circuit by changing a cell, the effectiveness of sensitivity-based approach is always limited. Compared to sensitivity-based approach, our approach results in 13.5% less leakage power consumption and 5% less total power consumption.

4.4 Summary

This work is the first work that improves traditional dual V_t technique by incorporating gate sizing and mechanical stress for power optimization. Mechanical stress not only provides a more power efficient alternative to gate sizing for delay reduction, it also facilitates gate sizing when the circuit is under tight timing budget. The work also proposes a novel urgent path guided approach to help select cells to create slacks. The experimental results show that the urgent path guided approach enables more cells to be changed to high V_t and thus consumes less leakage and total power than its purely sensitivity-based counterparts.

Table 4.6: Power consumption comparison among dual V_t only (D), dual V_t combined with gate sizing (D+S), combined dual V_t and gate sizing further incorporating mechanical stress (D+S+Str).

Benchmark	*D*			*D+S*				
	leakage (nW)	dynamic (nW)	#high	leakage saving	dynamic overhead	total saving	#high	#large
des3	329.33	517.68	2544	6.39%	0.25%	2.34%	2633	32
tv80s	212.05	264.18	6199	1.69%	0.09%	0.70%	6217	41
mc_top	250.13	643.47	6514	6.05%	0.00%	1.69%	6576	189
systemcaes	305.59	2713.9	9218	2.19%	0.56%	-0.28%	9273	203
ac97	862.98	2770.6	6226	32.2%	0.89%	6.97%	7367	101
pci_bridge32	507.48	943.93	15799	1.92%	0.10%	0.61%	15851	130
aes_cipher_top	1427.5	3234.2	16409	7.62%	2.55%	0.56%	17048	899
sha512	1920.8	1894.7	30328	0.41%	0.44%	-0.02%	30380	192
average				7.31%	0.61%	1.57%		

Benchmark	*D+S+Str*					
	leakage saving	dynamic overhead	total saving	#high	#large	#stressed
des3	20.2%	0.80%	7.03%	3085	178	1072
tv80s	0.00%	0.00%	0.00%	6217	38	2
mc_top	6.44%	1.01%	0.98%	6675	230	80
systemcaes	0.80%	0.58%	-0.44%	9360	373	167
ac97	2.36%	0.13%	0.30%	7429	103	17
pci_bridge32	38.4%	0.00%	13.3%	16649	191	59
aes_cipher_top	5.32%	2.37%	-0.18%	17772	1513	1247
sha512	4.64%	1.92%	1.37%	30970	640	1230
average	9.77%	0.84%	2.79%			

Table 4.7: Power consumption comparison between approaches with and without the guidance of urgent paths

Benchmark	Without urgent path				
	leakage (nW)	dynamic (nW)	#high	#large	#stressed
des3	322.62	521.63	2601	53	70
tv80s	212.09	264.18	6199	0	1
mc_top	240.97	646.45	6564	35	37
systemcaes	305.62	2713.9	9218	1	0
ac97	597.68	2871.7	7420	278	242
pci_bridge32	485.30	943.97	15908	35	41
aes_cipher_top	1427.5	3234.2	16409	0	1
sha512	1920.8	1894.7	30328	1	0

Benchmark	With urgent path					
	leakage reduction	dynamic overhead	total reduction	#high	#large	#stressed
des3	23.8%	0.29%	8.90%	3036	99	701
tv80s	1.71%	0.08%	0.72%	6217	38	2
mc_top	8.76%	0.55%	1.98%	6675	230	80
systemcaes	2.98%	1.14%	-0.72%	9360	373	167
ac97	4.40%	-2.54%	2.86%	7429	103	17
pci_bridge32	36.8%	0.03%	12.5%	16649	191	59
aes_cipher_top	12.5%	4.98%	0.38%	17772	1513	1247
sha512	5.03%	2.37%	1.36%	30970	640	1230
average	12.00%	0.86%	3.49%			

Table 4.8: Power consumption comparison between sensitivity-based approach and urgent path guided approach

Benchmark	Sensitivity based				
	leakage (nW)	dynamic (nW)	#high	#large	#stressed
des3	293.88	525.20	2724	97	67
tv80s	241.01	266.52	6257	159	16
mc_top	228.42	644.66	6636	82	6
systemcaes	309.28	2734.2	9264	120	11
ac97	781.22	2932.0	6704	499	33
pci_bridge32	408.77	944.02	16222	142	2
aes_cipher_top	1324.8	3459.7	17340	1565	129
sha512	2093.7	1950.6	30976	1457	172

Benchmark	Urgent path guided					
	leakage reduction	dynamic overhead	total reduction	#high	#large	#stressed
des3	16.3%	-0.40%	6.10%	3036	99	701
tv80s	13.5%	-0.80%	6.83%	6217	38	2
mc_top	3.74%	0.83%	0.37%	6675	230	80
systemcaes	4.13%	0.39%	0.07%	9360	373	167
ac97	26.9%	-4.55%	9.24%	7429	103	17
pci_bridge32	25.0%	0.03%	7.53%	16649	191	59
aes_cipher_top	5.75%	-1.86%	2.94%	17772	1513	1247
sha512	12.9%	-0.56%	6.94%	30970	640	1230
average	13.5%	-0.87%	5.00%			

Chapter 5

Enhancing Dual-V_t Design by Considering On-Chip Temperature Variation

As technology scales down to the deep sub-micron domain, chip power dissipation and power density are increasing rapidly. As a result, the heat accumulation on the chip causes the rise of on-chip temperature and its spatial variation known as hot spots. Such on-chip temperature variations are nonuniform across the chip and on high-performance microprocessor chip hot spots can be 50^oC hotter than other parts of the die. To reduce the peak on-chip temperature, most active research are on dynamic thermal management (DTM) at microarchitecture level [94, 101, 106, 107]; on resource allocation and binding at behavioral synthesis [113]; and on cell placement and floorplanning at physical design level [96, 97, 98, 110].

Temperature has a strong impact on leakage current, an exponential dependency according to the Berkeley BSIM model. Several work have been reported on the study of temperature aware leakage minimization at high level design [95, 99, 114], which we will review in next section. The goal of this work is to explore how to incorportate on-chip temperature variation into the low level dual V_t design methodology to enhance the leakage reduction.

Dual-V_t technique is a popular leakage reduction methodology in logic and physical synthesis. It is based on the fact that cells implemented by low V_t has short delay but large leakage comparing to the same cell implemented by high V_t. Therefore, it assigns high V_t to cells off the timing critical paths to reduce leakage while using low V_t cells on timing critical paths for timing closure. Current dual-V_t assignment algorithms use the delay and leakage data provided by the cell library and treat temperature as a pre-assumed constant [100, 111].

Ignoring the on-chip temperature variation will not fully explore the potential of dual-V_t methodology. For example, in the TSMC's 65nm low power dual-V_t library, the 2-input NAND cell's leakage is $0.016nW$ and $0.205nW$ at 25^oC and 125^oC, respectively when implemented using high V_t. However, these numbers change to $0.074nW$ and $3.973nW$ when the same cell is implemented using low V_t. Therefore, when we raise the cell's V_t from low to high, the leakage reduction will be $0.189nW$ if the cell is at a 25^oC region, while the saving will be as much as $3.899nW$ for the same cell at a 125^oC region. This implies that with temperature information, dual-V_t design can be more effective.

More specifically, treating temperature as a constant during dual-V_t design has the following drawbacks: (1) *Non-minimized leakage.* Without considering temperature explicitly, cells in high temperature region cannot be identified for high V_t to achieve maximal leakage saving. (2) *Hot spots and thermal runaway.* When temperature is higher than the pre-assumed constant, leakage will be larger than expected which will further raises temperature, creating hot spots and causing thermal runaway. (3) *Over-designed cooling system.* Using pre-assumed constant values such as the highest operating temperature will guarantee timing and avoid themal runaway. However, it will (significantly) over-estimate leakage and thus the cooling cost.

Fig. 5.1 on the left is the temperature profile on *aes_core* circuit from the Opencores benchmark [115] with V_t assignment guided by a state-of-the-art dual-V_t assignment algorithm [111] currently used in industrial CAD tools. *aes_core* circuit is a cryptocore which implements Advanced Encryption Standard (AES). We see that the peak temperature is 70.37^oC and there are 1309 cells (about 10%) have temperature 55.37^oC or higher.

We modify this design by our temperature-aware enhancement approach and we are able to reduce the peak temperature to 68.54^oC. More importantly, there are only 863 cells with 55.37^oC or higher and the total leakage saving in combinational part is about 17.27% of the original design. This can be seen clearly from the smaller high temperature area in Fig. 5.1 on the right.

In this work, we study how to incorporate on-chip temperature variation into the leakage model to enhance the leakage reduction of the dual-V_t design in the following problem: *given a gate level circuit implemented by a dual-V_t cell library for leakage minimization, how to improve the design in terms of leakage reduction while considering the temperature-leakage dependency and the on-chip temperature variation.*

5.1 Challenges of Temperature-Aware Dual-V_t Design

The first challenge is how to obtain the temperature information for each cell, which depends on many factors such as the location of the cell, the dynamic and

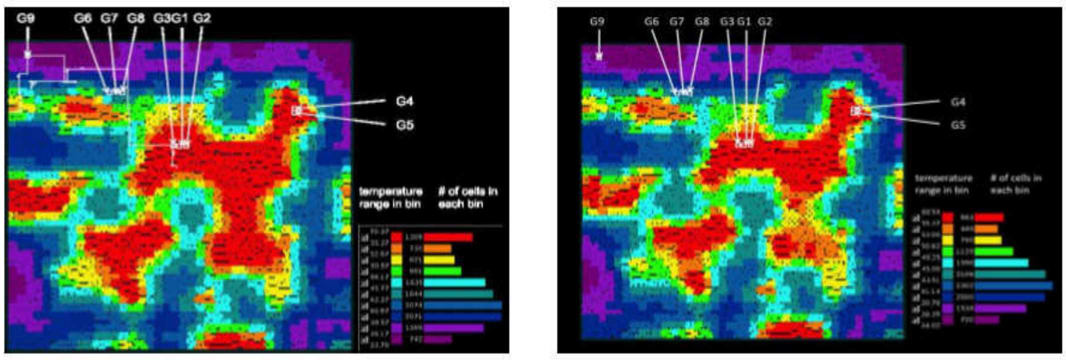

Figure 5.1: Temperature profile of the AES design using the dual-V_t algorithm proposed in [111] (left) and after our temperature-aware enhancement approach (right).

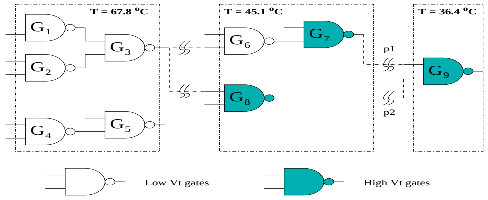

Figure 5.2: Nine cells extracted from AES design using the dual-V_t algorithm proposed in [111].

leakage power of the circuit, and the cooling system. Such temperature information is necessary for the evaluation of circuit leakage for any temperature-leakage models [109]. We perform a global placement with the given initial dual-V_t assignment. This gives us the physical location of each cell, based on which we can partition the circuits and apply the commonly used 3-D mesh temperature model [96] to obtain the steady-state temperature distribution.

Second, there are a few technical difficulties to implement our basic idea: assigning more cells with high V_t in hot regions in order to reduce leakage. We use an example of nine cells extracted from the above *aes_core* benchmark circuit (Fig. 5.2) to illustrate this.

1) We prefer to change more cells in hot region to high V_t to reduce both

leakage and temperature. However, changing multiple cells (e.g. G_1 and G_3, or G_4 and G_5) may create long delay overhead. But changing G_1 and G_2 together will create the same delay penalty as replacing G_3 alone while giving twice the leakage saving.

2) When we lower V_t on cells to make the timing, the selection of these cells are critical. For example, using low V_t on G_7 and G_8 can make up the timing for paths p_1 and p_2, respectively; while a low V_t implementation of G_9 can make up the timing for both paths. Clearly the second option is preferred for its smaller leakage increase.

3) We follow the current industry design standard to use worst-case delay as the cell delay, despite the known temperature-delay dependency. That is, changing the V_t of the same cell in different temperature has the same impact on circuit timing. However, changing G_6 to high V_t does not save as much leakage as changing G_3 and changing G_7 to low V_t will incur more leakage overhead than changing G_9. This makes the problem more complicated.

In summary, the key technical challenge in our approach is *how to select multiple low V_t cells and change them to high V_t with minimal impact on circuit timing and maximal leakage saving; and how to select high V_t cells and change them to low V_t for timing closure with minimal leakage overhead.* We propose effective heurisitcs solutions to solve the problem. Experimental results show that we are able to achieve an average of 11.2% more leakage reduction and 39% reduction of cells in hot regions comparing to the dual-V_t assignment approach [111].

5.2 Related Work

With the increasing magnitude of leakage power, many leakage reduction techniques have been proposed for low power designs. When the circuits are in standby mode, techniques such as power-gating, input vector control and forward body-biasing etc. have been proposed to reduce the static leakage power. A survey of these techniques can be found in [105].

Dual-V_t is one of the most effective methods to reduce both static and dynamic leakage power in the circuit. It was first proposed by Wei et al. [112]. In this approach, all the cells are initially assigned low V_t and a static timing analysis is performed to obtain the timing slack for each cell. Then it assigns high-V_t to cells that have sufficient timing slack following the topological order from primary inputs to primary outputs. Wang et al. [111] proposed a more efficient graph-partitioning-based dual-V_t assignment algorithm. They partition the circuit graph using *Max-Cut* algorithm. Cells in the same cut set can be considered for high-V_t assignment together without affecting the timing slack of each other. Srinivastava et al. consider dual-V_t, voltage scaling and gate sizing simultaneously due to their common dependency on delay/power trade-offs [108]. They calculate the leakage sensitivity of each cell and use a greedy approach to assign high V_t; if there is not enough timing slack, gate sizing is applied. Most recently, a dual-V_t assignment algorithm for sequential element in the circuit is proposed in [100].

Temperature, another important design consideration in deep submicron technology, has been extensively studied in many design phases. In physical design, [110] proposed a compact thermal model to calculate the temperature profile efficiently on the die. Based on the temperature distributioin, they proposed a placement algorithm such that the hot spots can be reduced without compromising area and wire length. [96] proposed a multigrid-like heuristic that simplifies the thermal equation at each level of partitioning and makes it possible to incorporate temperature directly as placement constraints. Temperature-aware floorplanning and clock tree design are also studied in [97, 98] and [103] respectively. In behavior synthesis, Mukherjee et al. have discussed temperature aware resouce allocation and binding for thermal management [104]. Yu et al. propose to balance different types of resource under area constraint to reduce peak temperature [113]. At microarchitecture and system level, Brooks and Martonosi define the major components and explore several techniques for DTM on microprocessor [94]. Skadron et al. describe the design of HotSpot thermal model and several DTM techniques [106]. Srinivasan and Adve propose a predictive DTM algorithm for multimedia applications where dynamic voltage scaling and architecture adaptation have been used [107].

Temperature's impact on leakage has also been studied in several works. Ku et al. propose a thermal-aware cache powerdown technique that lowers the temperature and thus leakage of the active parts [101]. He et al. use a system-level leakage model for leakage reduction on caches and discuss the interdependency between leakage and temperature [99]. Mukherjee et al. control peak temperature to reduce leakge during resource binding [104]. Yuan et al. develop an online algorithm to turn on and off the processor based on temperature and workload to reduce leakage [114].

We aim to use on-chip temperature variation to guide further leakage reduction by the dual-V_t design method. This is different from the above approaches because at a low design level (i.e., gate level) where leakage data is provided in the cell library, leakage and its reduction can be modeled more precisely. [95] is the only work we know that connects temperature and dual-V_t assignmnet. However, they do not consider temperature variation and the inversed temperature-delay dependency in their circuit is not observed in our 65nm cell library.

5.3 Leakage and Temperature Models

In this section, we describe the power and temperature models we used for temperature aware Vt assignment. These models have been validated by SPICE simulation and widely adopted in other literatures [96, 99, 109, 110].

5.3.1 Leakage and Power Model

Power consumption in CMOS circuits consists of dynamic and leakage power. Dynamic power depends on supply voltage, switching activity, clock frequency and load capacitance, but is insensitive to temperature. In our experiment, dynamic power is obtained directly from industry CAD tool.

Leakage power, on the other hand, has a strong dependency on temperature and can be modeled at many design levels [96, 99, 100, 109, 114]. Our proposed method is independent of the temperature-leakage model. We adopt the following quadratic model proposed in [109] because of its accuracy (comparing to SPICE simulation) and computational efficiency:

$$I_{leak}(T) = I_0(c_1(T - T_0)^2 + c_2(T - T_0) + 1), \tag{5.1}$$

where I_0 is the leakage current at the nominal temperature T_0, c_1 and c_2 are technology dependent constants, which vary from cell to cell and are normally obtained empirically.

The TSMC 65nm dual-Vt library has cell information characterized at different temperature corners. We first find the leakage power for each cell under minimum, nominal and maximum temperatures. We then perform curve fitting to compute the coefficients c_1 and c_2. This is performed twice for each cell, one at low V_t and one at high V_t. For instance, a INVD1 cell with drive strength *1X* in the library has $c_1 = 6.01 \times 10^{-3}$, $c_2 = -5.02 \times 10^{-2}$ at low V_t; and $c_1 = 3.54 \times 10^{-3}$, $c_2 = -2.58 \times 10^{-2}$ at high V_t.

5.3.2 Temperature Model

Temperature on VLSI chip exhibits both temporal and spatial characteristics. Because the time constant of on-chip heat conduction is usually several orders of magnitude larger than the clock periods in modern VLSI designs, the transient change in power does not have much impact on temperature distribution and hence we only consider the steady-state temperature profile.

Fig. 5.3 is a typical 3-D heat conduction model for VLSI chip [96]. The chip is partitioned into many thermal blocks. The heat diffusion between each thermal block and from thermal blocks to the ambient can be described by different thermal conductance value. Compact thermal model [110] shown in (5.2) can be deduced if

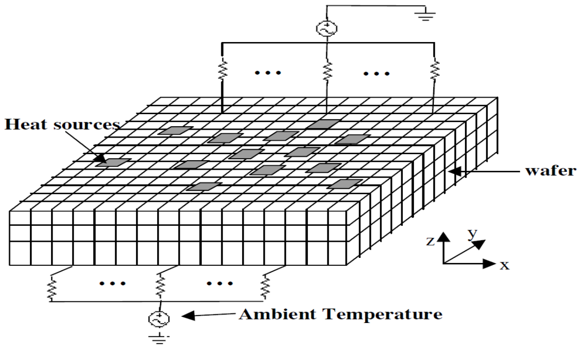

Figure 5.3: 3-D mesh model for heat conduction on chip [96].

we are only interested in the thermal blocks at the top layer with correspond to the thermal distribution on the silicon surface. The heat in each of these top blocks are mainly caused by the power dissipation of the standard cells in the block.

$$\begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m1} & G_{m2} & \cdots & G_{mm} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_m \end{bmatrix}, \tag{5.2}$$

where G is the thermal conductance matrix, and vector T and P are the temperature and power dissipation in each thermal block respectively. In our work, however, since the leakage power is associated with a given V_t assignment and is affected by temperature as in (5.1), the power vector $\vec{P}$ is a function of temperature $\vec{T}$ and $\vec{V_t}$. Equation (5.2) will be in the following format:

$$G \times \vec{T} = \vec{P}(\vec{T}, \vec{V_t}) \tag{5.3}$$

Theoretically, we should be able to compute the steady-state temperature under a certain V_t assignment. However, because of the quadratic dependency of $\vec{P}$

on $\vec{T}$, a closed form of the solution to this nonlinear equation (5.3) is impractical and unnecessary for large scale circuit design. Instead, we apply an iterative approach which keeps on updating $\vec{P}$ based on (5.1) and updating $\vec{T}$ based on (5.2) until the temperature converges. The convergence is within a few iterations under our experiment settings.

5.4 Temperature-Aware Dual-V_t Design

5.4.1 Overview of the Approach

We propose to enhance a dual-V_t assignment by assigning more cells with high V_t in hot regions in three phases. This is motivated by the observation that it is advantageous to distribute slack to cells in high temperature region where the change of low V_t to high V_t will give more leakage saving.

First, we do circuit placement based on the V_t assignment that we want to improve. Then we partition the circuit and compute each region's steady-state temperature as described in the previous section.

In the second phase, we repetitively adjust the initial V_t assignment by changing selective low V_t cells in high temperature region to high V_t (for leakage reduction), and changing high V_t cells in low temperature regions to low V_t (for timing closure). Steady-state temperature corresponding to the current V_t assignment is computed based on the iterative approach indicated by Section 5.3.2. This procedure stops when there is little leakage saving or the temperature variation among different regions across the chip becomes small. We will elaborate this phase in the rest of this section.

When we change high V_t cells to low V_t for timing closure, cells that are on the same path as these new low V_t cells may have increased slack. In the last phase, we take advantage of these positive slacks for further leakage reduction by assigning more cells high V_t. At the end, another round of thermal analysis will be performed to obtain the final leakage and temperature information.

5.4.2 Terminologies

If there is a path that goes from cell c_i to cell c_j, we say c_i is a transitive fanin of c_j and c_j is a transitive fanout of c_i. For cell c_i, define TI_i and TO_i as the *transitive fanin set* and *transitive fanout set* that consists of all c_i's transitive fanins and fanouts, respectively. We can construct TI_i and TO_i using the following formula:

$$\begin{aligned} TI_i &= \cup_{c_j}(TI_j \cup \{c_j\}) &(5.4)\\ TO_i &= \cup_{c_j}(TI_j \cup \{c_j\}), &(5.5)\end{aligned}$$

where $\cup$ is taken over all fanin c_j of c_i in (5.4) following a topological order $\{c_1, c_2, \cdots, c_n\}$ and the $\cup$ in (5.5) is taken on all fanout c_j of c_i in the reverse order $\{c_n, \cdots, c_2, c_1\}$.

The union of cell c_i's *transitive fanin set* and *transitive fanout set* is defined as Cov_i as follows:

$$Cov_i = \{c_j | c_j \in TI_i \cup TO_i\} \tag{5.6}$$

Cov_i is actually the set of cells whose timing can be affected by c_i.

Let IN_i and OUT_i be the number of paths going into c_i from primary inputs and going out to primary outputs from c_i, respectively. Given a topological order $\{c_1, c_2, \cdots, c_n\}$, they can be calculated by:

$$\begin{aligned} IN_i &= PI_i + \sum_j IN_j && (5.7) \\ OUT_i &= PO_i + \sum_j OUT_j, && (5.8) \end{aligned}$$

where PI_i is the number of primary inputs in c_i's fanin and PO_i is the number of primary outputs in c_i's fanout, the $\sum$ is taken over all fanin c_j of c_i in (5.7) in the order of $\{c_1, c_2, \cdots, c_n\}$ and the $\sum$ in (5.8) is taken in the reverse order $\{c_n, \cdots, c_1\}$ for all fanout c_j of c_i. Clearly we have

Lemma 1. $P_i = IN_i \times OUT_i$ gives the number of distinct paths from primary input to primary output that pass through cell c_i.

Lemma 2. For a circuit with n cells, the complexity to compute TI_i, TO_i, P_i and $Cov(c)$ is $O(n^2)$.

Finally, we define the following metric to measure the impact of changing a low V_t cell c_i to high V_t:

$$w_i(T) = \frac{\Delta L_i(T)}{\Delta D_i \times P_i} = \frac{I_{leak}^{V_t^{low}}(T) - I_{leak}^{V_t^{high}}(T)}{(D_{V_t^{high}} - D_{V_t^{low}}) \times P_i}, \tag{5.9}$$

where $\Delta L_i(T)$ and ΔD_i are the leakage saving at temperature T and worst-case delay increment, respectively, by raising cell i to high V_t at a region with steady-state temperature T. $\Delta D_i \times P_i$ gives a tight upper bound on the total delay impact across the entire circuit. This is because that in the worst case when none of the P_i

paths overlap, we need to find at least one cell on each path and change its V_t from high to low to make up the extra delay ΔD_i.

5.4.3 Raise V_t in Hot Regions

Fig. 5.4 depicts our heuristics on the selection of a set of low V_t cells for V_t increase. In line 3, we need to compute the leakage saving $\Delta L_i(T)$ for each cell with low V_t at the current steady-state temperature T. The use of metric $w_i(T)$ encourages the selection of cells with large leakage saving and small total delay overhead. We sort the cells in line 4 to facilitate such selection. The complexity is $O(|R| \log |R|)$, where $|R|$ is the number of cells in the hot region R picked in line 1.

As we have noticed in the introduction, we will select a set of low V_t cells and raise them to high V_t simultaneously to accelerate the process. We use the concept of maximal independent set (MIS) for the selection. Lines 5-7 illustrate a heuristic to construct an MIS, not necessarily an MIS with the maximal size (which is an NP-complete problem). Because each cell in R will be marked only once and we can take advantage of the pre-computed TI_i and TO_i, the complexity of this part is $O(n)$. Therefore, the complexity of the heuristics in Fig. 5.4 is bounded by $O(n \log n)$.

MIS is indeed an extension of the concept behind the levelization dual-V_t assignment algorithm [111] because cells at the same level belong to one independent set, but not necessarily maximal. Of course, one can use more sophisicated algorithm to build the MIS, which trades the complexity and performance and is out of the scope of this work.

The construction of MIS ensures that when we change multiple cells to high V_t, the delay penalty will not be additive to make timing closure easier.

Lemma 3. When we change the cells in MIS to high V_t, the delay on any path (or cell) is bounded by $\max\{\Delta D_i\}$ for all $c_i \in MIS$.

5.4.4 Reduce V_t for Timing Closure

With the delay penalty caused by changing V_t from low to high in the hot region for leakage reduction, we may violate the circuit's timing constraint by as much as the upper bound given in Lemma 3. The goal in this step is to meet the timing constraint before the next iteration of MIS selection in hot regions. As a dual problem to the previous one, we propose to find minimal number of high V_t cells in low temperature region and change them to low V_t, so that the leakage overhead can be minimized. Our heuristic algorithm is shown in Fig. 5.5 below.

Candidate cells that can be used to make for timing closure should be high V_t cells on negative slack path(s) (line 2). *Candidate* can be built in $O(n)$ by checking

```
find the region $R$ that has the highest temperature $T$;
mark the cells with high $V_t$ in $R$;
compute $w_i(T)$ by (5.9) for each low $V_t$ cell in $R$;
sort low $V_t$ cells in $R$ in decreasing order by $w_i(T)$;
for each unmarked cell $c_i$ in $R$ in the above order
    add cell $c_i$ into candidate set $MIS$;
    mark $c_i$ and all the cells in $R \cap (TO_i \cup TI_i)$
change $c_i$ in $MIS$ to high $V_t$
```

Figure 5.4: Selecting an MIS of low V_t cells from the hot region.

the cell one by one if it has high V_t and negative slack. For each cell c_i in the set $Candidate$, we compute the leakage increase and delay decrease when we reduce its V_t (line 3). Negative slack cells whose timing can get improved by c_i are in the set Eff_i (line 4). Eff_i can be built by one traversal on Cov_i while checking if the cell has negative slack. This for loop has complexity $O(n^2)$.

Then as long as there is timing violation (line 5), we find the best cell c_i based on a metric similar to w_i defined in (5.9) and change it to low V_t (lines 6, $O(n)$). According to the metric, cells in low temperature region are in favor because they cause less leakage increase comparing to the same cells in higher temperature region. Also, cells that compensate more negative slack cells and with larger delay saving and are favorable, because fewer such cells are likely to be needed. We perform incremental timing analysis in line 8, which has a complexity of $O(n)$ according to [102]. The set of negative slack cells, N, may shrink if a cell's slack becomes zero or positive (line 9). Updating Eff_i for each c_i remaining in $Candidate$ takes $O(n^2)$ for the same reasoning as above.

Since each time a cell gets assigned to low V_t, the while loop executes at most

```
1. $H$ = set of high $V_t$ cells; $N$ = set of cells with negative slack;
2. for each $c_i \in Candidate = H \cap N$
3.    compute $\Delta D_i(T), \Delta L_i(T)$ at $c_i$'s steady-state temperature $T$;
4.    $Eff_i = N \cap Cov_i$;
5. while (negative slack cell set $N \neq \phi$)
6.    find $c_i \in Candidate$ with the largest $\Delta D_i \times |Eff_i| / \Delta L_i(T)$;
7.    change $c_i$ to low $V_t$ and remove it from $Candidate$;
8.    incremememental timing analysis;
9.    $N, H$ change, update $Eff_i$ for each $c_i \in Candidate$;
```

Figure 5.5: Selecting a minimal set of high V_t cells to meet timing.

n times before $N = \phi$ (this is guaranteed because the timing will be met when all cells have low V_t). Therefore, the complexity of this heurisitc is bounded by $O(n^3)$.

5.4.5 Update of Steady-State Temperature and Iteration of Procedure

After we raise V_t in hot regions and reduce V_t in cold regions for timing closure, steady-state temperature corresponding to the current V_t assignment is computed based on the iterative approach indicated in Section 5.3.2. Total leakage under such temperature and leakage savings are also computed.

Based on the leakage saving and temperature variation from this round of V_t adjustment, we can decide whether further V_t adjustment is needed. If large leakage saving is obtained from the current round, we will repeat the procedure in Section 5.4.3 and Section 5.4.4 for further leakage reduction, and evaluate the steady-state temperature and leakage saving again.

In summary, because each low V_t cell can be included in an MIS at most once in Fig. 5.4 for leakage reduction, the number of iterations will be bounded by n. Therefore, the overall complexity of our algorithm is $O(n^4)$ in the worst case. As we have analyzed and will show by results, real run time complexity is much lower.

5.5 Experimental Results

We use 17 different types of cells from a TSMC 65nm low-power dual-V_t library, where each cell has both high V_t and low V_t implementation.

We apply our technique on five benchmarks from OpenCores [115]. The number of cells in the design ranges from a few thousands to over 50K as shown in the second column in Table 5.1. We synthesize these benchmarks using *Synopsys Design Compiler* and flatten the netlist. We then use the flat design flow to place the cells using *Synopsys IC Compiler*. This is because dual-V_t technique is usually applied on standard cells in flattened design. For hierarchical designs, we can first apply our technique within each macro and then apply it to the standard cells at top level. The ambient temperature is assumed to be 25^{o}C. We report leakage power at 25^{o}C and the dynamic power for each cell from *IC Compiler* and calculate the leakage power at any given temperature using (5.1).

Based on the thermal model in Section 5.3.2, we partition the chip into a 40 × 40 × 6 3-D mesh. The same thermal conductivity parameters as in [96] are adopted: 10 W/m^{o}C for the top, 8800 W/m^{o}C for the bottom, 7 W/m^{o}C for the side, and 150 W/m^{o}C for the silicon. Cells scatter over the 40 × 40 regions on the surface of the chip. A cell is said to belong to one region if the coordinates of the cell's center falls in that region. Cells in one region are considered to have the same temperature.

We first use *Synopsys Power Compiler* to perform dual-V_t assignment on the circuits. *Synopsys Power Compiler* utilizes the algorithm described in [111] and does not consider temperature variation over the chip and its effect on leakage. Then we apply our temperature-aware dual-V_t assignment technique to re-assign V_t values on some of the cells based on this initial assignment. The experimental results are reported in Table 5.1 to show leakage savings and Table 5.2 to show cell temperature changes achieved by our technique.

Table 5.1: Comparison in power consumption between the levelization-based dual-V_t assignment algorithm and our temperature-aware enhancement technique.

Benchmark	#cells	% of high V_t		Power (uW)		leakage
		initial	new	dynamic	leakage	saving
des3	2698	68%	67%	2563.6	120.0	6.1%
systemcaes	7048	83%	84%	5061.3	480.4	8.0%
ac97	8824	64%	66%	11506.1	2093.3	12.0%
aes_core	13816	77%	69%	10013.7	1678.8	17.3%
double_fpu	54937	84%	85%	31775.3	3460.1	12.7%
average		76%	74%			11.2%

Table 5.2: Comparison in cell temperature between the levelization-based dual-V_t assignment algorithm and our temperature-aware enhancement technique.

Benchmark	initial			new			runtimes (s)
	T_peak	T_avg	#hot	T_peak	T_avg	#hot	
des3	37.0	29.1	12	36.0	29.1	11	15
systemcaes	66.1	48.3	11	63.9	48.2	5	55
ac97	89.9	73.8	44	88.8	73.5	3	67
aes_core	70.4	46.7	1309	68.5	46.0	863	149
double_fpu	76.5	51.6	1911	75.5	51.5	1856	367

In Table 5.1, the third and fourth column show the percentage of high-V_t cells in the initial assignment and after applying our technique. On average, 76% of the total cells have been assigned high-V_t by the intial assignment, and 74% high-V_t cells by our technique. These two figures are close because the initial dual-V_t assignment algorithm is a greedy algorithm that already tries to utilize the timing slack on non-critical paths as much as possible to minimize leakage, while our technique seeks to assign such timing slack in a better way to achieve larger leakage reduction. The slight decrease in the percentage of high V_t cells further indicates that the location of high V_t cells matter more than the number of high V_t cells.

The fifth column shows the dynamic power in the circuit; the sixth and seventh columns show the leakage power in the combinational part of the design based on initial dual-V_t assignment and the leakage reduction by our temperature-aware V_t assignment technique. We only implemented the temperature-aware dual V_t assignment technique for combinational cells for comparison purpose, because [111] apply only to combinational circuits. We are currently working to expand our temperature-aware dual-V_t asssign to include sequential cells. In the two big circuits (aes_core and $double\_fpu$), our technique can achieve leakage reduction of 17.3% and 12.7% respectively. In the three small circuits, the leakage saving is smaller (8.7% on average). This is because the timing paths between flip-flops are much shorter in those small circuits, so that the temperature variation along the paths are smaller.

In Table 5.2, it compares the peak temperature, the average temperature and the number of "hot" cells in the circuits before and after applying our technique respectively. We see that after applying our technique the peak temperatures are reduced by one to two degrees celsius and the average temperature is almost the same. This is because of the following two reasons: first, the dynamic power in this library is still dominating especially when the temperature is low. Therefore, the reduction in leakage power does not affect the peak temperature significantly. Second, since we change some of the low-V_t cells in the hot regions to high V_t, we have to switch some high-V_t cells in the mild or cold regions to low-V_t in order to make up timing. This will slightly increase the temperature of those cold regions and therefore the average temperature remains at the same level. The “hot” cells are defined as the cells whose temperatures are higher than 90% of the peak temperature in the original circuit. The 4th and 7th columns show that the number of “hot” cells have been reduced by our technique.

Finally the last column in Table 5.2 shows the runtimes of our algorithm. The longest runtime is 367 seconds for the $55K$ instance design $double\_fpu$.

5.6 Summary

In this chapter we propose the first temperature-aware enhancement approach to improve the leakage performance in dual-Vt assignment algorithm. Experimental results show that we can achieve more leakage saving and fewer gates in high temperature regions while meeting the timing requirement. It is interesting and debatable whether the delay-temperature dependency should be explored for further leakage reduction. Such dependency does exist and has been well-documented and studied at high level such as microarchitectural level. Our preliminary results show that if we are allowed to take advantage of the delay at the steady-state temperature, which is normally much lower than the highest operating temperature, we will be able to enjoy a much more significant leakage saving and a more balanced temperature distribution across the chip.However, the risk of such approach is that timing might be violated for example when temperature during a stretch is higher than the steady-state temperature. Further researches can be done regarding this issue.

Chapter 6

Enhancing DVS Technique for Real-Time Systems by Incorporating Temperature-Leakage Interdependency

Chip temperature is playing a more and more important role in sub-65nm designs. High temperature not only create problems in system reliability and performance, it also causes sharp increase in leakage current, which in turn increases the system power dissipation and could further raises the temperature. When the cooling devices cannot absorb the heat, such positive feedback between leakage and temperature can result in thermal runaway [121, 129]. Developing an accurate temperature model is thus crucial for many purposes such as hotspot prediction and leakge estimation.

Temperature information can be obtained either from on-chip thermal sensors [122, 139] or from temperature models that treat on-chip heat transfer as an RC circuit using the following differential equation [129, 130, 133]:

$$C\frac{dT}{dt} + \frac{T - T_{amb}}{R} = P \tag{6.1}$$

where C and R are the lumped thermal parameters of the system, T_{amb} is the ambient temperature, and P is the power dissipation on the chip. The values of the constant parameters are listed in the results section.

Ideally, the power on the r.h.s. of equation (6.1) should include all sources of power disipation. Curve 2 in Figure 6.1 shows the numerically accurate transient temperature of equation (6.1) obtained by Runge-Kutta formula and Dormand-Prince pair [123, 126].

However, many models consider only dynamic power and ignore leakage (e.g., the HotSpot tool [133]), or treat leakage as a constant. Curve 4 in Figure 6.1 shows the temperature when leakage is ignored, where we see a significant underestimation of the steady state temperature by around 5 degree Kelvin. Curve 1 in Figure 6.1 uses the leakage value at the maximal manufacture specified temperature ($370K$ in this case) and overestimates the steady state temperature by about 4 degree Kelvin. More accurate models use a piece-wise linear function to estimate leakage at different temperature regions [121, 130]. But they are not suitable for real-time applications in general. We will elaborate these in the next section.

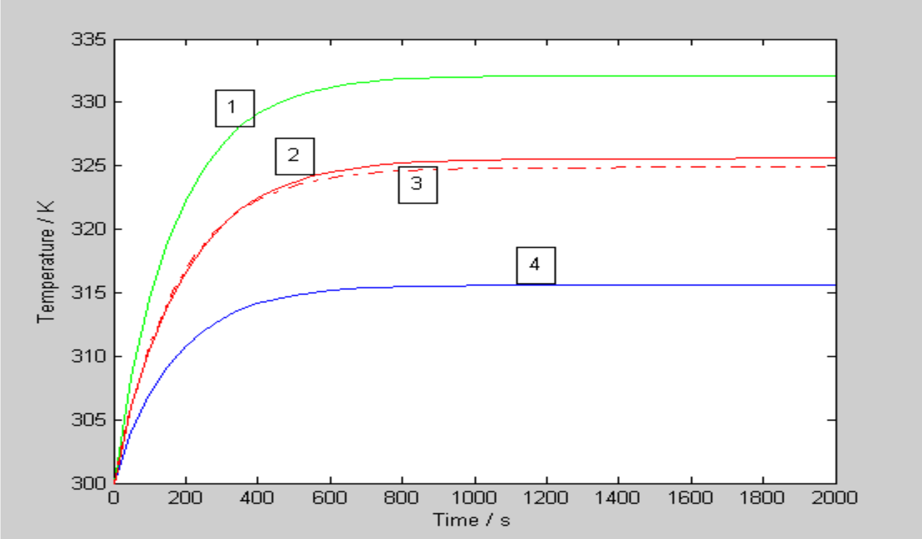

Figure 6.1: Temperature predicted by using different power models.

From this, we see clearly the need for an accurate temperature model that captures leakage's impact. Our first goal is to develop such a model. To do this, we consider leakage as a function of time in Equation 6.1 and derive an anyltical solution for the temperature model. Curve 3 (dashed curve) in Figure 6.1 is the temperature predicted by our model, which is a very good match to the baseline temperature Curve 2. Simulation demonstrates that our model's mean absolute error is within 0.5 degree Kelvin on average over different type of applications.

The second goal of this paper is to apply our derived temperature model to estimate the temperature-dependent leakage and guide dynamic voltage scaling approach for total energy minimization on real-time applications. We report the proven optimal solution for the case of single task and propose an online heuristic for multiple tasks. On 10 sets of real-time tasks, traditional DVS [136, 132] and the leakage-aware CS-DVS algorithms [125, 128] consume on average 6% and 9% more total energy than our approach. It is also interesting to learn that CS-DVS overestimates leakage and selects voltage higher than necessary, which results in a slight more energy consumption than the DVS algorithms that do not consider leakage.

6.1 Modeling Temperature-Leakage Interdependency

Our goal is to find an accurate model based on equation (6.1) to describe how temperature is impacted by power, particularly leakage.

6.1.1 Existing Models

Case 1. The most widely used solution to equation (6.1) treats power P as a constant that is independent of temperature and time. Then equation (6.1) becomes linear and can be trivially solved . Specifically, let T_0 be the temperature at time $t = 0$, temperature at time t can be expressed as

$$T(t) = P \cdot R + T_{amb} + (T_0 - P \cdot R - T_{amb})e^{-t/(R \cdot C)} \tag{6.2}$$

A representative example of this case is the popular HotSpot simulator [133] where P is treated as dynamic power only. Because that total power dissipation P is always higher than dynamic power, this will underestimate temperature as we have seen in Figure 6.1. This problem remains when we add leakage to P as a temperature-independent constant. For instance, when leakage is calculated at the maximal temperature, we will overestimate temperature as shown in Figure 6.1.

Therefore, we need to first model leakage as a function of temperature. Subthreshold leakage and gate leakage are the two major sources of IC circuit's leakage, and only the former is very sensitive to temperature [121, 129]. We adopt the following model for circuit's leakage current

$$I_{leak} = I_0 \cdot (A \cdot T^2 \cdot e^{\frac{\alpha V_{dd} + \beta}{T}} + B \cdot e^{\gamma V_{dd} + \delta}) \cdot N_{gate} \tag{6.3}$$

where the constants can be obtained empirically and for the 65nm technology, we have $A = 1.1e - 12, B = 1.0e - 14, \alpha = 466, \beta = -1225, \gamma = 6.3, \delta = 6.9, I_0 = 1.03e + 3$, and N_{gate} is the gate count of the circuit [129].

Case 2. There is no analytic solution for Equation (6.1) when we use Equation (6.3) for leakage. A popular approach is to use piece-wise linear approximation for leakage. For example, in [131], the authors use the following 2-piece linear model for the leakage dependent temperature:

$$
\begin{aligned}
P_{leak} &= P_{mid} - k_1(T_{mid} - T) \qquad T_{min} < T \leq T_{mid} \\
&= P_{max} - k_2(T_{max} - T) \qquad T_{mid} < T \leq T_{max}
\end{aligned}
$$

This linear dependency ensures that Equation (6.1) is linear and solvable. In addition, piece-wise linear approximation can provide arbitrarily accurate estimation when the length of each piece is sufficiently small. However, it is not clear how many pieces should be used and how to select mid-points (such as T_{mid} in the above equation) for the best fit. Besides, the coefficients (such as P_{mid}, k_1 and k_2) are all circuit dependent.

A recent study [130] proposes to separate leakage into two parts, the part that linearly depends on temperature and the high-order part. The non-linear part is the high-order terms in the Taylor series, which can be truncated around a reference temperature. They report the leakage estimation error when the number of pieces goes from 1 to 10. They show that the error can be reduced by 50% each time when we go from 1-piece to 2-piece and then to 3-piece, and five or more pieces are required to keep the error below 1% for their benchmarks. However,this Taylor series based approximation limits their model to be effective only around the reference temperature and equation (6.1) in general cannot be solved with these high-order terms.

6.1.2 The Proposed Model

We consider P as the sum of dynamic power and leakage power $I_{leak} \cdot V_{dd}$ and use equation (6.3) for I_{leak}. Equation (6.1) becomes

$$\begin{aligned}\frac{dT}{dt} &= \frac{I_0 \cdot A \cdot V_{dd} \cdot N_{gate}}{C} \cdot T^2 \cdot e^{\frac{\alpha V_{dd}+\beta}{T}} \\ &\quad -\frac{T}{R \cdot C} + \frac{T_{amb}}{R \cdot C} + \frac{P_{dyn}}{C} \\ &\quad +\frac{I_0 \cdot B \cdot V_{dd} \cdot N_{gate}}{C} \cdot e^{\gamma V_{dd}+\delta}\end{aligned}$$

This non-linear differential equation does not have analytic solution. However, we observe that the term $e^{\frac{\alpha \cdot V_{dd}+\beta}{T}}$ does not have dramatic change when T varies in the working temperature range. Therefore, we approximate this term by a constant and obtain the following solution

$$T(t) = -r + \frac{s-r}{\frac{T_0+s}{T_0+r} \cdot e^{\frac{-t}{R \cdot C} \cdot \sqrt{1-4 \cdot a \cdot b}} - 1} \tag{6.4}$$

where $r = \frac{-1-\sqrt{1-4 \cdot a \cdot b}}{2 \cdot a}$, $s = \frac{-1+\sqrt{1-4 \cdot a \cdot b}}{2 \cdot a}$, $a = I_0 \cdot A \cdot V_{dd} \cdot N_{gate} \cdot R \cdot e^{\frac{\alpha \cdot V_{dd}+\beta}{T_0+P_{dyn} \cdot R}}$, and $b = I_0 \cdot B \cdot e^{\gamma V_{dd}+\delta} \cdot V_{dd} \cdot N_{gate} \cdot R + P_{dyn} \cdot R + T_{amb}$.

The derivation of equation (6.4) is tedious, which is based on the following

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{x+a}{x+b} + c$$

Curve 2 in Figure 1 is drawn using this equation and we see it is very close to the baseline Curve 1 obtained numerically.

6.2 DVS Scheduling for Total Energy Minimization

In this section, we first present the models for system and energy and the basics of the dynamic voltage scaling (DVS) techniques. Then we study how to incorporate the temperature-leakage interdependency into DVS for total energy minmization.

6.2.1 Current DVS Approaches

We consider a system that will process real-time tasks. Each task is characterized by a deadline d_i and its workload w_i.

During execution at V_{dd}, the dynamic power P_{dyn}, leakage power P_{leak}, and clock frequency f are

$$P_{dyn} = K_1 \cdot C_{eff} \cdot V_{dd}^2 \cdot f \tag{6.5}$$

$$P_{leak} = I_{leak} \cdot V_{dd} \tag{6.6}$$

$$f = K_2 \cdot \frac{(V_{dd} - V_{th})^{\chi}}{V_{dd}} \tag{6.7}$$

where K_1 and K_2 are constants, C_{eff} is the effective capacitance, I_{leak} is the leakage current, and χ is technology dependent with value between 1.0 and 2.0 [129].

With a low V_{dd}, the system runs at a slow clock and consumes less P_{dyn}. DVS techniques are originally developed to leverage this non-linear dependency between power/energy and supply voltage to dynamically change V_{dd} and thus clock frequency such that each task will be finished by its deadline and the energy due to dynamic power will be minimized. It is proved that the best DVS strategy, which we refer to as the **traditional DVS**, will execute the task at the lowest frequency to complete the task exactly on the deadline [136, 132].

Variations of DVS approaches were proposed later with different focuses. [134] improved traditional DVFS efficiency by exploiting the path-grained adaptability. NBTI-aware DVFS is proposed in [118] that utilizes the guard band to improve power consumption as well as lifetime without compromising performance against NBTI degradation.

When leakage becomes significant in system's total power, the low power sleep modes are introduced so system only need power P_{sleep}, not dissipating dynamic or leakage power. We use E_{wakeup} for the energy overhead to activate the system from the sleep mode and a binary variable δ to indicate whether the system is in sleep mode ($\delta = 1$) or not ($\delta = 0$).

Leakage-aware DVS methods have been proposed to utilize this sleep mode and balance dynamic and leakage in order to reach minimal total energy [125, 128, 137]. Basically, the system will operate faster than the slowest clock frequency to finish the work early and then shut down (or go to sleep mode) to save leakage and hence total energy. For example, in [128] and [125], a critical speed DVS (**CS-DVS**) approach is proposed where the system will not run below a centain speed.

Nonsurprisingly, chip temperature is the next addition to DVS research due to leakage's strong dependency on temperature as we have shown in the previous section. There are many work targetting the temperature-dependent leakage or total energy reduction [116, 119, 120, 138]. They treat leakage current either as a constant or use piece-wise linear models to estimate leakage at different temperature where their temperature models do not properly consider leakage's impact. In [117], it proposed an online temperature aware DVFS technique which consists of two stages:

an offline temperature-aware optimization step and online voltage/frequency setting based on temperature sensor readings[1].

Combining Equations (6.3) and (6.4), we see that if a system is active for time 0 to t_{active} and then go to sleep for time t_{sleep}, its total energy will be:

$$\delta \cdot E_{wakeup} + P_{dyn} \cdot t_{active} + \int_0^{t_{active}} P_{leak} dt + P_{sleep} \cdot t_{sleep} \tag{6.8}$$

The temperature-dependent leakage is included in the integration. Temperature affects gate delay as well. However, the system's clock frequency f depends on many issues and we follow the industry practice by assuming that f only changes when throttling happens or when V_{dd} changes and does not change with temperature. If V_{dd} changes during the execution, the dynamic energy will also be replaced by an integration to reflect this. In the rest of this section, we re-evaluate DVS techniques under total energy model in Equation (6.8) which considers the temperature-leakage interdependency.

6.2.2 Optimal Solution for A Single Task

For a single task, the problem can be described as: *how to use the minimal total energy consumption to complete w units of computation within time interval* $[0, d]$.

We first consider the simplified case that the task starts execution from time zero with temperature T_0. We derive the following lemma to obtain the optimal solution.

[1] Reading from temperature sensor is a separate topic and is out of the scope of our discussion.

Lemma 1. To complete w units of computation and start execution from time zero with temperature T_0, leakage energy $E_{leak} = \int_0^{t_{active}} P_{leak}(t)dt$ increases as V_{dd} increases.

[Proof:] The temperature-dependent leakage power $P_{leak}(t)$ is given by equations (6.3) and (6.6). Hence, leakage energy E_{leak}, as expressed in the integration, is a function of V_{dd}. Let f be the clock frequency, then $t_{active} = \frac{w}{f}$. Since f is a function of V_{dd}, t_{active} is also a function of V_{dd}.

$$\begin{aligned}
\frac{dE_{leak}}{dV_{dd}} &= \frac{d}{dV_{dd}} \int_0^{t_{active}} P_{leak}(t)dt \\
&= \int_0^{t_{active}} \frac{dP_{leak}(t)}{dV_{dd}} dt + P_{leak}(t_{active}) \cdot \frac{dt_{active}}{dV_{dd}} \\
&= \int_0^{t_{active}} (I_{leak}(t) + \frac{dI_{leak}(t)}{dV_{dd}} \cdot V_{dd})dt + P_{leak}(t_{active}) \cdot \frac{d}{dV_{dd}}(\frac{w}{f}) \\
&= \int_0^{t_{active}} (I_{leak}(t) + \frac{dI_{leak}(t)}{dV_{dd}} \cdot V_{dd})dt - I_{leak}(t_{active}) \cdot \frac{w}{f} \cdot \frac{\chi \cdot V_{dd} - V_{dd} + V_{th}}{V_{dd} - V_{th}} \\
&\geq \int_0^{t_{active}} (I_{leak}(t) + \frac{dI_{leak}(t)}{dV_{dd}} \cdot V_{dd} - I_{leak}(t_{active}) \cdot \frac{V_{dd} + V_{th}}{V_{dd} - V_{th}})dt
\end{aligned}$$

The last inequality is based on the fact that $\chi \leq 2$. From equations (6.3),

$$\begin{aligned}
\frac{dI_{leak}(t)}{dV_{dd}} &= \frac{\partial I_{leak}}{\partial V_{dd}} + \frac{\partial I_{leak}}{\partial T} \cdot \frac{dT}{dV_{dd}} \\
\frac{\partial I_{leak}}{\partial V_{dd}} &= \alpha \cdot I_0 \cdot A \cdot T^2 \cdot e^{\frac{\alpha V_{dd} + \beta}{T}} \cdot N_{gate} + \gamma \cdot I_0 \cdot B \cdot e^{\gamma V_{dd} + \delta} \cdot N_{gate} \\
&\geq \min\{\alpha, \gamma\} I_{leak} \\
\frac{\partial I_{leak}}{\partial T} \cdot \frac{dT}{dV_{dd}} &= I_0 \cdot (A \cdot 2T - A \cdot (\alpha V_{dd} + \beta)) \cdot e^{\frac{\alpha V_{dd} + \beta}{T}} \cdot \frac{dT}{dV_{dd}} \\
&\geq 0
\end{aligned}$$

where the second inequality is based on the values of α, γ, and V_{dd} under current technology. Therefore,

$$\frac{dI_{leak}(t)}{dV_{dd}} \geq \min\{\alpha, \gamma\} I_{leak} \tag{6.9}$$

Moreover,

$$\begin{aligned}
\frac{dE_{leak}}{dV_{dd}} &= \int_0^{t_{active}} (I_{leak}(t) \cdot (\min\{\alpha, \gamma\} + 1) - I_{leak}(t_{active}) \cdot \frac{V_{dd} + V_{th}}{V_{dd} + V_{th}}) dt \\
&\geq \int_0^{t_{active}} (I_{leak}^{min} \cdot (\min\{\alpha, \gamma\} + 1) - I_{leak}^{max} \cdot \frac{V_{dd} + V_{th}}{V_{dd} + V_{th}}) dt \\
&\geq 0
\end{aligned}$$

where I_{leak}^{min} is the leakage under ambient temperature, and I_{leak}^{max} is the leakage under worst case temperature. In this work, we assume the ambient temperature of $300K$ and the worst case temperature of $370K$. *END.*

Lemma 1 together with the fact that using high V_{dd} consumes more dynamic energy and more sleep energy (due to the longer sleep time), we conclude the following theorem:

Theorem 1. *The most energy efficient solution for the simplified case is to use the lowest possible voltage to complete the task on its deadline.*

Earlier results contradict with this because in approaches such as CS-DVS, they treat leakage as a temperature-independent constant, and $\frac{dE_{leak}}{dV_{dd}}$ is not always positive; In approaches that use piece-wise linear approximation, $\frac{dI_{leak}(t)}{dV_{dd}} \geq \min\{\alpha, \gamma\} I_{leak}$ does not hold due to the discontinuity at the mid-points when two pieces meet.

When the initial temperature is high, it is sometimes better to let the system idle for a while before executing the task. The idle period at the beginning helps cool down the system, providing a lower starting temperature for the task and therefore resulting in smaller leakage energy consumption. In this case, it addresses the more general problem in which idle time can be leveraged when deciding the voltage level. First of all, it is always preferable to allocate the idle time before executing the task than after. The intuitive proof is given in Figure 6.2. Yet it is left to decide how long the idle time should be allocated, sometimes at the cost of higher voltage level for extremely high initial temperature.

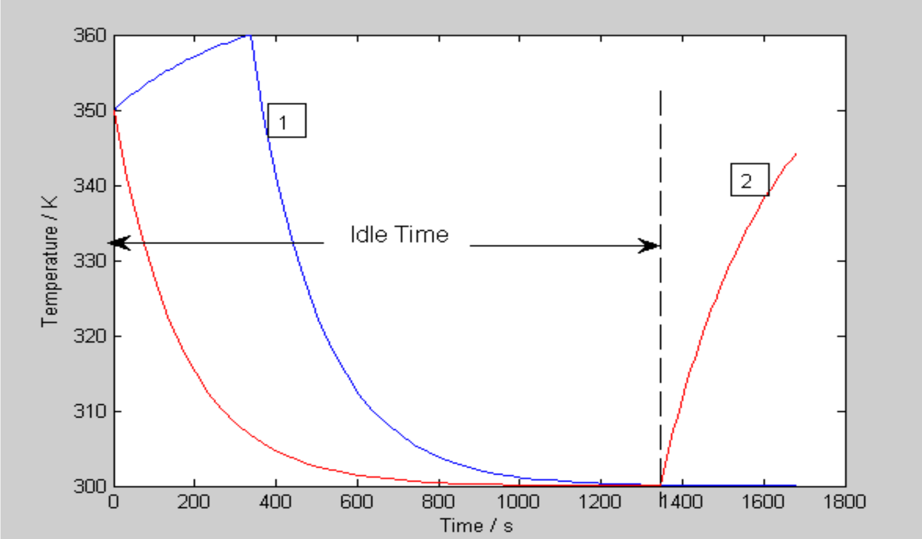

Figure 6.2: Temperature curve of traditional DVS (curve 1) and optimal DVS for a single task (curve 2).

Suppose the system idles for x units of time before executing the task, and the temperature cools down to $\tilde{T}_0$. Apparently, $\tilde{T}_0$ is a function of x. The remaining problem is to select a voltage level for the task to start at time x with temperature $\tilde{T}_0$, and finish before deadline d. We can instead think of the task to start at time zero with initial temperature $\tilde{T}_0$ and finish at deadline $d - x$. This is the simplified case we discussed above. According to Theorem 1, it is optimal to select the lowest possible voltage that completes the task on its deadline. Therefore V_{dd} can be selected accordingly once the idle time x is decided. We can regard V_{dd} as function of x. E_{leak} can therefore be expressed as follows:

$$E_{leak} = \int_0^{d-x} I_{leak} \cdot V_{dd} dt \tag{6.10}$$

where I_{leak} is expressed as in equation 6.3 and T is expressed as in equation 6.4. However we need to replace T_0 with $\tilde{T_0}$ in equation 6.4. Apparently, both I_{leak} and V_{dd} are function of x. We can differentiate E_{leak} over x to derive the x for minimal E_{leak}, however, only numerical solution can be obtained.

6.2.3 Online DVS Heuristic for Multiple Tasks

We now consider a set of non-preemptive real-time tasks, each of which arrives with a workload w_i and a deadline d_i. We want to design an online DVS scheduler that, at the finish time of task $i-1$, selects the next task i from the available tasks and decides the operating voltage with frequency f_i so that the total energy consumption is minimized while all the tasks are completed. The system can scale its operating voltage V_{dd} within $[V_{min}, V_{max}]$. The corresponding frequency range supported by the system is $[f_{min}, f_{max}]$.

Assume that at the current moment, there are k available tasks with workload and deadline (w_i, d_i). Without loss of generality, we assume $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_k}$. Based on the well-known property of the earliest deadline first (EDF) policy, selecting task (w_{i_1}, d_{i_1}) with the frequency:

$$f_{i_1} = \max\{\frac{w_{i_1}}{d_{i_1}}, \frac{w_{i_1}+w_{i_2}}{d_{i_2}}, \cdots, \frac{w_{i_1}+\cdots+w_{i_k}}{d_{i_k}}\} \tag{6.11}$$

ensures the completion of the task set if $f_{i_1} \leq f_{max}$. Otherwise, the set of tasks are not schedulable. Moreover, according to the notion of *critical interval* [135, 136], if $f_{i_1} \geq f_{min}$, it guarantees minimal dynamic energy consumption for these k tasks. Without further increasing the complexity of the online heuristic, we adopt this rule.

When $f_{i_1} \leq f_{min}$, we will be running slightly faster than the frequency decided by EDF policy. It comes the question of how to allocate the extra idle time. For a single task, it has been illustrated in the previous section that it is always better to allocate all the idle time before executing the task. For multiple tasks however, the same strategy does not necessarily apply. As indicated in Figure 6.2, having all idle time before the task will most likely result in high temperature upon its completion. The subsequent task therefore has to start with this high temperature and consume a lot of leakage. If instead, we could move part of the idle time from before to after the task, the system will cool down and the subsequent task can consume much less leakage energy.

Before illustrating our online DVS policy for multiple tasks, there is an important observation regarding the system's cooling process.

Figure 6.3 depicts the cooling curves from different initial temperatures based on equation (6.4). We see that no matter how big the initial temperature gap is (from 370 to 310 in our case), the gap narrows drastically. Experimental data shows that after cooling down for about $1.5 \times R \times C$, the temperature gap will be within 10 degree Kelvin. After that, chip temperature becomes close to T_{amb} and the cooling process slows down.

Based on this observation, we propose the following DVS policy as shown in

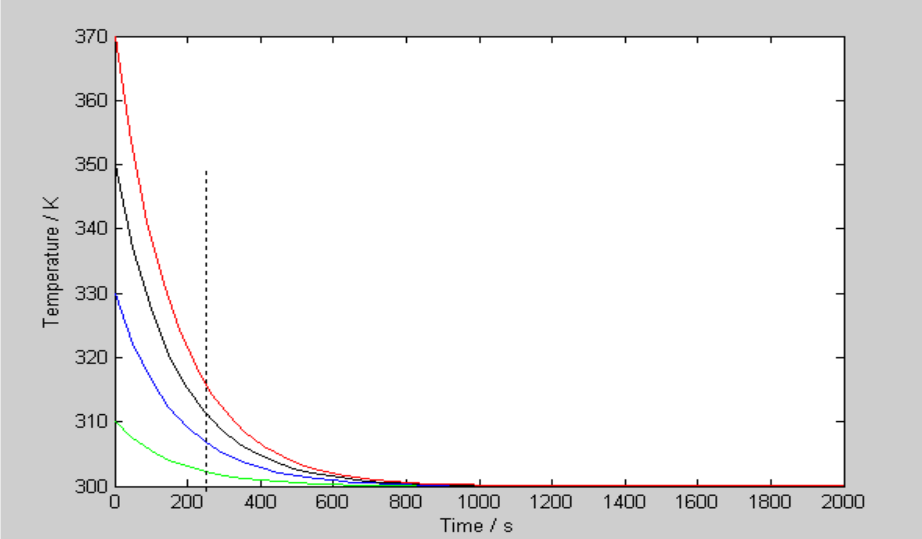

Figure 6.3: Temperature behavior during sleep mode under different starting temperatures

Figure 6.4.

```
0. on the completion of the current task $i$:
1. select the next task $\tau$ for execution based on EDF;
2. determine $V_{dd}$ for the execution of $\tau$ based on frequency in equation (6.11);
3. if $V_{dd} \geq V_{min}$, execute at $V_{dd}$ and go to line 0;
4. total_idleTime = $e_i(V_{dd})$ - $s_i$;
5. front_idleTime = min {1.5RC, total_idleTime};
6. set $s_{i+1}$ as $e_i(V_{min})$, go to line 0;
```

Figure 6.4: Online DVS scheduler for energy minimization.

In line 4-6, $e_i(V_{dd})$ (end time when executing at V_{dd}) is the guardbanding finish time of current task decided by EDF policy, while $e_i(V_{min})$ (end time after idling and then executing at V_{min}) is the actual finish time. s_i is the starting time of current task. $s_0 = 0$.

The proposed heuristic tries to cool down the system for the current task while also takes care of the starting temperature for the subsequent task. When the total idle time is longer than $1.5 \times R \times C$, extra idle time will be merged into the time budget of the subsequent task to help bring temperature down. Only when the total idle time is shorter than $1.5 \times R \times C$, all of it will be put before the current task. In the former case, $e_i(V_{min}) < e_i(V_{dd})$, and the extra idle time is merged into the time budget of the subsequent task by setting the start time of subsequent task as the end time of the current task (line 6); In the latter case, $e_i(V_{min}) = e_i(V_{dd})$.

Figure 6.5 shows the effectiveness of our DVS policy. The temperature curve 1 is based on traditional DVS that puts the total idle time after execution. Curve 2 is based on our proposed DVS policy. It cools down the system first, then executes the task, then idles again after execution. As a result, we see that curve 2 consumes significantly less leakage due to much lower operating temperature (more than 40 degree Kelvin), while its ending temperature quickly decreases to the same level as that of curve 1, even though the idle time of curve 2 after execution is much smaller than that of curve 1. It is because the cooling process has the inverse exponential nature that changes fast at the beginning and gradually slows down. It is also because the peak temperature after execution of curve 2 is much lower than that for curve 1.

6.3 Experimental Results

The goal of the simulation is to (1) validate the accuracy of the proposed temperature-leakage model; (2) compare our proposed online DVS heuristic with other approaches in terms of total energy saving on real-time systems.

Table 6.1 lists the thermal parameters extracted from the single core Intel

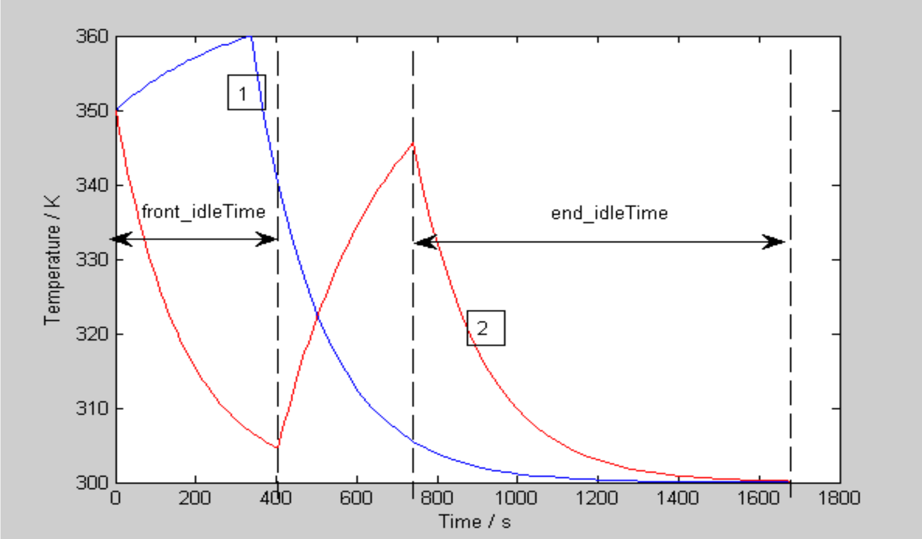

Figure 6.5: Temperature curve of traditional DVS (curve 1) and optimal DVS based on proposed policy for multiple tasks (curve 2).

Celeron Processor 540 at the 65nm technology. We emulate the thermal behavior and power consumption of single processor using Matlab with different initial temperature and different V_{dd}. Then we compute using three models (1) only dynamic power is considered; (2) leakage power under worst case temperature is considered; (3) our model that considers temperature-leakage interdependency. The numerially accurate results [123, 126] are used as the baseline for comparison. Table 6.2 reports the mean absolute error (MAE) over sufficient sampling points.

Clearly, method (1) ignores leakage and underestimates the temperature by $8.25K$ on average. As leakage increases as V_{dd} increases, we see the error increases too. Method (2) overestimates leakage by about 2.5 degree and it has relatively large error when V_{dd} is low. Our proposed model (3) is quite accurate with only $0.51K$ mean absolute error on average. In all cases, initial temperature's impact is not significant.

Table 6.1: Thermal parameters for a single core processor.

R	$1.2^oC/Watt$
C	$140.45mJ/^oC$
T_{amb}	$300K$
T_{worst}	$370K$
V_{dd}_ref	$1.0V$
V_{dd}	$0.8 \sim 1.0V$
f_ref	$2.08GHz$
P_{dyn}_ref	$37.35W$
P_{sleep}	$50\mu W$
V_{th}	$0.3V$

To test the proposed online DVS heuristic over other approaches, we randomly generate 10 different task sets as listed in Table 6.3. Each task set will be executed in a periodic manner, with period length T equal to the global deadline of the task set. The number of tasks ranges from 5 to 15. Task set's utilization factor is defined as $\frac{\sum\limits_{i=1}^{n} w_i}{T}$. When the utilization factor is high, system will run at the highest reference V_{dd} for most of the time, all the approaches will consume similar total energy. This has also been observed and reported in other studies such as [125] and [128]. Therefore we keep the utilization factor below 50% for all task sets.

We repeat each task set for ten times to monitor thermal behavior of the processor, as well as the total energy consumption. We assume the initial temperature as $350K$. Table 6.4 compares the total energy consumption under our approach (denoted as TDVS), traditional DVS, CS-DVS, and Non-DVS scheme. Since the utilization factor of the task sets is around 29%, most tasks can apply the lowest

Table 6.2: Mean absolute error over sufficient sampling points.

$T_{init}(K)$	$V_{dd}(V)$	$MAE1(K)$	$MAE2(K)$	$MAE3(K)$
300	0.8	4.42	3.27	0.14
300	0.9	7.28	3.11	0.31
300	1.0	12.40	1.44	1.08
330	0.8	4.54	3.14	0.17
330	0.9	7.44	2.93	0.33
330	1.0	12.68	1.65	0.75
370	0.8	4.70	2.99	0.38
370	0.9	7.71	2.70	0.54
370	1.0	13.04	1.24	0.92
average MAE		8.25	2.50	0.51

voltage $(0.8V)$ when DVS is enabled. Therefore, Non-DVS consumes much more energy. CS-DVS scheme adopts critical speed, whose corresponding V_{dd} is a little higher than the lowest possible V_{dd} used by DVS and TDVS and consequently consumes more energy than DVS and TDVS. This verifies Lemma 1. Finally, our proposed TDVS approach takes full consideration of the interdependency between leakage and temperature, and allocates the available idle time in an efficient way. We see that without temperature-awareness, DVS consumes 6.16% more energy than our approach, although the same V_{dd} are used for most tasks.

Figure 6.6 shows the transient temperature when we repeat a task set for ten times using our proposed scheduler. The ten star marks around the 325 degree line label the ending temperature of each period. In the first period, we start with 350K and end with 324.4663K. This becomes the starting temperature for the second period and it results in a little lower ending temperature, 324.4627K. For the rest

Table 6.3: Randomly generated task sets.

Taskset	Task Number	Utilization Factor
1	5	23.76%
2	11	40.41%
3	8	31.86%
4	6	33.44%
5	5	26.45%
6	11	29.20%
7	15	27.13%
8	8	22.84%
9	11	33.87%
10	9	21.41%
average	8.9	29.04%

periods, all the ending temperature remain the same. This indicates that our online heuristic becomes stable quickly, an indication for a good online algorithm.

Figure 6.7 reveals the mechanism of how our approach outperforms the other two DVS approaches. Curve 1, 2, 3 correspond to transient temperature in two randomly selected consecutive periods of task set 4 under TDVS, DVS, and CS-DVS scheme. TDVS allocates some idle time before certain task starts so that the temperature and leakage consumption are reduced throughout the task execution. It also allocates the rest of idle time after the task ends, so that it won't lay much temperature burden onto the next task. Note that CS-DVS is likely to exceed worst case temperature. This is because CS-DVS makes inaccurate estimation on the leakage by assuming it as a constant, and mistakenly select a V_{dd} higher than that of DVS and TDVS.

Table 6.4: Comparison of total energy consumption.

Taskset	$E_{TDVS}(10^4J)$	dE_{DVS}	dE_{CSDVS}	dE_{NONDVS}
1	41.4	5.27%	10.87%	69.81%
2	154.6	4.72%	12.24%	70.64%
3	105.8	4.66%	5.05%	69.98%
4	64.14	8.32%	14.35%	65.96%
5	46.66	4.82%	4.82%	69.74%
6	162.9	6.03%	7.13%	83.43%
7	192.2	4.96%	7.59%	78.11%
8	110.6	5.81%	9.95%	79.63%
9	149.4	10.59%	11.01%	94.16%
10	123.1	6.84%	9.32%	83.31%
average	-	6.16%	9.23%	76.48%

Table 6.5: Impact of different initial temperatures.

T_{init}	$E_{TDVS}(10^4J)$	dE_{DVS}
300	63.84	7.86%
330	64.01	8.11%
350	64.14	8.32%
370	64.31	8.56%

Finally, we investigate the impact of initial temperature on the efficiency of our approach. We use task set 4 again as an example. Table 6.5 ranges the initial temperature from $300K$ to $370K$, total energy consumption of TDVS and total energy consumption increase of DVS are reported in the second and third column. Data in the row of $350K$ matches that in Table 6.4. We see that our approach consistently saves more total energy than the traditional DVS by around 8%. When the initial temperature goes up, both approaches will consume more energy and the

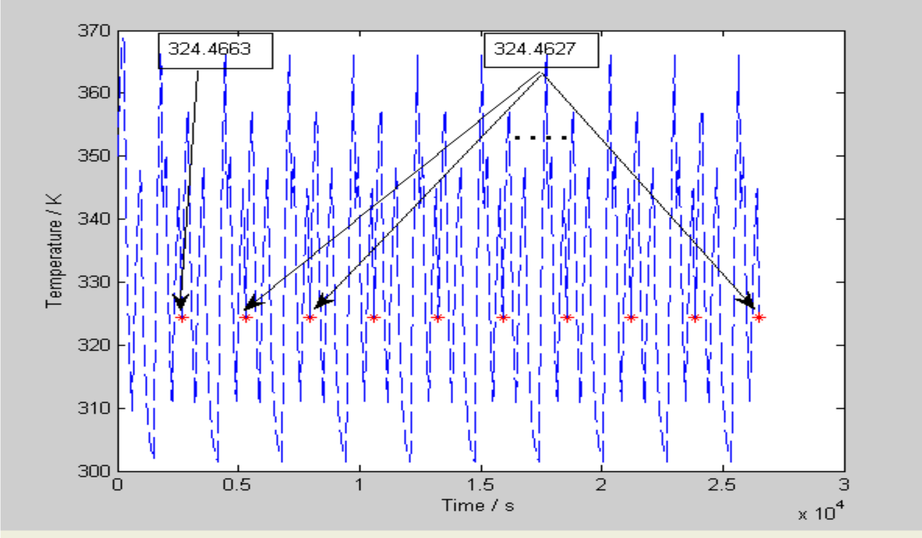

Figure 6.6: Ending temperature of each period under TDVS

advantage of our approach over DVS becomes slightly bigger. But the impact of initial temperature is not significant.

6.4 Summary

In this paper, we show that if leakage's impact on temperature is not considered properly, the temperature estimation can have large error which will affect the solution to all temperature-aware problems, in particular leakage current related problems due to its strong dependency on temperature. We study the temperature-leakage interdependency and propose a new analytic temperature model that gives very accurate temperature prediction. We further apply this model to guide the design of DVS algorithm for total energy minimization. We find that some popular leakage-aware DVS algorithm indeed does not work as well as the traditional DVS for the current technology. We propose an online heuristic DVS approach that is capable of saving more total energy than known algorithms.

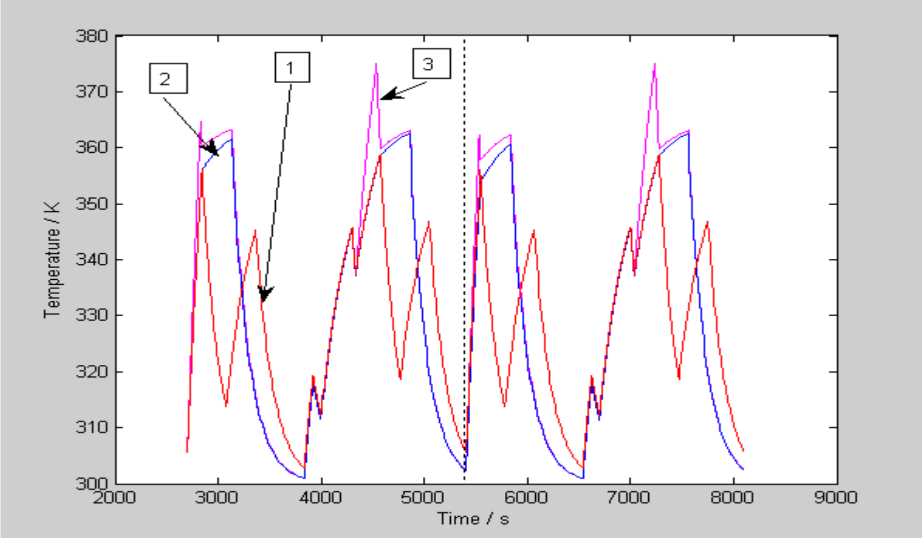

Figure 6.7: Transient temperature curves.

Chapter 7

Conclusion and Future Works

When technology scales down to deep submicron domain, there is an urgent need for more effective power efficient design techniques. We show in this dissertation that there is great potential to enhance the existing CAD tools and algorithms. One such enhancement is to combine multiple techniques simultaneously, another is to integrate physical design with high level synthesis. We demonstrate this approach through the following four case studies.

First, based on the observation that peak current occurs at clock transition and can be characterized by the maximum number of state bits switching in the same direction in the FSM model, we integrate state replication and state re-encoding into a low power encoding approach. This gives us 39.2% reduction in PSV with only 3% overhead in average power.

Second, we propose to use the newly developed process-induced mechanical stress to enhance the performance of dual V_t and gate sizing. Our algorithm based on the concept of *urgent path* shows that about 10% more leakage saving can be achieved when mechanical stress is used and this algorithm outperforms the popular sensitivity-based approach by 13.5%.

Next, we realize that, in the dual V_t technique, using a high V_t cell in a hot region gives much more leakage saving than using the same cell in a low temperature area. But the current EDA tools on dual V_t do not consider leakage's dependency on temperature. We propose a temperature-aware enhancement that can further reduce leakage by 11% and the number of cells in the hot region by 39%.

Finally, it is believed that a high voltage should be used in dynamic voltage scaling (DVS) approach to reduce total energy. After taking the leakage-temperature dependency into consideration, we prove that this belief is incorrect. Indeed, it will lead to a 9% overhead in total energy consumption compared to our proposed approach.

The future of this research will be on the *continuation* and *expansion* of this approach. For the continuation, we will look for opportunities, particularly with the collaboration with EDA industry, to integrate these highly practical enhancement techniques into the existing CAD tools. For the expansion, one interesting topic will be thermal-aware design and thermal management for three-dimensional (3D) integrated circuits (IC). To conclude this dissertation, I briefly describe this below.

The idea of 3D IC has been proposed for a while and the recent progress of through-silicon-via (TSV) technology marks the possible start of the real 3D IC era [141]. 3D integration has the advantage in decreasing interconnection delay, increasing integration density, improving performance, and reducing power consumption [142]. However, due to the vertical interaction of inter-layer heat sources and the 3D thermal gradient, thermal problem will be a bottleneck for 3D IC design. There have been several approaches on temperature-aware floorplanning and cell placement [144, 145, 146]. Thermal management approaches such as thermal-through-silicon-vias and micro-fluidic channel have also been proposed [147, 148, 149, 150]. Like the deep submicron design, there will also be many research challenges and opportunities for power efficient design for 3D IC. I believe that our methodology of enhancing existing techniques with new design features will thrive in this upcoming 3D IC era.

Appendix A

List of publications

1. A. Yao, J. Gu, G. Qu, and S.S. Bhattacharyya, "Energy Efficient Implementation of G.729 for Wireless VoIP Applications", ACM International Conference on Advanced Infocomm Technology, July 2008.

2. J. Gu, A. Yao, G. Qu, and A. Bouridane, “Minimizing Point-to-Point Transmission Energy with Error Correction Coding and Transmission Power Control”, ACM International Conference on Advanced Infocomm Technology, July 2008.

3. J. Gu, C. Zhuo, J. Qian, J. Zhou, and K. Chen, “Transient analysis of irregular power grid by rowbased iterative method”, Journal of Zhejiang University (Engineering Science), Vol. 43 No. 1, Jan. 2009.

4. J. Gu, Q. Zhou, and G. Qu, “Information Hiding for Trusted System Design”, Design Automation Conference, pp. 698-701, July 2009.

5. J. Gu, L. Yuan, and G. Qu, “Temperature-aware Dual-Vt Technique for Leakage Optimization”, International Workshop on Logic and Synthesis, August 2009.

6. J. Gu, L. Yuan, and G. Qu, “SSRR: Peak Current Reduction by Simultaneous State Replication and Re-Encoding”, International Workshop on Logic and

Synthesis, August 2009.

7. J. Gu, G. Qu, T. Chen, and A. Bouridane, “An Adaptive Energy Efficient Transmission Protocol in Wireless Ad-hoc Network”, NASA/ESA Conference on Adaptive Hardware and Systems, pp. 281-288, August 2009.

8. J. Gu, L. Yuan, and G. Qu, “Enhancing dual-Vt design with consideration of on-chip temperature variation”, IEEE International Conference on Computer Design, pp. 542-547, October 2010.

9. J. Gu, L. Yuan, G. Qu, and Q. Zhou, “Peak Current Reduction by Simultaneous State Replication and Re-Encoding”, IEEE/ACM International Conference on Computer-Aided Design, pp. 592-595, November 2010.

10. L. Yuan, S. Leventhal, J. Gu, and G. Qu, “TALk: A Temperature-Aware Leakage Minimization Technique for Real-Time Systems”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (accepted), August 2011.

11. J. Gu, L. Yuan, C. Zhuo, and G. Qu, “Improving Dual Vt Technology by Simultaneous Gate Sizing and Mechanical Stress Optimization”, IEEE/ACM International Conference on Computer-Aided Design (accepted), November 2011.

Bibliography

[1] ITRS 2009 Design Roadmap, http://www.itrs.net/Links/2009ITRS.

[2] R. Puri, et al., “EDA Challenges and Options: Investing for the Future”, *IEEE/ACM Design Automation Conference* pp. 1-2, 2010.

[3] R. K. Krishnamurthy, et al., “High-performance and Low-voltage Challenges for Sub-45nm Microprocessor Circuits”, *International Conference on ASIC* pp. 258-261, 2005.

[4] K. Roy, et al., “Leakage control for deep-submocron circuits”, *VLSI Circuits and Systems* pp. 135-146, 2003.

[5] D. Frank, R. Puri and D. Toma, “Design and CAD Challenges in 45nm CMOS and beyond”, *IEEE International Conference on Computer Aided Design* pp. 329-333, 2006.

[6] D. Blaauw and K. Gala, “Deep-Submicron Issues in High-Performance Design”.

[7] D. Rittman, “Power Optimization Within Nanometer Designs”.

[8] K. Bernstein, et al., “Design and CAD Challenges in sub-90nm CMOS Technologies”, *IEEE International Conference on Computer Aided Design* pp. 129-136, 2003.

[9] J. W. McPherson, “Reliability Challenges for 45nm and Beyond”, *IEEE/ACM Design Automation Conference* pp. 176-181, 2006.

[10] J. Lee and N. S. Kim, “Optimizing throughput of power- and thermal-constrained multicore processors using DVFS and per-core power-gating”, *IEEE/ACM Design Automation Conference* pp. 47-55, 2009.

[11] K. A. Bowman, S. G. Duvall, and J. D. Meindl, “Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration”, *IEEE Journal of Solid-State Circuits* pp. 183-190, 2002.

[12] A. P. Chandrakasan, et al., “Low-power CMOS digital design”, *IEEE Journal of Solid-State Circuits* pp. 473-484, 1992.

[13] J. M. Rabaey, et al., “Low power design methodologies”, *Kluwer Academic Publishers*, 1996.

[14] M. C. Johnson, D. Somasekhar, and K. Roy, “Leakage control with efficient use of transistor stacks in single threshold CMOS”, *Proc. of Design Automation Conference*, pp. 442-445, 1999.

[15] S. M. Kang, “Elements of Low Power Design for Integrated Systems”, *Proc. of International Symposium on Low Power Electronics and Design*, pp. 205-210, 2003.

[16] Z. Chen, et al., “IDDQ testing for deep submicron ICs: Challenges and Solutions”, *IEEE Journal of Design and Test*, pp. 24-33, 2002.

[17] Z. Chen, et al., “Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks”, *Proc. of International Symposium on Low Power Electronics and Design*, pp. 239-244, 1998.

[18] S. Mutoh, et al., “1-V power supply high-speed digital circuit technology with multi-threshold voltage CMOS”, *IEEE Journal of Solid-State Circuits*, pp. 847-854, 1995.

[19] J. Kao, A. Chandrakasan, and D. Antoniadis, “Transistor sizing issues and tool for multi-threshold CMOS technology”, *Proc. of Design Automation Conference*, pp. 495-500, 1997.

[20] Y. Oowaki, et al., “A sub-0.1 um circuit design with substrate-over-biasing”, *Proc. of International Solid-State Circuits Conference*, pp. 88-89, 1998.

[21] F. Assaderaghi, et al., “A dynamic threshold voltage MOSFET(DTMOS) for ultra-low voltage operation”, *Proc. of International Electron Devices Meeting*, pp. 809-812, 1994.

[22] L. Wei, Z. Chen, and K. Roy, “Double gate dynamic threshold voltage (DGDT) SOI MOSFETs for low power high performance designs.” *Proc. of International SOI Conference*, pp. 82-83, 1997.

[23] H. Kawaguchi, K. Nose, and T. Sakurai, “A CMOS scheme for 0.5V supply voltage with pico-ampere standby current”, *Proc. of International Solid-State Circuits Conference*, pp. 192-193, 1998.

[24] S. Lee and T. Sakurai, “Run-time voltage hopping for low-power real-time systems”, *Proc. of Design Automation Conference*, pp. 806-809, 2000.

[25] C. H. Kim and K. Roy, "Dynamic V_{th} scaling scheme for active leakage power reduction", *Proc. of the Conference on Design, Automation and Test in Europe*, pp. 163-167, 2002.

[26] A. Agarwal, H. Li, and K. Roy, "DRG-Cache: A data retention gated-ground cache for low power", *Proc. of Design Automation Conference*, pp. 473-478, 2002.

[27] K. Flautner, et al., "Drowsy caches: Simple techniques for reducing leakage power", *Proc. of International Symposium of Computer Architecture*, pp. 148-157, 2002.

[28] H. Mizuno, "An 18uA standby current, 1.8-V, 200-MHz microprocessor with self-substrate-biased data-retention mode", *IEEE Journal of Solid-State Circuits*, pp. 1492-1500, 1999.

[29] M. Takahashi, et al., "A 60-mw MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme", *IEEE Journal of Solid-State Circuits*, pp. 1772-1780, 1998.

[30] Y. Kanno, et al., "Level converters with high immunity to power-supply bouncing for high-speed sub-1-V LSI's", *Proc. of Symposium of VLSI Circuits*, pp. 202-203, 2000.

[31] K. Usami, et al., "Automated low-power technique exploiting multiple supply voltages applied to a media processor", *IEEE Journal of Solid-State Circuits* pp. 463-472, 1998.

[32] T. D. Burd, et al., "A dynamic voltage scaled microprocessor system", *IEEE Journal of Solid-State Circuits*, pp. 1571-1580, 2000.

[33] L. Benini, et al., "A survey of design techniques for system-level dynamic power management", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* pp. 299-316, 2000.

[34] M. K. Gowan , L. L. Biro , D. B. Jackson, "Power considerations in the design of the Alpha 21264 microprocessor", *Proceedings of the 35th annual conference on Design automation* pp. 726-731, 1998.

[35] Intel, "SA-1100 Microprocessor Technical Reference Manual", 1998.

[36] M. Alidina, et al., "Precomputation-based sequential logic optimization for low power", *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design* pp. 74-81, 1994.

[37] L. Benini, et al., "Designing low-power circuits: practical recipes", *IEEE Circuits and Systems Magazine* pp. 6-25, 2001.

[38] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for low power cmos circuits", *IEEE Trans. on Computer-Aided Design of ICs and Systems*, pp. 665-671, 1996.

[39] D. Chen, M. Sarrafzadeh, G. K. H. Yeap, "State encoding of finite state machines for low power design", *Proc. of International Symposium on Circuits and Systems*, pp. 2309-2312, 1995.

[40] S. Shigematsu, et al., "A 1-V high-speed MTCMOS circuit scheme for power-down applications", *Symposium on VLSI Circuits* pp. 125-126, 1995.

[41] T. Kuroda, et al., "A 0.9-V, 150-MHz, 10-mW, 4 mm/sup2/, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme", *IEEE Journal of Solid-State Circuits* pp. 1770-1779, 1996.

[42] H. Kawaguchi, et al., "A reduced clock-swing flip-flop (RCSFF) for 63% power reduction", *IEEE Journal of Solid-State Circuits* pp. 807-811, 1998.

[43] C. Kim and S. M. Kang, "A low-swing clock double-edge triggered flip-flop", *IEEE Journal of Solid-State Circuits* pp. 648-652, 2002.

[44] B. S. Kong, et al., "Conditional-capture flip-flop technique for statistical power reduction", *IEEE International Solid-State Circuits Conference* pp. 290-291, 2000.

[45] F. Hamzaoglu and M. R. Stan, "Circuit-level techniques to control gate leakage for sub-100nm CMOS", *Proceedings of the 2002 international symposium on Low power electronics and design*, 2002.

[46] J. T. Kao and A. P. Chandrakasan, "Dual-threshold voltage techniques for low-power digital circuits", *IEEE Journal of Solid-State Circuits* pp. 1009-1018, 2000.

[47] G. Yang, Z. Wang, and S. M. Kang, "Leakage-proof Domino circuit design for deep sub-100nm technologies", to be submitted for publication.

[48] C. Chen, A. Srivastava, and M. Sarrafzadeh, "On gate level power optimization using dual-supply voltages", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* pp.616-629, 2001.

[49] M. Igarashi et al., “A low-power design method using multiple supply voltages”, *Proceedings of the 1997 international symposium on Low power electronics and design* pp. 36-41, 1997.

[50] S. J. Shieh, et al., “Design of low-power domino circuits using multiple supply voltages”, *IEEE International Conference on Electronics, Circuits and Systems* pp. 711-714, 2001.

[51] A. Abdollahi, F. Fallah, and M. Pedram, “Leakage Current Reduction in CMOS VLSI Circuits by Input Vector Control”, *IEEE Transaction on Very Large Scale Integration (VLSI) Systems* pp. 140-154, 2003.

[52] L. S. Nielsen and C. Niessen, “Low-power operation using self-timed circuits and adaptive scaling of the supply voltage”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* pp. 391-397, 1994.

[53] K. Roy and S. Prasad, “SYCLOP: Synthesis of CMOS Logic for Low Power Applications”, *IEEE International Conference on Computer Design* pp. 464-467, 1992.

[54] A.H. Ajami, K. Banerjee, A. Mehrotra and M. Pedram, “Analysis of IR-drop Scaling with Implications for Deep Submicron P/G Network Designs”, in *Proc. of ISQED* 2003, pp. 35-40.

[55] L. Benini and G. De Micheli, “State Assignment for Low Power Dissipation”, *IEEE Journal of Solid-State Circuits*, pp.258-268, March 1995

[56] S. Murai. “Method and apparatus of verifying reliability of an integrated circuit against electromigration”, *US Patent 5995732*, Nov. 30, 1999.

[57] J.E. Murguia, et al. “Short-Time Failure of Metal Interconnection Caused by Current Pulses”, *IEEE Electron Device Letters*, Vol. 14, No. 10, pp. 481-483, Oct. 1993.

[58] P. Vuillod , L. Benini , A. Bogliolo , and G. De Micheli. “Clock skew optimization for peak current reduction”, *International Symposium on Low Power Electronics and Design*, pp.265-270, 1996.

[59] A. Vittal, et al., “Clock Skew Optimization for Ground Bounce Control”, *IEEE International Conference on Computer-Aided Design*, pp. 395-399, 1996.

[60] S. Huang, C. Chang, and Y. Nieh, “Fast Multi-Domain Clock Skew Scheduling For Peak Current Reduction”, *Asia and South Pacific Design Automation Conference*, pp. 254-259, 2006.

[61] Z. Yu, M. C. Papaefthymiou, X. Liu, “Skew Spreading For Peak Current Reduction”, *ACM Great Lakes Symposium on VLSI*, pp. 461-464, 2007.

[62] K. Rahimi, “Minimizing Peak Power in Synchronous Logic Circuits”, *ACM Great Lakes Symposium on VLSI*, pp. 247-252, 2007.

[63] A. Mukherjee, and R. Sankaranarayan, “Retiming and Clock Scheduling to Minimize Simultaneous Switching”, *IEEE International SOC Conference*, pp. 259-262, 2004.

[64] Y. Nieh, S.H. Huang, and S.Y. Hsu. “Minimizing peak current via opposite-phase clock tree”, *42nd ACM/IEEE Design Automation Conference*, pp. 182-185, 2005.

[65] R. Samanta, G. Venkataraman, J. Hu, “Clock Buffer Polarity Assignment for Power Noise Reduction”, *IEEE International Conference on Computer-Aided Design*, pp. 558-562, 2006.

[66] P. Chen, K. Ho, and T. Hwang, “Skew Aware Polarity Assignment in Clock Tree”, *ACM Transactions on Design Automation of Electronic Systems*, pp. 376-379, 2007.

[67] H. Jang, and T. Kim, “Simultaneous Clock Buffer Sizing and Polarity Assignment for Power/Ground Noise Minimization”, *ACM/IEEE Design Automation Conference*, pp. 794-797, 2009.

[68] L. Lu, et al., “Peak Current Reduction Using an MTCMOS Technique”, *Asia Symposium on Quality Electronic Design*, pp. 255-259, 2010.

[69] J. Wu, et al., “IR Drop Reduction via a Flip-Flop Resynthesis Technique”, *International Symposium on Quality Electronic Design*, pp. 78-83, 2008.

[70] S.H. Huang, C.M. Chang, and Y.T. Nieh. “State Re-encoding for Peak Current Minimization”, *ICCAD*, pp. 33-38, 2006.

[71] Y. Lee, K. Choi, and T. Kim, “SAT-Based State Encoding for Peak Current Minimization”, *International SoC Design Conference*, pp. 432-435, 2009.

[72] J. Pearl. “Heuristics: intelligent search strategies for computer problem solving”, Reading, Mass: Addison-Wesley Pub. Co., 1984.

[73] T. Villa et al. “Synthesis of FSMs: Logic Optimization”, Kluwer Academic Publishers, 1997.

[74] L. Yuan, G. Qu, T. Villa, and A. Sangiovanni-Vincentelli. "An FSM Re-Engineering Approach to Sequential Circuit Synthesis by State Splitting", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 6, pp. 1159-1164, June 2008

[75] Synopsys Design Compiler 2008.09 User Guide.

[76] Synopsys PrimeTime PX 2008.12 User Guide.

[77] *ftp://mcnc.mcnc.org.*

[78] A. R. Conn, et al., "Jiffytune: circuit optimization using time-domain sensitivities", *IEEE Trans. on Computer-Aided Design of ICs and Systems*, pp. 1292-1309, 1998.

[79] Q. Wang, et al., "Static power optimization of deep submicron CMOS circuits for dual V_t technology", *Proc. of ICCAD*, pp. 851-854, 1996.

[80] S. Sirichotiyakul, et al., "Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing", *Proc. of DAC*, pp. 436-441, 1999.

[81] L. Wei, K. Roy, and C. Koh, "Power minimization by simultaneous dual-V_{th} assignment and gate-sizing", *Proc. of CICC*, pp. 413-416, 2000.

[82] T. Karnik, et al., "Total power optimization by simultaneous dual-V_t allocation and device sizing in high performance microprocessors", *Proc. of DAC*, pp. 486-491, 2002.

[83] H. Chou, Y. Wang, and C. Chen, "Fast and effective gate-sizing with multiple-V_t assignment using generalized lagrangian relaxation", *Proc. of ASPDAC*, pp. 381-386, 2005.

[84] S. Shah, et al., "Discrete V_t assignment and gate sizing using a self-snapping continuous formulation", *Proc. of ICCAD*, pp. 704-711, 2005.

[85] M. Ketkar and S. S. Sapatnekar, "Standby power optimization via transistor sizing and dual threshold voltage assignment", *Proc. of ICCAD*, pp. 375-378, 2002.

[86] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment", *IEEE Trans. on Computer-Aided Design of ICs and Systems*, pp. 223-234, 2010

[87] A. Kahng, P. Sharma, and R. O. Topaloglu, "Chip optimization through STI-stress-aware placement perturbations and fill insertation", *IEEE Trans. on Computer-Aided Design*, pp. 1241-1252, 2008.

[88] K. Yamada, et al., "Layout-aware compact model of MOSFET characteristics variations induced by STI stress", *IEICE Trans. on Electronics*, pp. 1142-1150, 2008.

[89] L. T. Pang, et al., "Measurement and analysis of variability in 45 nm strained-SI CMOS technology", *IEEE Journal of Solid-State Circuits*, pp. 2233-2243, 2009.

[90] V. Joshi, et al., "Mechanical stress aware optimization for leakage power reduction", *IEEE Trans. on Computer-Aided Design of ICs and Systems*, pp. 722-736, 2010.

[91] N. Weste and K. Eshragian, "Principles of CMOS VLSI Design", *Reading, MA: Addison-Wesley*, 1988.

[92] T. Tsukahara, et al., "Design Methodology of CMOS Low Power", *Proc. of ASPDAC*, pp. 394-399, 2003.

[93] 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 ICs and Systems*, pp. 306-318, 2002.

[94] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors", in *Proc. of HPCA*,2001, pp.171-182.

[95] A. Calimera, R.I. Bahar, E. Macii, and M. Poncino, "Reducing leakage power by accounting for temperature inversion dependence in dual-Vt synthesized circuits", in *Proc. of ISLPED*, 2008, pp. 217-220.

[96] G. Chen and S. Sapatnekar, "Partition-driven standard cell thermal placement", in *Proc. of ISPD*, 2003, pp. 75-80.

[97] C-T. Chu, X. Zhang, L. He and T. Jing, "Temperature aware microprocessor floorplanning considering application dependent power load", in *Proc. of ICCAD*, 2007, pp. 586-589.

[98] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs", in *Proc. of ICCAD*, 2004, pp. 306-313.

[99] L. He, W. Liao, and M.R. Stan, “System level leakage reduction considering the interdependence of temperature and leakage” in *Proc. of DAC*, 2004, pp. 12-17.

[100] J. Kim and Y. Shin “Minimizing Leakage Power in Sequential Circuits by Using Mixed Vt Flip-Flops” in *Proc. of ICCAD*, 2007 pp. 797-802.

[101] J.C. Ku, S. Ozdemir, G. Memik and Y. Ismail, “Thermal Management of On-Chip Caches Through Power Density Minimization”, in *Proc. of Micro*, 2005, pp. 283-293.

[102] J. Lee and D.T. Tang, “An Algorithm for Incremental Timing Analysis”, in *Proc. of DAC*, 1995.

[103] J. Long, J. Ku, S.O. Memik, and Y. Ismail, “A Self-adjusting Clock Tree Architecture to Cope with Temperature Variations”, in *Proc. of ICCAD*, 2007, pp. 75-82.

[104] R. Mukherjee and S.O. Memik, “An Integrated Approach to Thermal Management in Behavioral Synthesis”, *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, No. 11, Nov 2006, pp.1165-1174.

[105] K. Roy et al., “Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicron CMOS Circuits”, in *Proc. of IEEE*, vol. 91, No. 2, pp.305-327, Feb. 2003.

[106] K. Skadron, M.R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, “Temperature-Aware Microarchitecture: Modeling and Implementation”, *ACM Trans. on Architecture and Code Optimization*, Vol. 1, pp. 94-125, Mar. 2004.

[107] J. Srinivasan and S.V. Adve, “Predictive Dynamic Thermal Management for Multimedia Applications”, *International Conf. on Supercomputing*, 2003.

[108] 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.

[109] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, “Full chip leakage estimation considering power supply and temperature variations”, in *Proc. of ISLPED*, 2003, pp.78-83.

[110] C.Tsai et atl., “Temperature-Aware Placement for SOCs”, *Proc. of IEEE*, 2006, pp. 1502-1518.

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

[112] 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.

[113] J. Yu, Q.Zhou, and J. Bian, "Peak temperature control in thermal-aware behavioral synthesis through allocating the number of resources", in *Proc. of ASPDAC*, pp. 85-90, 2009.

[114] L. Yuan, S. Leventhal, and G. Qu, "Temperature-aware leakage minimization technique for real-time systems", in *Prof. of ICCAD*, pp. 761-764, 2006.

[115] http://www.opencores.org

[116] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature", *Journal of ACM*, Vol. 54, No. 1, pp. 1-39, 1007.

[117] M. Bao, et al., "On-line Thermal Aware Dynamic Voltage Scaling for Energy Optimization with Frequency/Temperature Dependency Consideration", *Design Automation Conference*, pp. 490-495, 2009.

[118] M. Basoglu, M. Orshansky and M. Erez, "NBTI-Aware DVFS: A New Approach to Saving Energy and Increasing Processor Lifetime", *International Symposium on Low Power Electronics and Design*, pp. 253-258, 2010.

[119] T. Chantem, X. Hu, and R.P. Dick, "Online work maximization under a peak temperature constraint", *ISLPED*, pp. 105-110, 2009.

[120] J.J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints", *RTAS*, pp. 141-150, 2009.

[121] J.H. Choi, A. Bansal, M. Meterelliyoz, J. Murphy, and K. Roy, "Leakage Power Dependent Temeprature Estimation to Predict Thermal Runaway in FinFET Circuits", *ICCAD*, pp. 583-586, 2006.

[122] B. Datta and W. Burleson, "Low-Power and Robust On-Chip Thermal Sensing Using Differential Ring Oscillators", *IEEE Midwest Symposium on Circuits and Systems*, 2007.

[123] G. Forsythe, M. Malcolm, and C. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, New Jersey, 1977.

[124] I. Hong, G. Qu, M. Potkonjak, and M.B. Srivastava. "Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processor", *19th IEEE Real-Time Systems Symposium.* pp. 178-187, December 1998.

[125] R. Jejurikar, C. Pereira and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems", *ACM/IEEE Design Automation Conference*, 2004.

[126] D. Kahaner, C. Moler, and S. Nash, Numerical Methods and Software, Prentice-Hall, New Jersey, 1989.

[127] A. Krum, "Thermal Management",F. Kreith, et al., *The CRC Handbook of Thermal Engineering*, CRC Press, 2000.

[128] 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.

[129] W. Liao, L. He, and K. Lepak, "Temperature-Aware Performance and Power Modeling", *Technical report UCLA Engr. 04-250*, 2004.

[130] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temeprature-dependent intergrated circuit leakage power estimation is easy", *DATE*, pp. 204-209, 2007.

[131] R. Rao and S. Vrudhula, "Performance Optimal Processor Throttling Under Thermal Constraints", *CASES*, 2007.

[132] Y. Shin, K. Choi, and T. Sakurai, "Power Optimization of Real-Time Embedded Systems on Variable Speed Processors", *Proc. of ICCAD*, pp. 365-368, 2000.

[133] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-Aware Microarchitecture", *Proc. of ISCA*, 2003.

[134] G. Yan, et al., "MicroFix: Exploiting Path-grained Timing Adaptability for Improving Power-Performance Effciency", *International Symposium on Low Power Electronics and Design*, pp. 395-400, 2009.

[135] F. Yao, A. Demers, S. Shenker, "A Scheduling Model for Reduced CPU Energy" *Symposium on Foundations of Computer Science*, pp. 374-382, 1995.

[136] 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*, pp. 1827-1837, 2005.

[137] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, “Theoretical and Practical Limits of Dynamic Voltage Scaling”, *Proc. of DAC*, 2004.

[138] S. Zhang and K. S. Chatha, “Approximation algorithm for the temperature-aware scheduling problem”, *ICCAD*, pp. 281-288, 2007.

[139] Y. Zhang and A. Srivastava, “Accurate temperature estimation using noisy thermal sensors”, *DAC*, pp. 472-477, 2009.

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

[141] S. Fujita, et al., “Perspectives and issues in 3D-IC from designers’ point of view”, *Proc. of International Symposium on Circuits and Systems*, pp. 73-76, 2009.

[142] B. Black, et al., “3D processing technology and its impact on IA32 microprocessors”, *Proc. of International Conference on Computer Design*, pp. 316-318, 2004.

[143] Y. ma, et al., “Thermal effects of leakage power in 3D ICs”, *Proc. of International Conference on Green Circuits and Systems*, pp. 578-583, 2010.

[144] J. Cong, J. Wei, and Y. Zhang, “A thermal-driven floorplanning algorithm for 3D ICs”, *Proc. of International Conference on Computer-Aided Design*, pp. 306-313, 2004.

[145] W-L. Hung, et al., “Interconnect and thermal-aware floorplanning for 3D microprocessors”, *Proc. of International Symposium on Quality of Electronic Design*, pp. 98-104, 2006.

[146] P. Zhou et al., “3D-STAF: Scalable Temperature and Leakage Aware Floorplanning for Three-Dimensional Integrated Circuits”, *Proc. of International Conference on Computer-Aided Design*, pp. 590-597, 2007.

[147] J. Cong, and Y. Zhang, “Thermal-driven multilevel routing for 3-D ICs”, *Proc. of Asia and South Pacific Design Automation Conference*, pp. 121-126, 2005.

[148] J. L. Ayala, et al., “Through Silicon Via-Based Grid for Thermal Control in 3D Chips”, *Proc. of Nano-Net International ICST Conference*, pp. 90-98, 2009.

[149] D. Sekar, et al., “A 3D-IC Technology with Integrated Microchannel Cooling”, *Proc. of International Interconnect Technology Conference*, pp. 13-15, 2008.

[150] H. Mizunuma, C-L. Yang, and Y-C. Lu, “Thermal Modeling for 3D-ICs with Integrated Microchannel Cooling”, *Proc. of International Conference on Computer-Aided Design*, pp. 256-263.