UNIVERSITY OF CINCINNATI

Date: 3-Nov-2010

I, Hao Xu, hereby submit this original work as part of the requirements for the degree of:

Doctor of Philosophy

in Computer Science & Engineering

It is entitled:

Runtime Leakage Control in Deep Sub-micron CMOS Technologies

Student Signature: Hao Xu

This work and its defense approved by:

Committee Chair: Ranganadha Vemuri, PhD

Harold Carter, PhD

Chien-In Henry Chen, PhD

Carla Purdy, C, PhD

Wen Ben Jone, PhD
Runtime Leakage Control in Deep Sub-micron CMOS Technologies

A Dissertation submitted to the

Graduate School
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

in the Department of Electrical and Computer Engineering
of the College of Engineering and Applied Science

2010

by

Hao Xu

Bachelor of Engineering (Electrical Engineering)
Nanjing University, China
June 2001

Master of Science (Information and Signal Processing)
Nanjing University, China
June 2004

Thesis Co-advisors and Committee Co-chairs:
Ranga Vemuri, Ph.D. and Web-Ben Jone, Ph.D.
Abstract

MOSFET scaling into deep sub-micro realm has resulted in significant increase in leakage power consumption. In 45nm technology generation and beyond, leakage power consumption will catch up with, and may even dominate, dynamic power consumption. This makes leakage power reduction an indispensable component for low power designs in deep sub-micro technologies. Many leakage control techniques have been introduced and studied so far. They can be characterized into two classes: runtime techniques and design-time techniques. Design-time techniques only modify the circuit and thus have limited capability of leakage reduction. On the other hand, runtime techniques tune the circuit into low-leakage mode according to the variation of circuit workload. When a circuit or a system has substantial slackness in its workload, runtime techniques can yield significant leakage saving. Hence runtime techniques, such as power gating and reverse body biasing, are widely used in industrial practices, and extensively studied in current researches as well.

Since the invention of runtime leakage control techniques (RTLC), most of they have been applied in a very crude manner. Several key questions regarding the design methodologies of RTLC remain unanswered, including how to design the optimum control policy, what is the optimum granularity of applying RTLC, and how to reduce leakage in circuit active mode. Before these questions are answered, RTLC can only be of an ad-hoc style. On top of these major questions, several other common problems in deep sub-micro technologies need to be considered before RTLC converges to a practical technique. These common problems include temperature and process variation, robustness issues in deep sub-micro technologies etc. They make the design of RTLC even more complicated and challenging.

In response to all these questions and challenges, our research aims at answering the major open questions of RTLC, tackling the practical problems in deep sub-micro technologies and finally forging a systematical solution for RTLC. To this end, this thesis studies the corresponding modeling, optimization, design methodology and design automation issues. Our whole study is driven by the following two main ideas. First, we consider that aggressive idleness exploitation is the key to achieve maximal leakage control. Second, we consider that applying RTLC in a finer manner is the key to enable aggressive idleness exploitation. Our study is based on two leakage control techniques: power gating and reserve body biasing. During our analysis, one type of RBB technique, $V_{\text{th}}$ hopping, turns out to be more effective to control leakage in a finer manner. Therefore it becomes the center of our final solution.
To

My Mother, My Father and My Wife
Acknowledgements

My life in the past five years was basically deadline-driven. This kind of life-style were originated in the late 2007, when I just catch a paper deadline, and had a short chat with my advisor, Dr. Ranga Vemuri. I believe his words at that time were: “What is the next deadline? PhD student should not rest.” Although Dr. Vemuri and I did not have conversations very frequently, but I can almost recite every sentence he has spoken to me. I usually kept thinking about our conversation for a few days, and still found enlightening words in it.

Rhodes 838, Dr. Wen-Ben Jone’s office, has always been a source of pleasure and inspiring sparks for me. Dr. Jone and I had discussions in a wide range of topics from research to life experiences, to News and TV series. Every time when I stepped out of his office, I can always see a colorful cloud in the sky of Cincinnati, even if it was a gloomy day. I am very grateful that Dr. Jone also tried his best to provide financial support to me.

Research and papers writing occupy 95% of my time, but not all. I did go to my lab and hung out with my lab-mates, sometimes. I am lucky to have met so many sincere persons in Cincinnati: my batchmates Balaji, Suman, Manoj, my seniors Vijay, Shubhankar, Bala, Almitra, Angan, Huiying, Xin Jia and Jianping, and other great lab-mates Annie, Mike, Romana, Shikhanshu, Ajaay, Natwar, Jon, Amayika, Aditi and Lakshmi. Amazingly, many of my lab-mates have a great sense of humor. That’s probably another tradition of the DDEL.

I am also very thankful to my good friends who have accompanied me along the way: Ke Li, Bo Zhang, Jian Xu, Qiang Han, Yanyun Wang, Danni Shi, Huaping Wu. We had many cheerful moments together. The past five years would be unimaginable to me without them.

Finally, I need to say thank you to Mr. Chris Amann, the supervisor of my graduate assistant work. Chris has given me plenty of freedom to pursue academic goals, to deal with person needs, and meanwhile managed me to work efficiently in Van Wormer. Chris has wide spread of knowledge, ranging from ideology, society, animals and computers. I have learned very much from him, and would not have achieved this much without his support.
Contents

List of Figures v
List of Tables ix

1 Introduction 1
   1.1 The Leakage Power Problem in Nano-scale Technologies 4
      1.1.1 Leakage Power Consumption 4
      1.1.2 Temperature and Process Variation Impact 7
   1.2 Leakage Control in Nano-scale Technologies 8
      1.2.1 Current RTLC Techniques 9
      1.2.2 Challenges for RTLC in Nano-scale Technologies 11
   1.3 Contribution and Organization of This proposal 13

2 Optimum Control Policy for Run-time Leakage Control 15
   2.1 Energy Breakeven Time and Wake-up Time 16
      2.1.1 Definitions 16
      2.1.2 Existing Approaches for EBT Estimation 17
   2.2 Energy Breakeven Time Modeling for Power Gating 18
      2.2.1 Leakage Current Reduction Model 18
      2.2.2 Energy Breakeven Time Modeling for Power Gating 27
      2.2.3 Algorithm and Experimental Results 31
   2.3 Energy Breakeven Time Modeling for Reverse Body Biasing 37
2.3.1 Circuit Modeling During RBB Mode Transition .................................. 37
2.3.2 Energy Breakeven Time Modeling for RBB ........................................ 41
2.3.3 Optimum Design Point of RBB .......................................................... 42
2.3.4 Experimental Results ........................................................................ 43
2.4 Conclusion and Future Work ................................................................. 45

3 Optimum Granularity for Run-time Leakage Control ............................... 47
3.1 Leakage Saving Modeling OF RBB .......................................................... 50
3.2 Leakage Saving and Control Cost With Finer Granularity ....................... 52
   3.2.1 Temporal Idleness Exploitation ......................................................... 52
   3.2.2 Spatial Idleness Exploitation ............................................................ 54
   3.2.3 Control Cost With Finer Granularity ................................................. 56
3.3 Design Automation For Determining The Optimum Granularity ............... 57
   3.3.1 Problem Formulation ...................................................................... 58
   3.3.2 Algorithm to Find A Near-optimum Granularity ............................... 59
3.4 Experimental Results ........................................................................... 59
3.5 Conclusion and Future Work ................................................................. 62

4 Run-time Leakage Control for Active-mode Circuit ................................... 63
4.1 Candidate Techniques Comparison for RALC ......................................... 66
   4.1.1 Four Figures-of-merit to Compare PG and RBB ................................. 66
   4.1.2 Experiments to Compare PG and RBB ............................................... 68
   4.1.3 Overall Considerations ................................................................... 69
4.2 Selective Light $V_{th}$ Hopping (SLITH) .................................................. 70
   4.2.1 Basic $V_{th}$ Hopping Modeling ........................................................ 70
   4.2.2 Light $V_{th}$ Hopping ...................................................................... 72
   4.2.3 Selective $V_{th}$ Hopping ................................................................. 72
   4.2.4 Selective Light $V_{th}$ Hopping (SLITH) ........................................... 74
   4.2.5 Experimental Results of SLITH ......................................................... 76
4.3 Joint Dynamic and Leakage Power Reduction With SLITH .................... 78
4.4 Conclusion and Future Work ........................................ 81

5 Reliability Issues in Leakage Control ................................. 82

5.1 Reliable Power Mode Transition ................................... 84

5.2 Fast Activation Techniques ......................................... 86
  5.2.1 Initial Boost ............................................. 86
  5.2.2 Current Shaping .......................................... 87
  5.2.3 Multi-thread Activation ................................... 90
  5.2.4 Design Automation Flow .................................. 92

5.3 Experimental Results .............................................. 94

5.4 Conclusion and Future Work ....................................... 96

6 Microarchitectural Level Adaptive Leakage Control ............... 97

6.1 Leakage Control Techniques with the Presence of TV/PV ....... 99
  6.1.1 Power Gating, Body Biasing and Light-Weight $V_{th}$ Hopping .... 99
  6.1.2 Leakage Control with TV/PV Impact ........................ 101

6.2 Microarchitectural Level Leakage Control ........................ 102
  6.2.1 Mutual Exclusion, Locality and Exceptions .................. 102
  6.2.2 Leakage Saving Potentials .................................. 103

6.3 The Whole Scheme ................................................ 105
  6.3.1 Leakage Control Resolution and Granularity ................. 106

6.4 Experimental Results .............................................. 107

6.5 Conclusion ........................................................ 109

7 Summary and Future Work ........................................... 110

7.1 Our Contributions .................................................. 111

7.2 Future Work ....................................................... 112

7.3 Power Gating and Body Biasing Modeling with the Presence of Process Variation ........ 113

7.4 Low Power Versus Reliability .................................... 113

7.5 Future Idleness Detection .......................................... 114
List of Figures

1.1 The Nature of Power Changes with Technology ........................................... 2
1.2 Three Main Sources of Leakage in Nano-scale Technologies ....................... 5
1.3 Temperature Impact on Leakage Components ............................................ 7
1.4 Process Variation Impact on Leakage Components .................................... 8
1.5 An Inverter With Ground Gating ............................................................... 9
1.6 An Inverter With Reverse Body Biasing ................................................... 10
1.7 Ideal Solution Coping with Three Challenges .......................................... 13

2.1 Error Caused By EBT Model In [1] ............................................................. 17
2.2 Voltage Biases of NMOS and PMOS With Ground Gating ......................... 19
2.3 LCR Modeling of A Ground Gated NMOS ............................................... 20
2.4 Two-input NAND Gate LCR Modeling ................................................... 21
2.5 Off-state Transistors In Series ................................................................. 22
2.6 Off-state Transistors In Parallel ............................................................... 23
2.7 LCR Modeling At Circuit Level ................................................................. 24
2.8 Derivation Of Piecewise Total Charging Current ....................................... 26
2.9 Piecewise Total Energy Saving $E(t)^r$ .................................................... 28
2.10 EBT Variation For The Circuit Sleeping In Different States ....................... 29
2.11 Create Time Regions To Calculate Overall Energy Saving ......................... 30
2.12 8-input AND Gate Topology ................................................................. 33
2.13 Correlation of Lumped Gate Leakage Current Model on AND8 .................... 34
2.14 Error of Leakage Energy Consumption Estimates on C7552 ....................... 35
2.15 EBT Distributions of AND8, C432 and C3540
2.16 PMOS RBB Implementation for RALR
2.17 Substrate Charging of RBB
2.18 Leakage Current of A Complex Gate
2.19 Model of PMOS RBB Circuit
2.20 Optimum Bias Voltage of RBB
2.21 Optimum Bias Voltage VS. Idle Time T on C1355 65nm
2.22 Total Energy Consumption Estimates of RBB on 32nm C7552
2.23 RBB Optimum Bias Voltage on 65nm C1355 at Idle Time 13.1ns
2.24 Total Energy Saving Model with Different $V_P$ on 65nm C1355

3.1 Coarse-grained VS. Fine-Grained Power Gating
3.2 Coarse-grained, Fine-grained and Optimum-grained
3.3 Optimum Granularity for Runtime Leakage Control
3.4 $V_{th}$ Hopping Implementation
3.5 EBT Variation of Each Gate in C880
3.6 Leakage Saving Impv. by Temporal Partitioning on C880
3.7 Spatial Idleness Due To Partial Input Dependency
3.8 Leakage Saving Impv. by Spatial Partitioning on C432
3.9 Monitoring and Prediction of Mutual Exclusion
3.10 Routing Cost of Bias Volt. Rails for A 2-way Partition
3.11 Improv. of Temporal and Spatial Cut VS. Idleness Prob.
3.12 Improv. of Leakage Saving VS. Extra Area Cost -C7552

4.1 Run-time Active Mode Leakage Reduction
4.2 Pre-computation and Power Gating
4.3 Energy Saving Comparison of PG and RBB on 65nm C1355
4.4 Model of PMOS $V_{th}$ Hopping
4.5 Light $V_{th}$ Hopping on Inverter Chain
4.6 Gate EBT Variation on C880 with 32nm Technology
6.10 ALU Leakage Saving VS. Resolution/Granularity ........................ 109

7.1 Future Workload Information in FSM ................................. 115
List of Tables

2.1 Resistance Impact On Time-varying Leakage Energy .................................. 33
2.2 Correlation of Lumped Gate Leakage Current Model on AND8 .......................... 34
2.3 Error of Leakage Energy Consumption Estimates ........................................ 35
2.4 Model VS. Simulation on Average EBT ....................................................... 36
2.5 Error % of Total Energy Consumption Estimates of RBB ............................ 44

3.1 Experimental Results of $V_{th}$-hopping with M=1-5 .................................... 61

4.1 Energy Overhead of PG and RBB on C1355 .............................................. 67
4.2 Leakage Reduction Exponent of PG and RBB ............................................ 68
4.3 Four Figures-of-merit of PG and RBB ...................................................... 68
4.4 Energy Breakeven Time (ns) of PG and RBB ............................................ 69
4.5 Traditional VS. Light $V_{th}$ Hopping on Inverter Chain 32nm ...................... 72
4.6 Optimum SLITH on An Inverter Chain with 32/45nm Technologies ............. 77
4.7 One-cycle and Two-cycle SLITH with 32nm Technology ............................ 79
4.8 Two-cycle SLITH with 45nm Technology ................................................ 79

5.1 Area/Power/Wake-up Time of Three Approaches ...................................... 95

6.1 Typical Values of PG, BB and LW-VH on ALU 32nm ................................... 101
6.2 Leakage Power and Idleness of FUs (32nm) ............................................. 103
Chapter 1

Introduction

Ever since the invention of the integrated circuit (IC) in the late 1950’s, technology scaling has been the powerful engine driving the semiconductor industry forward for over 50 years. The continuous reduction of transistor feature size and increase of circuit integration level perfectly answer the question of how to make more out of less. Today, an IC with its area smaller than a penny can hold 125 million transistors.

However in the last decade, the speed of technology scaling has been held back by various physical limitations when the transistor feature size approaches deep sub-micro. In particular, parameter uncertainty of smaller devices and heat dissipation due to tighter integration have gradually become two major bottlenecks hindering further integration.

In order to maintain the growth speed as predicted in the Moore’s law, researches have been carried out to attack these two problems. To address the parameter uncertainty problem, conventional circuit design methodologies have been revisited from a stochastic perspective. Besides that, new design paradigms such as fault-tolerant computing also show potentials to alleviate the impact of parameter uncertainty.

On the other side, the heat dissipation problem has been studied in various dimensions. The fundamental solution, design for low power or low energy, has been investigated through the past decade. Nevertheless, designing a circuit with the lowest power or energy consumption is not enough, since the heat dissipation problem poses another constraint on the circuit runtime temperature. If the temperature of a hot spot on a chip exceeds a particular threshold during runtime,
soft errors or even physical damages can occur to the chip. Hence in recent years, thermal management has emerged as another active research area. Finally, as a result of limits in heat dissipation, it is hardly feasible to significantly increase circuit operation frequency any more. Hence parallelism, for example multicore applications, is becoming the primary direction to improve circuit performance in the future.

Design for low power or low energy (DFP) has been an old art since long time ago. The initial motivation though, was to meet the constraint of energy consumption for special purpose electronics devices such as portable devices and human implantation devices. Since then, DFP has drawn more and more attention due to the increasing packaging cost for designs with higher power consumption. Today, the stringent constraint on heat dissipation has forced DFP into an indispensable component of the modern design flow.

Although the history of DFP was long, the nature of the power problem was changing over time. Figure 1.1 shows the timeline of this change. Before the technology reached 0.3um, dynamic power is the only source of power consumption. The power problem can be expressed as simple as:

$$\text{Power} = CV_{DD}^2 f$$  \hspace{1cm} (1.1)

Hence at that time, the DFP techniques were straightforward. Designers could either try to reduce the effective capacitance ($C$) of the circuit, or reduce the operation frequency ($f$) or voltage ($V_{DD}$). Later on when the technology went down, the leakage power starts to take a significant role in the total power consumption. Before the technology reached 100nm, the power problem can be expressed as:

$$\text{Power} = CV_{DD}^2 f + I_{\text{leak}}V_{DD}$$  \hspace{1cm} (1.2)

The emergence of leakage power consumption has created a boost on the variety of DFP tech-
niques. Designers working on the corresponding technologies needed to consider more than just $C/f/V\text{DD}$. They need to consider DFP techniques for leakage control as well, such as MTCMOS [2], power gating [3], reverse body biasing [4] or input vector control [5].

When the technology went below 100nm to deep sub-micro, the power problem became more complicated. Process variation ($PV$) in these technologies affects the design from all aspects, especially on power. On top of that, the mutual dependency between leakage power and runtime temperature variation ($TV$) made the power problem even more intricate. The power problem in deep sub-micro can be expressed as:

$$\text{Power} = F(C, V_{DD}, f, I_{\text{leak}}, TV, PV)$$  \hspace{1cm} (1.3)

As a result, designers working in deep sub-micro technologies are facing more challenges due to the increasing complexity of the power problem. Leakage power control techniques are unavoidably becoming a critical component of DFP. What’s more, both temperature and process variation need to be taken into consideration when performing power estimation and designing DFP. To control leakage, many techniques have been introduced and studied so far. They can be characterized into two classes: runtime techniques and design-time techniques. Design-time techniques only modify the circuit and have limited capability of leakage reduction. On the other hand, runtime techniques tune the circuit into low-leakage mode according to the variation of circuit workload. When a circuit or a system has substantial slackness in its workload, runtime techniques can yield significant leakage saving. Hence runtime techniques, such as power gating and reverse body biasing, are widely used in industrial practices, and extensively studied in current researches as well.

However, ever since the invention of Run-Time Leakage Control techniques (RTLC), most of them have been applied in a very crude manner. Several key questions regarding the design methodologies of RTLC remain unanswered. For instance, how to design the optimum control policy for RTLC? What is the optimum granularity to apply RTLC on a circuit? How to control leakage for a circuit in active mode? Before these questions are answered, RTLC can only be of an ad-hoc style. On top of these major questions, both temperature and process variations need to factored in to form a practical solution. This again, is a challenging task, because runtime temperature and process variation are affected by many other factors. There are already countless researches on each one of these two problems.

In response to these questions and challenges, this proposal aims at answering the key open questions of RTLC, tackling the practical problems, and forging a systematic solution for RTLC in
deep sub-micro technologies. In the rest of this chapter, we start with an in-depth introduction on the nature of the leakage power, the state-of-the-art RTLC techniques and challenges for RTLC in deep sub-micro technologies. Our definition of "deep sub-micro technologies" primarily refers to any static CMOS technology whose feature size is less than 100nm. In our study, the modeling and experiments were mainly performed on 65nm, 45nm and 32nm technologies.

1.1 The Leakage Power Problem in Nano-scale Technologies

As we have seen, the nature of the power problem has become increasingly complicated in the past. In deep sub-micro technologies, the total power consumption of a circuit comprises of two components: dynamic power consumption and leakage power consumption. Each of them consists of several sub-components. Contrary to the past, now there are two new factors, process variation and temperature variation, which have noneligible impact on the total power consumption. In particular, power variation has significant impact on both dynamic and leakage power consumption, while runtime temperature has huge impact on leakage power consumption. To serve the objective of this proposal, in the following, we only focus on leakage power consumption and the impact of temperature and process variations on it.

1.1.1 Leakage Power Consumption

Leakage power consumption in deep sub-micro technologies consists of three main components: sub-threshold leakage, gate leakage and band-to-band tunneling leakage. With technology scaling, each of these leakage components increases drastically, resulting in an increase in the total leakage current. It was observed that other leakage components, such as the gate induced drain leakage (GIDL), the punchthrough current, etc., are not as important as these three. So in the following, we only explain sub-threshold leakage, gate leakage and band-to-band tunneling leakage. Figure 1.2 illustrates the diagram of these three leakage sources.
Subthreshold Leakage

The subthreshold current in a transistor is due to the diffusion of the minority carriers from the source to the drain. The subthreshold current in a transistor is given by [6] as:

\[ I = A \cdot e^{1/mV_T(V_G - V_S - V_{th0} - \gamma'V_S + \eta V_D S)} \cdot (1 - e^{-V_{DS}/v_T}) \]

with \[ A = \mu_0 C_{ox} W/L_{eff}^2 v_T (1.8 e^{-\Delta V_{th}/\eta v_T}) \]  

where \[ V_{th0} \] is the zero bias threshold voltage, \[ v_T \] is the thermal voltage, \[ \gamma' \] is the linearized body effect coefficient, and \[ \eta \] is the DIBL coefficient. In deep sub-micro technologies, the short-channel effects (SCE) reduces the threshold voltage, thereby increasing the subthreshold current. Due to the SCE, the subthreshold current increases with an increase in the drain bias (drain-induced barrier lowering, DIBL) and the reduction in the channel length. In sub-100nm transistors, the quantization of electron energy in the channel reduces the threshold voltage due to a high oxide field, thereby highing the subthreshold current.

Gate Leakage

At the ultrathin gate-oxide regime, due to the high electric field and the low oxide thickness, an electron can tunnel through the gate oxide. This is known as the direct tunneling of an electron and it results in a large gate leakage in deep sub-micro transistors [7]. The gate tunneling current density in a MOSFET is given by:

\[ J_{DT} = A_g \left( \frac{V_{ox}}{T_{ox}} \right)^2 \exp\left( \frac{-B_g (1 - (1 - \frac{V_{ox}}{\phi_{ox}})^2)}{\frac{V_{ox}}{T_{ox}}} \right) \]  

where \[ V_{ox} \] is the potential drop across oxide, \[ \phi_{ox} \] is the barrier height of the tunneling electron (or hole), and \[ T_{ox} \] is the oxide thickness. \[ A_g \] and \[ B_g \] are physical parameters. The gate current increases
exponentially with an increase in the supply voltage (higher oxide field) and a reduction in the oxide thickness (lower tunneling thickness). The gate leakage in a MOSFET is composed of the following components: a) the gate-source/drain overlap region current (\(I_{gso}\) and \(I_{gdo}\)); b) the gate-channel current (\(I_{gc}\)), part of which goes to the source (\(I_{gcs}\)) and rest goes to the drain (\(I_{gcd}\)); and c) the gate-substrate current (\(I_{gsb}\)). The overlap tunneling current (\(I_{gdo}\) and \(I_{gso}\)) dominates gate leakage in the OFF state (\(V_{gs} = 0\)), whereas the gate-channel current becomes significant in the ON state (\(V_{gs} = 1\)). \(I_{gsb}\) is negligible compared to the overlap and the gate-channel tunneling [7].

Gate leakage was predicted to catch up with subthreshold leakage in scaled technologies. Hence it has drawn some attention recently on the modeling and reduction of it [8, 9]. In our study, we consider gate leakage is negligible in the presence of high-k material [10].

**Junction BTBT Leakage**

In deep sub-micro MOSFETs, highly doped “halo” regions are used in the substrate side of the drain-substrate and source-substrate junctions to suppress the SCE [7]. This increases the junction electric field and causes the significant tunneling of electrons from the valence band of the p-region, to the conduction band of the n-region. This leakage current is known as the junction BTBT current (\(I_{JN}\)). In the off-state of a transistor (for NMOS, \(V_g=0, V_d=V_{dd}, V_s=0, V_b=0\)), the junction tunneling results in a leakage current from the drain to the substrate. The junction BTBT leakage exponentially increases with an increase in the junction doping and the supply voltage, both of which increase the junction field. The junction BTBT in a MOSFET with halo doping can be expressed as:

\[
I_{JN} = W A \xi_{JN} V_{db} e^{(\frac{B E_g^3}{E_g^2})} \exp(\frac{B E_g^3}{\xi_{JN}})
\]

\[
\xi_{JN} = \sqrt{\frac{2q N_{halo} N_{halo}^2 (V_{db} + V_{bi})}{\xi_s i (N_{halo} + N_{halo}^2)}}
\]

where \(W\) is the width, \(\xi_{JN}\) and \(V_{bi}\) are the electric field and the built-in-potential at the drain-substrate junction, respectively; \(V_{DB}\) is the drain-substrate bias; \(A\) and \(B\) are the physical parameters; \(E_g\) is the band-gap; \(N_{halo}\) and \(N_{S/D}\) are the effective doping density in the substrate (halo doping) and source/drain (S/D) side of the junctions, respectively. It can be observed that the junction BTBT increases exponentially with an increase in the doping density (particularly higher halo doping) and the drain-substrate bias (\(|V_{DB}|\)) [7].
1.1.2 Temperature and Process Variation Impact

Among all three main leakage sources, subthreshold leakage has a strong dependency on temperature, since subthreshold current is governed by the carrier diffusion that increases with an increase of temperature. Because tunneling probability of an electron through a potential barrier does not depend directly on temperature, the gate and the junction band-to-band tunneling are less sensitive to temperature variations. However, increasing temperature reduces silicon's band gap, which is the barrier height for tunneling in BTBT. Hence, the junction BTBT should increase with an increase in temperature [11].

Figure 1.3 [11] shows the effect of temperature variation on each individual leakage component of the previously mentioned 25-nm NMOS device based on device simulations. In Figure 1.3, we observe that the subthreshold leakage increases exponentially with temperature, the junction BTBT increases slowly with temperature, and the gate leakage is almost independent of temperature variation. Hence, it can be concluded that the individual leakage components and the total leakage depend strongly on temperature (or mode of operation) [11].

Subthreshold leakage is a strong function of channel length, oxide thickness, supply voltage, and temperature. In addition, gate leakage depends on gate oxide thickness, channel length, and supply voltage level. Therefore, variation in leakage can occur due to a fluctuation of each of these variables, which is very severe in sub-100nm regimes. Borkar et al. [12] presents measurement results on the variation of frequency and leakage current on a wafer, as shown in Figure 1.4. As we can see, the maximum variation of leakage current due to process variation can be as high as 20 times. Due to these significant impacts, many researches have been conducted on the modeling and
optimization of leakage with the presence of temperature variation [13, 14, 15], as well as process variation [16, 17, 18].

1.2 Leakage Control in Nano-scale Technologies

Many leakage control techniques have been introduced and studied so far. They can be characterized into two classes: design-time techniques and runtime techniques. Design-time techniques include dual-threshold design (MTCMOS) [2], gate-length biasing [19], doping profile, or oxide thickness, channel length engineering etc. [11]. Design-time techniques mainly modify the circuit to minimize its leakage within a performance constraint. Their capacity is limited since the performance cannot be sacrificed. On the other hand, runtime techniques, such as power gating [3], reverse body biasing [4] and input vector control [5], tune the circuit into low-leakage mode according to the variation of circuit workload during runtime.

Many electronic systems are designed in a way that not all components in the system are working simultaneously. Besides, a system may not be working continuously due to the variation in the system workload. As we will discuss in Chapter 3, these two features yield the opportunities for runtime techniques to reduce the leakage power consumption when the system or part of the system is idle. When a system has substantial slackness in its workload, runtime techniques can yield significant leakage saving.
1.2.1 Current RTLC Techniques

Power Gating (Sleep Transistor Technique)

Power gating (PG) technique inserts an extra series-connected (sleep) high $V_{th}$ transistor between a circuit and its power supply or ground. When the extra transistor is turned off, leakage current will be reduced substantially due to the stacking effect. The extra transistor is on during normal operation. However, the extra stacked transistor makes the drive current of the forced-stack gates lower, resulting in slight performance penalty. Figure 1.5 shows an example of an inverter with ground gating. Only one type (either PMOS or NMOS) of high $V_{th}$ transistor is sufficient for leakage reduction. The NMOS insertion scheme is preferable, since the NMOS on resistance is lower than that for a PMOS at the same width. The NMOS (footer) can thus be smaller than a corresponding PMOS (header). In order to minimize the performance penalty, the area of the extra transistor is usually made to be large. This causes area penalty. Hence a large amount of existing researches studies the sizing problem of PG [62, 20, 21, 22, 23]. When the extra transistor switches, it causes large rush current into the power supply or ground. Hence there are also researches studying the ground bouncing problem of PG [24, 25, 26, 27].

![Figure 1.5: An Inverter With Ground Gating](image)

Instead of using high $V_{th}$ sleep transistors, a super cutoff CMOS (SCCMOS) circuit uses low $V_{th}$ transistors with an inserted gate bias generator [28]. In standby mode, the gate is applied to $V_{DD} + 0.4V$ for PMOS ($V_{SS} − 0.4V$ for NMOS) by using all the internal gate bias generator to fully cut off the leakage current. Compared to the high $V_{th}$ scheme, in which it becomes difficult to turn on the high $V_{th}$ sleep transistor at very low supply voltages, SCCMOS circuits can operate at very low supply voltages.

In our analysis for PG, since PG has minor impact on BTBT leakage [11], subthreshold leakage
is the only leakage source considered.

**Reverse Body Biasing**

Reverse body biasing (RBB), or variable-threshold CMOS (VTCMOS), changes the biasing voltage of transistor body to alter its $V_{th}$. Normal biasing voltage will be applied during circuit normal mode. When the circuit is idle, reverse biasing voltage will be applied to the transistor body to increase its $V_{th}$, and thus reduce the subthreshold leakage. As a side effect, BTBT leakage increases due to larger voltage differential between body and source/drain. Figure 1.6 illustrates the basic RBB scheme applied on an inverter. Keshavarzi et al. reported that RBB lowers IC leakage by three orders of magnitude in a 0.35um technology [29]. However, more recent data shows that RBBs effectiveness in lowering $I_{off}$ decreases as technology scales, because of the exponential increase in source-substrate and drain-substrate band-to-band tunneling leakage at the source-substrate and drain-substrate p-n junctions (because of halo doping in scaled devices). Moreover, the shorter channel lengths as technology scales and the lower channel doping (to reduce $V_{th}$) worsen SCE and diminish the body effect. This in turns weakens RBB’s $V_{th}$ modulation capability [11]. For scaled technologies, researchers have proposed using forward body biasing (FBB) to achieve better current drive with fewer SCEs [30].

Although the effectiveness of RBB is predicted to reduce in future technology generations, it is still an effective technique. As we will show in Chapter 4, it has several advantages, which make it especially suitable for RTLC. In our analysis for RBB, both subthreshold leakage and BTBT leakage will be considered.
**Input Vector Control**

Due to the stacking effect, the subthreshold leakage through a logic gate depends on the applied input vector. This makes the total leakage current of a circuit dependent on the states of the primary inputs. This feature can be used to for leakage reduction. The main research problem of input vector control is searching the best input vector to minimize the leakage current [6]. The most straightforward way is to enumerate all combinations of primary inputs. For a circuit with \(n\) primary inputs, there are \(2^n\) combinations for input states. Due to the exponential complexity with respect to the number of primary inputs, such an exhaustive method is limited to circuits with a small number of primary inputs. For large circuits, a random search-based technique can be used to find the best input combinations. This method involves generating a large number of primary inputs, evaluating the leakage of each input, and keeping track of the best vector giving the minimal leakage current [31]. A more efficient way is to employ the genetic algorithm to exploit historical information to speculate on new search points with expected improved performance to find a near-optimal solution [32].

As a runtime technique, input vector control has two drawbacks. First its leakage reduction rate is highly dependent on the circuit topology. Experimental results in [33] show that for some circuits, the reduction rate is less than 10%. The second drawback is that every time when input vector control is applied, it causes a large dynamic energy consumption overhead. Experimental results in [33] show that this overhead is much larger than the overhead for RBB and PG. Due to these two drawbacks, input vector control is not considered as a candidate technique for RTLC. We will not further analyze it in this proposal.

**1.2.2 Challenges for RTLC in Nano-scale Technologies**

The fundamental mechanism of RTLC is to put the circuit into low-leakage mode when the slackness in the workload causes idleness in the circuit. So the first challenge is how to maximally exploit circuit idleness. Idleness exploitation is not a new concept. It is extensively studied for dynamic power control [34]. Although the concept is similar, idleness exploitation for leakage power control is more difficult. As we will show in Chapter 4, it has two obstacles that do not exist in dynamic power control: the energy overhead problem and wake-up delay problem. Besides, as we will explain in Chapter 5, RTLC usually causes severe robustness problem, which is not a significant
problem for dynamic power control either. So a good solution for the first challenge should be able to address these three problems, meanwhile achieves the maximal idleness exploitation.

The second challenge is how to detect idleness in the circuit. Idleness detection in dynamic power control is an easy task, since we only need to identify the circuit, which will be idle in the next clock cycle. However for leakage power control, it usually needs to identify multiple cycles of idleness in the future. This turns out to be much more difficult than identifying single-cycle idleness. Chapter 6 will discuss the idleness detection at different abstraction levels.

Even we know how to detect and exploit idleness, the control decision made without considering temperature and process variation is not reliable. So the third challenge is how to factor in the impact of temperature and process variation. In general, for researches related with temperature and process variations, there are two approaches: design-time modeling or runtime monitoring. Design-time modeling approach has been extensively used for both temperature [13, 14, 15] and process [16, 17, 18] variations. Basically, it computes a theoretical range of target estimates, such as leakage current. Then the RTLC control decision can be made based on this range. However for leakage current, this range can be very large due to the strong impact of temperature and process variation on leakage. If the range is too large, the estimation becomes meaningless. On the other hand, runtime monitoring approach usually inserts a temperature or current sensor into the circuit. The sensed information is then used to assist runtime decision making. Recently, runtime monitoring techniques are mostly used in pose-silicon compensation for process variation. [35] is the first study to apply it to control RTLC. In Chapter 6, we will give more detailed discussion.

Figure 1.7 illustrates an ideal solution to cope with all three challenges. An idleness detection module receives workload information, circuit status information from a FSM, and information on temperature and process variation (either from design-time modeling or runtime monitoring). It evaluates all these information and issues control signals to the idleness exploitation module. The idleness exploitation module is drawn outside of the datapath and FSM for illustration purpose. In fact, it can be inserted into different point of datapath and FSM to maximize the idleness exploitation. Figure 1.7 depicts the desirable solution for RTLC. Among three challenges, we consider that the first challenge is the most complicated and fundamental problem of RTLC. Our investigation in this proposal centers around the first challenge, or the design of the idleness exploitation module in Figure 1.7. We also propose several methods as future works to tackle the second and the third challenges.
1.3 Contribution and Organization of This proposal

To the best of our knowledge, this proposal is the first attempt to systematically study the design methodology of runtime leakage control. As technologies moving to deep sub-micro, this topic is as important as, if not more than, the design methodology of dynamic power control. After dynamic power control, we consider this topic as the second major challenge in the history of DFP. The contributions of this proposal are as following:

1) We propose three major challenges for RTLC: how to maximally exploit idleness, how to detect idleness and how to factor in the impact of temperature and process variation.

2) We identify three major questions regarding how to maximally exploit idleness: how to design optimum control policy, what is the optimum granularity and how to reduce leakage in circuit active mode.

3) We derive two models for the circuit during PG and RBB mode transition. These two models can serve many purposes, such as energy breakeven time estimation, wake-up time estimation, ground bouncing estimation, and determination of optimum design point.

4) We propose the idea of temporal and spatial idleness exploitation with finer granularity, and use it to implement an algorithm to determine the optimum granularity for RTLC.

5) We quantitatively compare the effectiveness of using PG and RBB for RTLC and find out RBB is superior in many aspects.

6) We identify the reason that causes the gap between dynamic and leakage power control techniques. Based on that, we propose the idea of selective light \( V_{th} \) hopping (SLITH) to bridge the gap, and enable RTLC in circuit active mode.
7) We study the design methodology for fast power gating wake-up with guaranteed power integrity. Two novel techniques, namely current shaping and multi-thread activation are proposed.

8) We systematically analyze the leakage control potentials at microarchitectural level, and propose a comprehensive scheme to maximize leakage control with an adaptive feature for TV/PV compensation.

Among these contributions, the idea of temporal and spatial idleness exploitation with finer granularity can be used out of the scope of this study. It can be used as a general method of idleness detection and utilization for other systems.

This proposal is organized as following. Chapter 2 to 4 focus on answering the three major questions of the first challenge. Chapter 2 contains the modeling process for PG and RBB and answers the question of the optimum control policy. Chapter 3 explains the idea of temporal and spatial idleness exploitation, and answers the question of the optimum granularity. Chapter 4 compares the PG and RBB, and proposes the selective light $V_{th}$ hopping (SLITH) scheme to enable RTLC in circuit active mode. Chapter 5 proposes a physical design approach to ensure realizability on top of all the leakage control techniques. Chapter 6 proposes a microarchitectural level method with TV/PV compensation to tackle the second and third challenge at the same time. Finally Chapter 7 concludes the thesis and proposes several future work directions.
Chapter 2

Optimum Control Policy for Run-time Leakage Control

Power gating (PG) and reverse body biasing (RBB) are effective RTLC techniques. But their application during runtime imposes two problems:

1) Each time when RTLC is applied, the energy overhead for mode transition is incurred. If RTLC is applied too frequently, this overhead can exceed the leakage energy saving, and thus cause the net energy saving to be negative.

2) When a circuit is in RTLC mode, it takes time for the circuit to recover to the normal operational mode. Thus if the circuit receives workloads before it has fully recovered, the circuit may not function properly.

Because of these two problems, the control policy design of RTLC is important. An optimum control policy guarantees that the above two problems are taken care of, meanwhile the leakage reduction is maximized. In this section, we will show the current solution of control policy design and our contribution to this problem.

This chapter is organized as follows. Section 2.1 gives the definition for energy breakeven time and wake-up time, and discusses the study of them in current literatures. Section 2.2 derives an accurate energy breakeven time model for PG. Section 2.3 derives an accurate energy breakeven time model for RBB. Finally, Section 2.4 concludes the chapter.
2.1 Energy Breakeven Time and Wake-up Time

2.1.1 Definitions

In order to quantitatively study the first problem, the concept of energy breakeven time (EBT) [36], or equivalent sleep time [37], or minimum idle time [33]) has been proposed. Similarly, to quantitatively study the second problem, the concept of wake-up time (WUT) has been proposed [37].

Precisely, assume that at time $t$ after RTLC is applied, the leakage energy saving of the circuit is $E_S(t)$. And the energy overhead for mode transition is $E_P$. Then EBT ($T_B$) is defined as down-time of the circuit, at which leakage energy saving catches up with the energy overhead:

$$E_S(T_B) = E_P$$ (2.1)

Thus if the circuit stays in the RTLC mode for a period longer than EBT, energy saving can be archived. The first problem can be solved if the control policy makes sure that RTLC is only applied when the circuit idle time $T_{idle}$ is larger than EBT:

$$T_{idle} > T_B$$ (2.2)

On the other hand, WUT ($T_W$) is simply defined as the time required for the circuit to recover from RTLC mode to normal mode. Thus in order to solve both the first and the second problems, the control policy needs to make sure that RTLC is only applied when $T_{idle}$ satisfies:

$$T_{idle} > T_B + T_W$$ (2.3)

Notice that we will show later that WUT is generally small. So here we assume that during the mode transition from RTLC to normal mode, the circuit does not yield any energy saving. Hence $T_B$ and $T_W$ does not have any overlap so that they can be summed up in Equation 2.3.

Both EBT and WUT are important design parameters for RTLC. In [37], Tschanz et al. measured that for a ALU module in 130nm technology, the EBT and WUT for PG are around 80 and 2 clock cycles, respectively. The EBT and WUT for RBB are around 100 and 4 cycles, respectively. We can see that EBT is much larger than WUT and becomes the major problem for control policy design. So in the following content of this section, we will focus on the study of EBT.
2.1.2 Existing Approaches for EBT Estimation

Leakage has high dependency on circuit topology, input vectors and technologies used. So the EBT value varies for different circuits in different technologies. It needs to be modeled for designing the control policy of RTLC. We first look at the EBT modeling of PG. A simple and commonly used EBT model of PG is presented in [1]:

\[ T_B = \frac{E_p}{P_{active} - P_{sleep}} \]  

(2.4)

where \( P_{active} \) and \( P_{sleep} \) are the leakage power consumption in normal mode and PG mode, respectively. This expression assumes that the leakage current is a constant (\( I_{min} \)) after the PG is applied. According to our introduction in Section I, this assumption is not true. In fact after PG is applied, the leakage current gradually goes through a variation period and finally settles down at \( I_{min} \) at time \( T_{steady} \). So the assumption in [1] introduces error when estimating leakage current, energy saving and thus EBT, as shown in the shade area in Figure 2.1.

For PG used in standby mode, since the circuit sleep time is long (\( T_{idle} >> T_{steady} \)), this error is negligible. However more aggressive leakage reduction necessitates more accurate EBT modeling. In this case, this error is not negligible. Some recent researches have tried to correct this error by considering the variation of leakage current before \( T_{steady} \). Yu et al. [38] multiply the total leakage energy saving with an empirical value (0.73) to cover the variation. In [39] and [36], Hu et al. and Usami et al. derive EBT models factoring in the variation. However, their models do not consider the impact of circuit topologies and circuit states. Since leakage has a strong dependency on the circuit states [11], the models in [39] and [36] are rather high level models and cannot be used when accurate estimates are necessary.

Similarly for RBB, as we have shown in Section I, the leakage reduction after applying RBB is again a gradual and dynamic process. So in conclusion, the following two problems needs to be
resolved in order to derive an accurate EBT model, either for PG or RBB.

1) The leakage variation before $T_{steady}$ needs to be modeled.

2) The circuit topology and states need to be considered.

So far these two problems have not been resolved in any literatures. In the following content of this chapter, we focus on seeking the solution of these two problems, so that to derive two accurate EBT models for PG and RBB, and thus determine the optimum control policy.

2.2 Energy Breakeven Time Modeling for Power Gating

In this section, we first derive the leakage current reduction model for any given circuit topology. Once the variation of leakage current is obtained, we derive a single-state EBT model. Finally considering all possible circuit states, we derive a multiple-sate EBT model.

In our research, we only study the ground power gating, since it is widely adopted in most design practices. Supply power gating can be studied with a similar method. Also, we consider the sub-threshold leakage current as the major leakage source, since the gate leakage is projected to have a reduction of $100 \times$ with the use of high-k dielectrics [10]. Additionally, due to the exponential dependency of the sub-threshold leakage on temperature, the sub-threshold leakage is still the dominant leakage component during runtime. So we focus on the sub-threshold leakage and refer to it as leakage.

2.2.1 Leakage Current Reduction Model

We derive leakage current reduction (LCR) model from transistor level to circuit level.

Leakage Current Reduction Modeling At Transistor Level

We start from the full-fledged subthreshold leakage current model for a single transistor [6]:

$$I = A \cdot e^{1/mV_T(V_G - V_S - V_{th} - \gamma'V_S + \eta V_{DS})} \cdot (1 - e^{-V_{DS}/v_T})$$

with $A = \mu_0 C_{ox}' \frac{W}{L_{eff}} (v_T)^2 e^{1.8} e^{-\Delta V_{th}/\eta v_T}$ \hspace{1cm} (2.5)
where $V_{th0}$ is the zero bias threshold voltage, $v_T$ is the thermal voltage, $\gamma'$ is the linearized body effect coefficient, and $\eta$ is the DIBL coefficient. When $V_{DS} \gg v_T$, the term $(1 - e^{-V_{DS}/v_T})$ in the above equation can be neglected. Assuming that the temperature and the body bias voltage are fixed, we can simplify Equation 2.5 into:

$$I = \hat{I} \cdot e^{K_1 V_G + K_2 V_D - K_3 V_S}$$

(2.6)

where $K_1$, $K_2$ and $K_3$ are technology-dependant exponents, and $\hat{I}$ is the leakage current under the normal condition terminal voltages. Note that the simplification of the term $(1 - e^{-V_{DS}/v_T})$ in Equation 2.5 causes inaccuracy when $V_{DS}$ is comparable with $v_T$. (When $V_{DS}$ equals to $4v_T$, the error is 1.8\%.) Hence our model is accurate when $V_{DS} > 4v_T$. When $V_{DS} < 4v_T$, the leakage current is very small such that it can be approximated as zero in EBT calculation.

Now apply ground gating to a single transistor. With ground gating, the terminal voltages of the off-state NMOS or PMOS have the patterns as shown in Figure 2.2. Note that $V_{VG}$ stands for virtual ground voltage.

The leakage current of them can be expressed as:

$$\begin{align*}
I_N &= \hat{I} \cdot e^{K_1 V_{VG} + K_2 V_D - K_3 V_S} \\
V_G &= V_{VG} (NMOS) \\
V_D &= V_{DD} (NMOS) \\
V_S &= V_{DD} (NMOS)
\end{align*}$$

(2.7)

Solving the above equations yields:

$$\begin{align*}
I_N &= \hat{I} \cdot e^{K_1 V_{VG} + K_2 V_{DD} - K_3 V_{VG}} = \hat{I}_N \cdot e^{-K_N V_{VG}} \\
I_P &= \hat{I} \cdot e^{K_1 V_{DD} + K_2 V_{VG} - K_3 V_{DD}} = \hat{I}_P \cdot e^{-K_P V_{VG}}
\end{align*}$$

(2.8)

where $\hat{I}_P (\hat{I}_N)$ is the leakage current of PMOS (NMOS) with zero $V_{VG}$, and $K_P (K_N)$ is the leakage reduction exponent of PMOS (NMOS). So a ground gated off-state transistor can be modeled as a
$V_{VG}$-controlled current source. The footer has a similar model since essentially it is a NMOS transistor with high threshold voltage.

Next, we study the LCR of a ground gated NMOS. As soon as the footer is turned off, the leakage current starts to charge the internal nodes. As a result, the $V_{VG}$ level gradually rises. Consequently, the leakage current is reduced according to Equation 2.8. As shown in Figure 2.3, we model this as a charging process of the virtual ground capacitor by $V_{VG}$-controlled current sources. The resistance of on-state transistors and interconnects does not affect the charging process, since the charging is due to an equivalent current source. The resistance is then neglected in our model. We have designed an experiment in Section IV to verify this.

![Figure 2.3: LCR Modeling of A Ground Gated NMOS](image)

By considering $V_{VG}$ and leakage current as functions of time, the charging process can be characterized as:

\[
\begin{align*}
V_{VG}(t) &= \frac{1}{C} \int_0^t I_{\text{charge}}(t) dt \\
I_{\text{charge}}(t) &= I_{\text{NMOS}}(t) - I_{\text{footer}}(t) \\
I_{\text{NMOS}}(t) &= \hat{I}_N \cdot e^{-K_NV_{VG}(t)} \\
I_{\text{footer}}(t) &= \hat{I}_F \cdot e^{-K_FV_{VG}(t)}
\end{align*}
\]

(2.9)

where $t$ is the time after the ground is gated, and $C$ is the equivalent capacitor attached to the virtual ground. In this case, $C$ includes the junction capacitances of the NMOS and the footer, and gate capacitance of the NMOS. $\hat{I}_F$ is the footer leakage current when $V_{VG}$ equals to $V_{DD}$. $K_F$ is the leakage reduction exponent of the footer. In the above equations when the $V_{VG}$ level is low, the leakage current of the footer is very small and can be ignored. (It will be considered in the circuit level model.) Equations 2.9 can then be solved as:

\[
\begin{align*}
V_{VG}(t) &= \frac{1}{K_N} \ln \left( \frac{K_N\hat{I}_N}{C} \left( t + \frac{C}{K_NI_N} \right) \right) \\
I_{\text{NMOS}}(t) &= \frac{C}{K_N(t + \frac{C}{K_NI_N})}
\end{align*}
\]

(2.10)
The above equations give the transistor level LCR model.

**Leakage Current Reduction Modeling At Gate Level**

In the following, we derive the LCR model at the gate level.

1) Example of Two-input NAND Gate

Consider a two-input NAND gate. Since the input vectors have significant impacts on the leakage current, we model the gate by each input vector. As shown in Figure 2.4, the two-input NAND gate is modeled into four $V_{VG}$-controlled leakage current sources ($I_i$), with equivalent capacitances ($C_i$) attached to the virtual ground of the gate. The next question is to find out the relationship between the leakage current ($I_i$) of the gate and the $V_{VG}$. Since the off-state transistors control the amount of leakage current, our answer starts from studying two basic gate structures: off-state transistors in series and parallel.

2) Off-state Transistors In Series

Case four in Figure 2.4 is the simplest series structure with two off-state NMOS transistors in stack. Our first goal is to find out the impact of stacking effect on the leakage current. Consider a general case of transistors in series as shown in Figure 2.5a. Assume that the upper terminal voltage of the four-transistor stack is $V_a$, and the virtual ground voltage is $V_{VG}$. By using Equation 2.6, the leakage current for each transistor is given by:
where $I_{\text{series}}$ is the leakage current of the stack. By substituting $\tilde{I}_i (i=1,2,3,4)$ with $e^{\tilde{I}_i}$, Equations 2.11 turns into:

\[
\begin{align*}
I_1 &= \tilde{I}_1 e^{K_1 V_{VG} + K_2 V_a - K_3 V_1} = I_{\text{series}} \\
I_2 &= \tilde{I}_2 e^{K_1 V_{VG} + K_2 V_1 - K_3 V_2} = I_{\text{series}} \\
I_3 &= \tilde{I}_3 e^{K_1 V_{VG} + K_2 V_2 - K_3 V_3} = I_{\text{series}} \\
I_4 &= \tilde{I}_4 e^{K_1 V_{VG} + K_2 V_3 - K_3 V_{VG}} = I_{\text{series}}
\end{align*}
\]

(2.11)

Solving the above equations yields:

\[
\begin{align*}
K_2 V_a - K_3 V_1 + \tilde{I}_1 &= \ln(I_{\text{series}}/e^{K_1 V_{VG}}) \\
K_2 V_1 - K_3 V_2 + \tilde{I}_2 &= \ln(I_{\text{series}}/e^{K_1 V_{VG}}) \\
K_2 V_2 - K_3 V_3 + \tilde{I}_3 &= \ln(I_{\text{series}}/e^{K_1 V_{VG}}) \\
K_2 V_3 - K_3 V_{VG} + \tilde{I}_4 &= \ln(I_{\text{series}}/e^{K_1 V_{VG}})
\end{align*}
\]

(2.12)

With the stacking effect, the leakage current still has an exponential dependency on $V_{VG}$. The difference is that the leakage reduction exponent turns into an equivalent exponent $K_S$, and the zero-$V_{VG}$ leakage current of the stack turns into $\tilde{I}_{\text{series}}$. Now consider a special case where one of the transistors in the stack is on as shown in Figure 2.5b. Assume that the voltage difference between $V_1$ and $V_2$ is negligible. This case is then equivalent to a three-transistor stack. Hence if some transistors in the stack are in the on-state, the total leakage current of the stack still satisfies Equation 2.13.
3) Off-state Transistors In Parallel

Case one in Figure 2.4 is the simplest example of off-state transistors in parallel. Since two off-state PMOS have the same leakage reduction exponent \( K_P \), the total leakage remains an exponential function, despite of the transistors size. Similarly, for a parallel structure with only one transistor in each branch (Figure 2.6a), the total leakage is an exponential function of \( V_{VG} \). The parallel structure can be complex when a branch has more than one transistor. Due to the stacking effect, it can have different \( K \) values depending on the number of off-state transistors in that branch. For example in

\[
I_{parallel} = \hat{I}_a e^{-K_a V_{VG}} + \hat{I}_b e^{-K_b V_{VG}} + \hat{I}_c e^{-K_c V_{VG}} + ... \tag{2.14}
\]

where \( K_i \) (\( i = a, b, c \)) is the equivalent leakage reduction exponent of each branch. Since \( K_i \) can be different, the total leakage current is no longer a simple exponential function. However, we still approximate \( I_{parallel} \) into a lumped exponential function of \( V_{VG} \) because of two reasons:

a) Most gates have the simple parallel structure shown in Figure 2.6a, for example, NAND gates, NOR gates and Buffers.

b) For complex parallel structures, the \( K \) values can be different because the number of off-state transistors in each branch is different. In this case, due to the stacking effect, the branches with the least number of off-state transistors (branches \( b \) and \( c \) in Figure 2.6b) have much larger leakage current than other branches (at least one order of magnitude \([6]\)) and become the dominant ones. Then \( I_{parallel} \) is close to the sum of dominant branches current. Since the dominant branches current have the same \( K \), \( I_{parallel} \) can be approximated as an exponential function:

\[
I_{parallel} = \hat{I}_b e^{-K_b V_{VG}} + \hat{I}_c e^{-K_c V_{VG}} \approx (\hat{I}_b + \hat{I}_c) e^{-K_b V_{VG}} \quad (K_b = K_c) \tag{2.15}
\]

![Figure 2.6: Off-state Transistors In Parallel](image-url)
We have shown that the leakage current of any number of off-state transistors in series is an exponential function of $V_{VG}$. And in parallel, it can be approximated into an exponential function. So for any type of gate structure, we construct a lumped exponential function of $V_{G}$ to model the total leakage current of the gate. Conclusively, we define the full-fledged gate model as follows. For each input vector $i$, the gate is modeled as an equivalent virtual ground capacitor $C_i$ and a $V_{VG}$-controlled current source $I_i$ satisfying:

$$I_i = \tilde{I}_i e^{-K_i V_{VG}}$$  \hspace{1cm} (2.16)

where $\tilde{I}_i$ is the zero-$V_{VG}$ leakage current, and $K_i$ is the equivalent reduction exponent of the gate. Use this model to substitute the NMOS model in Equations 2.9 and solve them in a similar way. The gate level LCR model can be obtained.

**Leakage Current Reduction Modeling at Circuit Level**

At the circuit level after the ground is gated, each gate in the circuit can be modeled as a $V_{VG}$-controlled current source with a capacitor attached to the virtual ground of the circuit, as shown in Figure 2.7. Taking the footer leakage current into account, we derive the circuit level model by 3 steps.

![Figure 2.7: LCR Modeling At Circuit Level](image)

a) For those gates ($h$), which have the same gate type ($j$) and receive the same input vector ($i$), they can be linearly combined into a single current source ($I_{ij}$), with a total equivalent capacitor ($C_{ij}$). Hence in this step, we build a downsized circuit model with maximally $j2^i$ current sources.

$$\begin{align*}
I_{ij} &= (\sum_h \tilde{I}_{ijh}) e^{-K_{ij} V_{VG}} \\
C_{ij} &= \sum_h C_{ijh}
\end{align*}$$  \hspace{1cm} (2.17)
b) Now the total charging current to the virtual ground of the circuit turns into the summation of all $I_{ij}$ of each gate in the circuit, subtracted by the footer leakage current:

$$
\begin{cases}
I_{\text{charge}} = I_{\text{circuit}} - I_{\text{footer}} \\
I_{\text{circuit}} = \sum_{ij} \hat{I}_{ij} e^{-K_{ij}V_{VG}} \\
I_{\text{footer}} = \hat{I}_F e^{-K_FV_{VG}}
\end{cases}
$$

(2.18)

The total virtual ground capacitor of the circuit is a linear summation of all the equivalent capacitors $C_{ij}$. By considering the $V_{VG}$ and the current as functions of time, the LCR process of the circuit can be characterized as:

$$
\begin{cases}
V_{VG}(t) = \frac{1}{C_{\text{total}}} \int_0^t I_{\text{charge}}(t) dt \\
I_{\text{charge}}(t) = \sum_{ij} \hat{I}_{ij} e^{-K_{ij}V_{VG}(t)} - \hat{I}_F e^{-K_FV_{VG}(t)} \\
C_{\text{total}} = \sum_{ij} C_{ij}
\end{cases}
$$

(2.19)

The above equations do not have a closed-form solution. To address this issue, we construct a piecewise linear model for the leakage current of each gate, as well as the footer. In detail, we first divide the full VDD into $R$ consecutive voltage regions ($< V_{rl}^r, V_{rh}^r >$), where $V_{rl}^r$ ($V_{rh}^r$) represents the low (high) bound voltage of region $r$. In each region $r$ ($r = 1..R$), we perform linear regression on the exponential leakage current models in Equations 2.18. We then have:

$$
\begin{cases}
I_{ij}^r = -S_{ij}^r V_{VG} + Z_{ij}^r \\
I_{\text{footer}}^r = -S_F^r V_{VG} + Z_F^r
\end{cases}
$$

(2.20)

where $S_{ij}^r$ and $Z_{ij}^r$ are the linearized coefficients for gate level leakage current models in each region $r$, and $S_F^r$ and $Z_F^r$ are the linearized coefficients for the footer leakage model. Linear regression incurs error $D_{ij}^r$ ($D_F^r$) to the leakage current models. It can be alleviated by increasing the $R$ value. However, the computation complexity also increases when $R$ is larger. So the selection of $R$ is a trade-off between accuracy and speed. In Algorithm I, we set an error threshold ($D_{th}$) to determine $R$.

c) Once we obtain the linearized leakage current models, we use them to substitute all exponential leakage current models in Equations 2.19. Then the total charging current also transforms into a piecewise linear model:

$$
I_{\text{charge}}^r = \sum_{ij} (-S_{ij}^r V_{VG} + Z_{ij}^r) - (-S_F^r V_{VG} + Z_F^r)
$$

$$
\begin{align*}
&= (-S_e^r V_{VG} + Z_e^r) - (-S_F^r V_{VG} + Z_F^r) \\
&= -S_e^r V_{VG} + Z_e^r
\end{align*}
$$

(2.21)
where $S_r^r$ and $Z_r^r$ are the linearized model coefficients for the total leakage current of the circuit in each voltage region $r$, and $S^r$ and $Z^r$ are the linearized model coefficients for the total charging current to the virtual ground. Figure 2.8 illustrates the transform in Equation 2.21. Substitute the total charging current in Equations 2.19 with the above linearized model. Make $V_{VG}$ and the charging current as functions of time. Solving it yields:

\[ \begin{cases} V_{VG}^r(t) = \frac{Z_r^r}{S_r^r} + \frac{S^r V_{h}^r - Z^r}{S^r} e^{-\frac{S^r}{C_{total}} (t-\tau^r)} \\ I_{charge}^r(t) = I_0^r e^{-\frac{S^r}{C_{total}} (t-\tau^r)} \end{cases} \tag{2.22} \]

where $I_0^r$, $V_0^r$ and $\tau^r$ are the initial charging current, initial voltage and initial time of the region $r$, respectively. The initial voltage and time for region 1 are both zero. The initial charging current of region 1 is the original leakage current without ground gating. In region $r$ ($r=2..R$), these three initial conditions can be obtained by solving the following equations in region $r-1$ recursively:

\[ \begin{cases} V_{VG}^{r-1}(\tau^r) = V_h^{r-1} \\ I_0^r = I_{charge}^{r-1}(\tau^r) \\ V_0^r = V_h^{r-1} \end{cases} \tag{2.23} \]

Finally, the total leakage current of the circuit is given by:

\[ I_{ckt}^r(t) = -S_c^r V_{VG}^r(t) + Z_c^r \tag{2.24} \]

It gives the piecewise LCR model of the whole circuit.

**Secondary Physical Phenomena**

Some other secondary physical phenomena will occur during the leakage reduction process. For example in the previous study, the equivalent capacitance attached to the virtual ground is considered
to be a constant. However, it is shown that the gate capacitance of a transistor has a dependency on its $V_{GS}$[40], which is a changing value during the $V_{VG}$ charging process. So our model needs to take this phenomenon into consideration. To address this problem, we obtain an equivalent capacitor $C_{total}^r$ in each voltage region $r$, and use it to replace the $C_{total}$ in Equations 2.22.

### 2.2.2 Energy Breakeven Time Modeling for Power Gating

In this section, we first derive the EBT model for the circuit, which has only one specific state when entering the sleep mode. Secondly, we give the definition of average EBT for the circuit during runtime, with multiple possible states. Lastly, we develop a method to estimate the average EBT.

#### EBT Modeling for Circuit In A Specific State

Based on the piecewise LCR model in Equation 2.24, we have the energy saving model $(E_r(t)^r)$ in each region $r$:

$$E_{\text{piece}}^r(t) = \int_{\tau^r}^{t} V_{DD} (I_{ckt0} - I_{ckt}^r(t)) dt$$  \hspace{1cm} (2.25)

where $I_{ckt0}$ is the original leakage current of the circuit without power gating. It can be obtained by applying Equation 2.24 in voltage region 1 at time zero:

$$I_{ckt0} = -S^1_c V_{VG}^1(0) + Z^1_c = Z^1_c$$  \hspace{1cm} (2.26)

$E_{\text{piece}}^r(t)$ represents the energy saving only in region $r$, until time $t$. The total energy saving $(E^r(t))$ of all previous regions until time $t$ is:

$$E^r(t) = \sum_{m=1}^{r-1} E_{\text{piece}}^m + E_{\text{piece}}^r(t)$$  \hspace{1cm} (2.27)

where,

$$E_{\text{piece}}^m = \int_{\tau^m}^{\tau^{m+1}} V_{DD} (I_{ckt0} - I_{ckt}^m(t)) dt$$  \hspace{1cm} (2.28)

where successive region $m$ spans from $\tau^m$ to $\tau^{m+1}$. $E_{\text{piece}}^m$ then represents the energy saving in region $m$ from the beginning to the end of the region, as shown in Figure 2.9. So the summation of all $E_{\text{piece}}^m$ from 1 to $r-1$ represents the total energy saving until region $r-1$. 

\[27\]
Equation 2.27 gives the piecewise total energy saving model. Based on it, we are able to solve the EBT ($T_B$) by matching the energy saving with energy penalty ($E_P$):

$$E^r(T_B) = E_P$$ (2.29)

The energy penalty is the dynamic power consumption for switching the gate capacitance ($C_{\text{footer}}$) of the footer:

$$E_P = C_{\text{footer}} V_{DD}^2$$ (2.30)

EBT can be solved by putting Equations 2.22, 2.24, 2.26, 2.27 and 4.5 together:

$$\begin{cases} 
  \int_{\tau_r}^{t} (Z_c^1 - t_{\text{ckt}}(t)) dt = \frac{E_P - \sum E_{\text{piece}}}{V_{DD}} \\
  t_{\text{ckt}}(t) = -S_c^r V_{VG}^r(t) + Z_c^r \\
  V_{VG}^r(t) = \frac{Z_c^r}{S_r} + \frac{S_r V_{0}^r - Z_r^r}{S_r} e^{-\frac{S_r}{C_{\text{total}}}(t-\tau_r)}
\end{cases}$$ (2.31)

Simplifying the above equations yields:

$$a_1 e^{a_2(t-\tau_r)} + a_3 t + a_4 = 0$$ (2.32)

where $a_1$ to $a_4$ are the constants given by $Z_r^r$, $S_r^r$, $Z_c^r$, $S_c^r$, $V_{0}^r$, $C_{\text{total}}^r$, $Z_c^1$ and $E_P$. The above equation leads to no closed-form solution. Once again, we use linear regression to approximate it.

In detail, assuming that EBT happens in voltage region $\theta$ (Figure 2.9), linear regression will be performed on the leakage current model $I_{\text{ckt}}^\theta(t)$ and yields:

$$I_{\text{ckt}}^\theta(t) = -Ft + G$$ (2.33)

Linear regression on $I_{\text{ckt}}^\theta(t)$ is only needed in region $\theta$, instead of every region. This evolves the determination of $\theta$. It will be explained in detail in Section III.C.

Substitute the $I_{\text{ckt}}^\theta(t)$ in Equation 2.31 with 2.33 and set $r$ as $\theta$. Simplifying them yields:

$$A(T_B - \tau^\theta)^2 + B(T_B - \tau^\theta) - C = 0$$
Solving it yields:

\[ T_B = \tau^\theta + \sqrt{\frac{B^2}{4A^2} + \frac{C}{A} - \frac{B}{2A}} \]  

(2.35)

The above equation gives the EBT model for the circuit under a specific state.

**Average EBT Definition For Multiple States Circuit**

So far, we have modeled the EBT with the assumption that the circuit enters the sleep mode with one specific state. However, during runtime, the circuit may be in different states when the footer is turned off. Since in different states, the leakage power consumption can vary significantly [6], our EBT estimated for one state may not be accurate for another state, as shown in Figure 2.10.

![Figure 2.10: EBT Variation For The Circuit Sleeping In Different States](image)

So our next question is: Is there an average EBT value \((T_U)\) for the circuit, which has \(U\) possible states when entering the sleep mode?

Assuming that \(O_u\) is the number of occurrences for the circuit sleeping under state \(u\), our average EBT definition is:

\[ \sum_u O_u E^r_u(T_U) = E_P \sum_u O_u \]  

(2.36)

\(E^r_u(t)\) is our piecewise energy saving model for the circuit sleeping under state \(u\). \(E^r_u(T_U)\) is the energy saving at time \(T_U\). By multiplying \(E^r_u(T_U)\) with its corresponding \(O_u\) and summing them up, the left side of the above equation is the overall energy saving of all sleep occurrences, at time \(T_U\). The right side is the total energy penalty. In some states, the energy saving \(E^r_u(T_U)\) will be more than its penalty \(E_P\), and in other states less. However, the definition of the average EBT guarantees that the overall saving compensates the penalty. Equation 2.36 can also be expressed by
the probability \( P_u \) of the occurrences of each state:

\[
\sum_u P_u E^r_u(T_U) = E_P
\]  

(2.37)

We use the above probability expression in the following study.

**Average EBT Estimation**

We develop the average EBT estimation method in 4 steps:

a) Build piecewise LCR model \( I^r_u(t) \) for each state \( u \) in each region \( r \), using Equation 2.24. Build piecewise energy saving model \( E^r_u(t) \) using Equation 2.27. Also, calculate the initial time \( \tau_u^r \) using equation 2.23.

b) Denote overall energy saving of every sleep occurrence at time \( t \) as \( \bar{E}^r(t) \). By the average EBT definition, we have:

\[
\bar{E}^r(t) = \sum_u P_u E^r_u(t)
\]  

(2.38)

So we need to obtain the overall energy saving as a function of time. However, the piecewise energy model \( E^r_u(t) \) cannot be simply summed up by the voltage region \( r \). This is because each \( E^r_u(t) \) may have different time range \( (\tau_u^r, \tau_u^{r+1}) \) for the same \( r \). For example in Figure 2.11, the individual energy saving of the three states \( E_1^1(t), E_2^1(t), E_3^1(t) \) in voltage region 1 can not be summed up, because they have different ending time.

We create time regions to address this issue. In detail, we sort all the initial time \( \tau_u^r \) into a low-to-high list. Between every two elements in the list, create a time region \( w = 1..W, W = RU \), as shown in Figure 2.11.

![Figure 2.11: Create Time Regions To Calculate Overall Energy Saving](image)

Thus, we can sum up \( E^w_u(t) \) by the time region \( w \) and build a piecewise overall energy saving.
mode $\overline{E^w}(t)$:

$$\overline{E^w}(t) = \sum_u P_u \overline{E^w_u}(t) \quad (2.39)$$

c) By starting from the first region, we traverse through each time region $w$. In each $w$, calculate the overall energy saving ($\overline{E^w}$) until the end time of $w$. This can be achieved simply by calculating $\overline{E^w}(t)$ at time $\tau^{w+1}$:

$$\overline{E^w} = \overline{E^w}(\tau^{w+1}) \quad (2.40)$$

Assume that in time region $\theta$, the overall saving catches up with the penalty:

$$\overline{E^\theta-1} < E_P \leq \overline{E^\theta} \quad (2.41)$$

It means that the average EBT occurs in $\theta$. Similar to the single state EBT, we perform linear regression on the $I^\theta_u(t)$ for each state $u$. It yields:

$$I^\theta_u(t) = -F_u t + G_u \quad (2.42)$$

d) Now, we can solve the average EBT in time region $\theta$. By the average EBT definition, and using the linearized circuit leakage current model in Equation 2.42, we have:

$$\begin{cases}
\overline{E^\theta}(T_U) = E_P \\
\overline{E^\theta}(t) = \overline{E^\theta-1} + \sum_u P_u \int_{\tau^\theta}^t (Z^1_u - I^\theta_u(t))dt \\
I^\theta_u(t) = -F_u t + G_u
\end{cases} \quad (2.43)$$

Solving the above equations yields:

$$T_U = \tau^\theta + \sqrt{\frac{B^2}{4A^2} + \frac{C}{A} - \frac{B}{2A}} \quad (2.44)$$

where

$$\begin{cases}
A = \sum_u \frac{1}{2} P_u F_u \\
B = \sum_u P_u (Z^1_u - G_u) \\
C = \frac{E_P - \overline{E^\theta-1}}{V_{DD}}
\end{cases}$$

Equation 2.44 gives the final solution of the average EBT.

### 2.2.3 Algorithm and Experimental Results

In this section, we design an algorithm for average EBT estimation. Based on the algorithm, we develop a CAD tool to automatically estimate the average EBT for any given circuit. Then we
conduct experiments to compare the estimates of our CAD tool with the HSPICE simulation results. The ISCAS85 benchmark circuits in 32nm, 45nm and 65nm technologies [41] are used in the experiments. The gate level implementations are from [42]. We insert footers into the benchmark circuits to implement the ground gating. Since footer sizing does not affect the correctness of our model, it is designed to be equal to the total width of all the parallel NMOS in the circuit for simplicity. The simulation temperature is set to be 110°C to emulate the runtime temperature. The gate leakage is set to be zero.

**Algorithm for Average EBT estimation**

The whole solving process of the average EBT can be described in Algorithm I.

**Algorithm 1: Average EBT Estimation**

```
INPUT: Cell Library, Technology, Circuit netlist; $P_u, D_{th}$
OUTPUT: Average EBT of the circuit

---Library Characterization---
Derive $I_{ij}$, $K_{ij}$, $C_{ij}$ from Technology and Cell Library;
Choose initial $R$;
while $MAX(D_{r_{ij}}, D_{rf}) > D_{th}$ do
    Increase $R$;
    LinearRegression yields $S_{r_{ij}}, Z_{r_{ij}}, D_{r_{ij}}, S_{rf}, Z_{rf}, D_{rf}$;
end while

---LCR Modeling---
foreach $u=1$ To $U$ do
    foreach $r=1$ To $R$ do
        Calculate $S_{r_{cu}}, Z_{r_{cu}}, S_{r_c}, Z_{r_c}, C_{r_{total}}$;
        Build piecewise LCR model $I_u^c(t)$, energy saving model $E_u^c(t)$;
        Calculate initial time $\tau_{r_u}^c$;
    end foreach
end foreach

---Average EBT Estimation---
Sort $\tau_{r_u}^c$ and create time regions $w$;
Build piecewise overall energy saving model $E^c_w(t)$;
$w=1$;
while $E^w < E_P$ do
    $w = w + 1$;
    Calculate overall energy saving $E^w$ until $w$;
end while
Find average EBT time window $\theta = w$;
LinearRegression on $I_0^u(t)$ yields $F_u, G_u$;
Solve average EBT $T_u$ in $\theta$;
```
### Experiments on Resistance Impact to The Model

To verify the resistance impact on the model, we replaced all the original parasitic resistances in each circuit by 5 (30) times of their own values. Then the simulated $E_{exp}(n)$ of the new circuit, with larger resistances is compared with the simulated $E_{exp}(n)$ of the original circuit. As shown in Table I, the larger resistance causes a maximal variation of 1.48% to the $E_{exp}(n)$ in all three technologies. This experiment proves that resistance can be neglected in the model.

#### Table 2.1: Resistance Impact On Time-varying Leakage Energy

<table>
<thead>
<tr>
<th>BM. Cnts.</th>
<th>Gate</th>
<th>32nm 5x 30x</th>
<th>45nm 5x 30x</th>
<th>65nm 5x 30x</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>160</td>
<td>0.20% 0.14%</td>
<td>0.03% 0.53%</td>
<td>0.21% 0.74%</td>
</tr>
<tr>
<td>C1355</td>
<td>546</td>
<td>0.12% 0.30%</td>
<td>0.10% 0.23%</td>
<td>0.18% 1.25%</td>
</tr>
<tr>
<td>C2670</td>
<td>1193</td>
<td>0.01% 0.13%</td>
<td>0.06% 0.36%</td>
<td>0.13% 1.04%</td>
</tr>
<tr>
<td>C3540</td>
<td>1669</td>
<td>0.02% 0.08%</td>
<td>0.08% 0.45%</td>
<td>0.22% 1.48%</td>
</tr>
<tr>
<td>C7552</td>
<td>3521</td>
<td>0.01% 0.07%</td>
<td>0.02% 0.28%</td>
<td>0.45% 1.21%</td>
</tr>
</tbody>
</table>

### Lumped Gate Level Leakage Current Model Accuracy

In Section II.B, we model the leakage current of a gate as a lumped exponential function of $V_{VG}$. To verify this, we conduct experiments on an 8-input AND gate in 32nm technology. This complex gate consists of a 2-input NAND gate, two 3-input NAND gates and a 3-input NOR gate. Given a certain input vector as shown in Figure 2.12, this gate includes all the serial and parallel structures discussed in Section II.B.

![Figure 2.12: 8-input AND Gate Topology](image)

According to our method, the leakage current of a gate is modeled as a lumped $V_{VG}$-controlled current source. So in this experiment, we set the $V_{VG}$ of the gate to 7 different values and measure...
the leakage current of the gate. The predicted leakage current values and HSPICE simulations are shown in Table II and Figure 2.13. The error is negligible when the $V_{VG}$ level is low ($V_{VG} < 500mV$). The error percentage is higher when the $V_{VG}$ level is high ($V_{VG} > 500mV$). However when $V_{VG}$ is high, since the absolute value of the leakage current is very small, the error has minor impact on the EBT model.

Table 2.2: Correlation of Lumped Gate Leakage Current Model on AND8

<table>
<thead>
<tr>
<th>$V_{VG}$(mV)</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model $I_{leak}$(nA)</td>
<td>56.9</td>
<td>24.7</td>
<td>10.7</td>
<td>4.6</td>
<td>2.0</td>
<td>0.9</td>
<td>0.4</td>
</tr>
<tr>
<td>Sim. $I_{leak}$(nA)</td>
<td>56.9</td>
<td>23.9</td>
<td>10.2</td>
<td>4.5</td>
<td>2.2</td>
<td>1.2</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Figure 2.13: Correlation of Lumped Gate Leakage Current Model on AND8

**Circuit Level Leakage Reduction Process Model Accuracy**

To verify the circuit level LCR model, we compare the model estimates on the time-varying leakage energy consumption with simulation data $E_{exp}(n)$ for each benchmark circuit in each technology. The average and worst case errors of our model estimates are shown in Table III. Figure 2.14 shows the correlation over time on benchmark circuit C7552 in 32nm technology. The time (X axis) is normalized by EBT and spans from 0 to 12 EBT. The accurate modeling of LCR in this time range captures the variation of the leakage current. Especially, the error at the time point of 1 EBT and vicinity determine the accuracy of EBT estimation.

**Average EBT Estimation Accuracy**

To verify the average EBT estimation, we generate 64 random input vectors for each benchmark circuit. The circuit is then put into sleep 64 times with a different state each time. The probabilities
Table 2.3: Error of Leakage Energy Consumption Estimates

<table>
<thead>
<tr>
<th>BM.</th>
<th>Gate Cnt.</th>
<th>32nm Max. Avg.</th>
<th>45nm Max. Avg.</th>
<th>65nm Max. Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>160</td>
<td>2.1% 1.6%</td>
<td>1.2% 0.9%</td>
<td>5.6% 2.8%</td>
</tr>
<tr>
<td>C499</td>
<td>202</td>
<td>4.2% 1.3%</td>
<td>1.3% 0.7%</td>
<td>4.4% 1.6%</td>
</tr>
<tr>
<td>C880</td>
<td>383</td>
<td>3.4% 2.1%</td>
<td>3.5% 2.0%</td>
<td>5.9% 3.1%</td>
</tr>
<tr>
<td>C1355</td>
<td>546</td>
<td>5.2% 2.0%</td>
<td>3.9% 1.8%</td>
<td>3.8% 1.9%</td>
</tr>
<tr>
<td>C1908</td>
<td>880</td>
<td>3.3% 1.4%</td>
<td>4.4% 2.1%</td>
<td>5.8% 1.9%</td>
</tr>
<tr>
<td>C2670</td>
<td>1193</td>
<td>4.6% 2.5%</td>
<td>4.3% 1.6%</td>
<td>6.7% 3.5%</td>
</tr>
<tr>
<td>C3540</td>
<td>1669</td>
<td>4.1% 2.4%</td>
<td>3.5% 2.2%</td>
<td>6.5% 3.1%</td>
</tr>
<tr>
<td>C5315</td>
<td>2307</td>
<td>3.9% 1.9%</td>
<td>2.7% 1.8%</td>
<td>6.4% 3.0%</td>
</tr>
<tr>
<td>C6288</td>
<td>2406</td>
<td>4.1% 2.2%</td>
<td>3.6% 1.6%</td>
<td>2.4% 1.1%</td>
</tr>
<tr>
<td>C7552</td>
<td>3521</td>
<td>5.7% 2.7%</td>
<td>3.3% 2.5%</td>
<td>6.2% 2.2%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>5.7% 2.0%</td>
<td>4.4% 1.7%</td>
<td>6.7% 2.4%</td>
</tr>
</tbody>
</table>

Figure 2.14: Error of Leakage Energy Consumption Estimates on C7552

of the occurrences of each state are assumed to be the same. \( P_u = 1/64 \). In each sleep occurrence, we simulate the leakage current \( I_{exp} \) before ground gating, and simulate the time-varying leakage energy consumption \( E_{exp}(n)u \) after ground gating. Also, we simulate the switching energy penalty of the footer. We multiply this penalty by a factor \( \delta \) in order to emulate the extra energy penalty of the control circuit. Since the control circuit penalty is very likely to be smaller than the footer switching penalty, we set \( \delta \) as 2 to be conservative. Finally with all the data, we find out a sample point \( N \) satisfying:

\[
E_{penalty} = N \sum_{1-64} \frac{1}{64} V_{DD} I_{exp}^u - \sum_{1-64} \frac{1}{64} E_{exp}(N)^u
\]  

(2.45)

The time corresponding to \( N \) is the simulated average EBT. It is compared with the model estimates in Table IV. The results show that our model estimates have on the average 1.8%, maximally 3.0% error when compared with HSPICE.
Table 2.4: Model VS. Simulation on Average EBT

<table>
<thead>
<tr>
<th>BM. Unit=ns</th>
<th>32nm Sim. Mod. Er.</th>
<th>45nm Sim. Mod. Er.</th>
<th>65nm Sim. Mod. Er.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.95 ± 2.00 2.3%</td>
<td>3.94 ± 3.97 0.8%</td>
<td>14.84 ± 14.45 2.7%</td>
</tr>
<tr>
<td>C432</td>
<td>1.58 ± 1.61 1.9%</td>
<td>3.89 ± 3.98 2.3%</td>
<td>14.55 ± 15.42 0.9%</td>
</tr>
<tr>
<td>C499</td>
<td>1.50 ± 1.54 2.7%</td>
<td>3.75 ± 3.80 1.3%</td>
<td>14.29 ± 14.22 0.6%</td>
</tr>
<tr>
<td>C880</td>
<td>1.46 ± 1.49 2.0%</td>
<td>3.55 ± 3.60 1.9%</td>
<td>13.36 ± 13.21 1.1%</td>
</tr>
<tr>
<td>C1355</td>
<td>1.28 ± 1.32 2.8%</td>
<td>3.19 ± 3.27 2.3%</td>
<td>12.49 ± 12.71 1.7%</td>
</tr>
<tr>
<td>C1908</td>
<td>1.36 ± 1.39 2.3%</td>
<td>2.73 ± 2.77 1.6%</td>
<td>12.99 ± 13.17 1.4%</td>
</tr>
<tr>
<td>C2670</td>
<td>1.41 ± 1.45 2.4%</td>
<td>3.51 ± 3.60 2.5%</td>
<td>13.60 ± 13.62 0.2%</td>
</tr>
<tr>
<td>C3540</td>
<td>1.44 ± 1.48 2.7%</td>
<td>3.63 ± 3.68 1.3%</td>
<td>13.94 ± 13.93 0.0%</td>
</tr>
<tr>
<td>C5315</td>
<td>2.19 ± 2.26 3.0%</td>
<td>4.35 ± 4.44 2.1%</td>
<td>20.58 ± 20.17 2.0%</td>
</tr>
<tr>
<td>C6288</td>
<td>1.39 ± 1.43 2.5%</td>
<td>3.47 ± 3.56 2.5%</td>
<td>13.43 ± 13.45 0.1%</td>
</tr>
<tr>
<td>Overall</td>
<td>2.5%</td>
<td>1.9%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>

In the above table for 32nm technology, the variation of the average EBT for different circuit topologies can be 71% (C1908 VS. C6288). However, by using the EBT model in [38, 39, 36], the average EBT for different circuits will be the same. Hence their models can have at least 71% inaccuracy when circuit topology is considered.

Circuit states also have significant impact. Using the single state circuit EST model in Section III.A, we have an individual EBT value \( T_u \) for each state \( u \). Figure 2.15 shows the distribution of the total 64 \( T_u \) for AND8, C432 and C3540. The X axis is the variation percentage of \( T_u \). The Y axis is the number of the states, which fall into a same variation range. It can be observed that for larger circuits, the distribution of each individual \( T_u \) tends to be more uniform and concentrated. Since the average EBT is determined by the distribution of \( T_u \), this observation indicates that for a large circuit, its average EST is likely to converge quickly by considering a limited number of circuit states.

![Figure 2.15: EBT Distributions of AND8, C432 and C3540](image)

HSPICE takes 47 hours to obtain the average EBT of C7552, while our model estimation takes
1 minute (3000× speedup).

### 2.3 Energy Breakeven Time Modeling for Reverse Body Biasing

In this section, we derive an accurate EBT model for RBB. In the following, we use PMOS RBB as an example. NMOS RBB can be modeled in a similar way. Body biasing can be implemented by a charge bump circuitry [37], which usually has significant charging time. In order to satisfy the critical timing requirement of RALR, we use the \(V_{th}\) hopping scheme explained in [11], where high-\(V_{th}\) control transistors are inserted to switch the substrate voltage between the normal value (\(V_{DD}\)) and the biased value (\(V_P\)), as shown in Figure 2.16. The sizing of the control transistors are important. The size of the one (\(P_1\)) controlling the bias voltage determines the charging speed of the substrate, and thus the leakage reduction speed. The size of the one (\(P_2\)) controlling the normal voltage determines the discharge speed of the substrate, and thus the wake up time. Their sizing will be discussed in Section III.D. Here we derive a general energy saving model for all sizes.

#### 2.3.1 Circuit Modeling During RBB Mode Transition

When the control signal of RBB is asserted, the energy overhead for switching the control transistor is:

\[
E_C = C_C V_P^2 \tag{2.46}
\]

where \(C_C\) is the gate capacitance of the control transistors (\(C_{C1}\) of \(P_1\) and \(C_{C2}\) of \(P_2\)). Note that the voltage of both control signals should be \(V_P\), instead of normal \(V_{DD}\). Next, we study three major physical phenomena that occur after RBB is applied.
Phenomenon 1: Substrate Charging

Once the control signal switches the substrate to the bias voltage source, the bias voltage source starts to charge up the substrate, via the control transistor $P_1$. By assuming that the resistance of $P_1$ is $R_1$, this process can be characterized by:

$$
\Delta V_b(t) = (V_P - V_{DD})(1 - e^{-\frac{t}{R_1C_b}})
$$

where $\Delta V_b(t)$ is the increment of the P substrate voltage, and $C_b$ is the total capacitance of the P substrates.

$C_b$ consists of two types of capacitance: the capacitance ($C_{bV}$) between P substrate and $V_{DD}$ and the capacitance ($C_{bG}$) between P substrate and the ground. The charging currents to $C_{bV}$ and $C_{bG}$ have different impact. For example, Figure 2.17 shows applying RBB to an inverter with input ’1’. It can be modeled as a three-terminal device, as shown in the right side. The charging current $I_{bV}$ directly goes to the ground, while the charging current $I_{bG}$ of $C_{bG}$ goes through the off-state PMOS and then reaches the ground. Similar to the self-discharging current in PG, this $I_{bV}$ occupies a portion of the maximum current ($I_{off}$) that can leak through the circuit, and thus reduces the leakage current ($I_{VDD}$) from $V_{DD}$. However, $I_{bV}$ consumes energy of the bias voltage source. So unlike PG, no extra energy is saved in this case.

Phenomenon 2 and 3: Leakage Variation

With the increase of the substrate voltage, the second phenomena of RBB is the reduction of the subthreshold leakage. It is formulated in [43] as:

$$
I_{sub} = \hat{I}_{sub}e^{B_s\Delta V_b}
$$

(2.48)
where $I_{\text{sub}}$ is the subthreshold leakage under zero bias, and $B_s$ is a technology dependent parameter. $\Delta V_b$ is the increment of the substrate voltage $V_b$. Meanwhile, the BTBT leakage increases [43]:

$$I_{\text{btbt}} \approx \tilde{I}_{\text{btbt}} e^{-B_t \Delta V_b}$$

(2.49)

where $\tilde{I}_{\text{btbt}}$ is the BTBT leakage under zero bias, and $B_t$ is a technology dependent parameter.

Next, we study the leakage variation of a gate when applying RBB. Figure 2.18 shows a large fan-in gate with output '0' for illustration purpose. Its pull-down network is on, so the off-state PMOS in the pull-up network are the sources of leakage current. Assume that the pull-up network has four parallel branches, and in each branch there are four PMOS in series. With a particular input vector, it has the on and off pattern as shown in the figure. RBB changes the $V_b$ of all the off-state PMOS to the same value. Thus using Equation 2.48 for each transistor in $b_1$, we have:

$$\begin{align*}
I_1' &= I_{b_1} e^{B_s \Delta V_b} = I_{b_1} e^{B_s \Delta V_b} = I_{b_1} \\
I_2' &= I_{b_2} e^{B_s \Delta V_b} = I_{b_2} e^{B_s \Delta V_b} = I_{b_2} \\
I_3' &= I_{b_3} e^{B_s \Delta V_b} = I_{b_3} e^{B_s \Delta V_b} = I_{b_3} \\
I_4' &= I_{b_4} e^{B_s \Delta V_b} = I_{b_4} e^{B_s \Delta V_b} = I_{b_4} \\
\end{align*}$$

(2.50)

where $I_{b_1}$ and $I_{b_1}'$ is the subthreshold leakage of $b_1$ before and after RBB, respectively. We can observe that the subthreshold leakage current of each branch is an exponential function of $\Delta V_b$.

Now for the complex gate, we have:

$$I_{\text{gate}} = I_{b_1} e^{B_s \Delta V_b} + I_{b_2} e^{B_s \Delta V_b} + I_{b_3} e^{B_s \Delta V_b} + I_{b_4} e^{B_s \Delta V_b}$$

$$= (I_{b_1} + I_{b_2} + I_{b_3} + I_{b_4}) e^{B_s \Delta V_b}$$

$$= (\tilde{I}_{\text{gate}}) e^{B_s \Delta V_b}$$

(2.51)

Figure 2.18: Leakage Current of A Complex Gate
So the subthreshold leakage of the gate is also an exponential function of $\Delta V_b$. Furthermore, since all the gates in the circuit have the same exponent $B_s$, the total subthreshold leakage of the whole circuit can be modeled as:

$$I_S = \sum_i \hat{I}_{S_i} e^{B_s \Delta V_b}$$

(2.52)

where $\hat{I}_{S_i}$ is the zero-biased subthreshold leakage of each gate $i$. Similarly, the BTBT leakage of the whole circuit is:

$$I_T = \sum_i \hat{I}_{T_i} e^{-B_t \Delta V_b}$$

(2.53)

where $\hat{I}_{T_i}$ is the zero-biased BTBT leakage of each gate $i$.

**Circuit Model of RBB**

Based on the previous analysis, we model a body-biased circuit as shown in Figure 2.19. The leakage current of all gates in the circuit is lumped into three $V_b$-controlled current sources: sub-threshold leakage $I_S$, substrate to ground BTBT leakage $I_{TG}$, and substrate to $V_{DD}$ BTBT leakage $I_{TV}$. The substrate capacitance of all gates is lumped into two capacitors: $C_{bV}$ ($C_{bV} = \sum_i C_{bV_i}$) and $C_{bG}$ ($C_{bG} = \sum_i C_{bG_i}$). Note that PMOS RBB only reduces the leakage of the gates whose pull-up network is off. For those gates with a pull-down network in off-state, NMOS RBB should be applied. However, since the leakage of a pull-down network does not change with PMOS RBB, it is not included in the circuit model.

![Figure 2.19: Model of PMOS RBB Circuit](image-url)
2.3.2 Energy Breakdown Time Modeling for RBB

With the circuit model, we first derive the energy saving model of RBB. The BTBT leakage currents in Figure 2.19 are:

\[
\begin{align*}
I_{TV} &= \sum_i \hat{I}_{TVi} e^{-B_1 \Delta V_b} \\
I_{TG} &= \sum_i \hat{I}_{TGi} e^{-B_2 \Delta V_b}
\end{align*}
\]  

(2.54)

where \(\Delta V_b\) can be solved using Equation 2.47. The charging currents of substrate capacitance \(C_{bV}\) and \(C_{bG}\) are:

\[
\begin{align*}
I_{bV} &= (V_P - V_b) \frac{C_{bV}}{R_1(C_{bV} + C_{bG})} \\
I_{bG} &= (V_P - V_b) \frac{C_{bG}}{R_1(C_{bV} + C_{bG})}
\end{align*}
\]

(2.55)

Hence, the total current from the bias voltage \(V_P\) to \(V_{DD}\) is:

\[I_{bias} = I_{TV} + I_{TG} + I_{bV} + I_{bG}\]

(2.56)

Making all currents as functions of time, we have the energy consumption of the bias voltage source at any time \(t\):

\[E_{bias}(t) = \int_0^t V_P I_{bias}(t) \, dt\]

(2.57)

Now, applying KCL current law at \(V_{DD}\) in Figure 2.19, we have:

\[I_{V_{DD}} + I_{TV} + I_{bV} = I_S\]

(2.58)

Hence, the leakage current drawn from \(V_{DD}\) is:

\[
\begin{align*}
I_{V_{DD}} &= I_S - I_{TV} - I_{bV} \\
I_S &= \sum_i \hat{I}_{Si} e^{B_1 \Delta V_b}
\end{align*}
\]

(2.59)

Making all the currents as functions of time, we have the energy consumption of \(V_{DD}\) at any time \(t\):

\[E_{V_{DD}}(t) = \int_0^t V_{DD} I_{V_{DD}}(t) \, dt\]

(2.60)

Using Equations 2.46, 2.47, together with 2.57 and 2.60, we have the overall energy saving model \((E_{BB}(t))\) of RBB:

\[E_{BB}(t) = V_{DD}(\hat{I}_S + \hat{I}_T)t - C_C V_P^2 - E_{bias}(t) - E_{V_{DD}}(t)\]

(2.61)

where \(\hat{I}_S\) and \(\hat{I}_T\) are the total subthreshold and BTBT leakage current of the circuit without RBB, respectively.

Once the energy saving model is obtained, the single-state and multiple-state EBT of RBB can be obtained using the similar approach of PG.
2.3.3 Optimum Design Point of RBB

In [43], the optimum value of bias voltage $V_P$ is determined by maximizing the total leakage reduction ratio. However, this optimum value (OPT1 in Figure 2.20) is obtained by assuming that the circuit stays in RBB mode for a long time and the energy overhead ($E_b$) for charging the substrate is not considered. For RALR, since the state transition occurs frequently, $E_b$ is not negligible. As shown in Figure 2.20, $E_b$ has a quadratic dependency on $V_P$, while the leakage energy reduction increases slowly with $V_P$ when the substrate voltage is high enough. Thus, by counting in the energy overhead, there exists another optimum $V_P$ (OPT2 in Figure 2.20) for RALR, at which the net energy saving is maximized.

![Figure 2.20: Optimum Bias Voltage of RBB](image)

To fully charge up the substrate, the energy overhead is:

$$E_b = (C_{bg} + C_{bV}) \Delta V_P^2 + C_{bg} V_{DD} \Delta V_P$$

(2.62)

The leakage energy saving ($E_{saving}(t)$) is:

$$E_{saving}(t) = (\tilde{I}_S - I_S) \cdot t + (\tilde{I}_T - I_T) \cdot t$$

$$= (1 - e^{B_s \Delta V_P}) \tilde{I}_{St} t + (1 - e^{-B_t \Delta V_P}) \tilde{I}_{Tt} t$$

(2.63)

Assume that we want to achieve the maximal net energy saving at time $T$, then the net energy saving ($E_{net}(t)$) at $T$ is:

$$E_{net}(T) = E_{saving}(T) - E_b - E_C$$

(2.64)

From Equations 2.62, 2.63 and 2.46, we can see that the above equation is a function of $\Delta V_P$.

The maximal value of $E_{net}(T)$ can be obtained by solving:

$$\frac{\partial E_{net}(T)}{\partial \Delta V_P} = 0$$

(2.65)

The control transistor $P_1$ also has an optimum sizing. If $P_1$ is designed to be too small, then substrate charging will be slow and thus the leakage reduction will be less effective. If $R_1$ is too
large, then the switching overhead of $P_1$ will exceed the leakage saving. The optimum sizing ($Z_1$) of $P_1$ can be determined as follows. $R_1$ and $C_C$ in Equation 2.61 and 2.47 can be considered as functions of $Z_1$. Thus, Equation 2.61 turns into a function of $Z_1$. If we want to achieve the maximal energy saving at time $T$, the optimum $Z_1$ can be determined by solving:

$$\frac{\partial E_{BB}(T)}{\partial Z_1} = 0 \quad (2.66)$$

In the above analysis, $T$ controls the circuit idleness which we want to optimize for. To optimize RBB for standby leakage reduction, $T$ can be set to a large value. Otherwise for RALR, $T$ can be set to a small value. Figure 2.21 shows the $V_P$ values optimized for different $T$ on benchmark circuit C1355 in 65nm technology. When $T$ is set to be sufficiently large, the energy overhead for state transition is negligible. Hence the optimum $V_P$ is close to the optimum bias voltage in [43]. In Figure 2.21, this value is 3V. (The normal bias voltage is 0.9v.) This bias voltage yields the optimum energy saving only when $T$ is larger than 3ms. So it is not applicable for RALR. For RALR applications, for example when $T$ is 100ns, The optimum $V_P$ is only 1.5V, as shown in Figure 2.21.

2.3.4 Experimental Results

We conducted experiments to compare our model estimates with HSPICE simulation results. The ISCAS85 benchmark circuits in 32nm, 45nm and 65nm technologies [41] are used in the experiments. The gate level implementation and parasitic information of the benchmarks is from [42]. Each circuit is given a particular input for all the experiments. The simulation temperature is set at 110C to emulate the runtime temperature. The gate leakage is set to be zero.
The PMOS RBB is implemented as illustrated in Figure 2.16. The bias voltage and size of the control transistors are set to enable the maximum energy saving at the energy breakeven time of PG. $P_1$ and $P_2$ are of the same size for simplicity. Similar to PG, we double the switching energy of them.

**Energy Saving Model Verification of RBB**

For RBB model verification, we simulated the energy consumption at $V_{DD}$, as well as at the bias voltage $V_P$. Then we sum them up to obtain the total energy consumption. Table 2.5 shows our RBB model has on the average 1.3%, maximally 5.5% error on the total energy consumption estimation. Figure 2.22 illustrates the model versus simulation for C7552.

<table>
<thead>
<tr>
<th>Ckt.</th>
<th>32nm</th>
<th>45nm</th>
<th>65nm</th>
<th>Ckt.</th>
<th>32nm</th>
<th>45nm</th>
<th>65nm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M. A.</td>
<td>M. A.</td>
<td>M. A.</td>
<td></td>
<td>M. A.</td>
<td>M. A.</td>
<td>M. A.</td>
</tr>
<tr>
<td>C432</td>
<td>2.4 1.5</td>
<td>2.8 1.6</td>
<td>5.5 2.5</td>
<td>C2670</td>
<td>2.1 0.9</td>
<td>2.4 1.1</td>
<td>3.1 1.3</td>
</tr>
<tr>
<td>C499</td>
<td>1.9 1.0</td>
<td>4.3 1.5</td>
<td>3.2 2.0</td>
<td>C3540</td>
<td>2.1 1.0</td>
<td>3.0 1.2</td>
<td>2.3 1.3</td>
</tr>
<tr>
<td>C880</td>
<td>1.6 0.9</td>
<td>3.5 2.0</td>
<td>4.2 2.0</td>
<td>C5315</td>
<td>1.7 1.0</td>
<td>2.6 1.3</td>
<td>2.4 1.7</td>
</tr>
<tr>
<td>C1355</td>
<td>2.0 1.0</td>
<td>1.9 0.9</td>
<td>3.0 1.4</td>
<td>C6288</td>
<td>2.4 1.3</td>
<td>4.0 1.2</td>
<td>3.4 1.1</td>
</tr>
<tr>
<td>C1908</td>
<td>2.1 1.0</td>
<td>1.9 1.0</td>
<td>2.1 1.2</td>
<td>C7552</td>
<td>1.6 1.0</td>
<td>2.1 1.5</td>
<td>1.3 0.4</td>
</tr>
<tr>
<td>Overall</td>
<td>2.4 1.1</td>
<td>4.3 1.3</td>
<td>5.5 1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.22: Total Energy Consumption Estimates of RBB on 32nm C7552
Optimum Design Point of RBB

We conducted experiments to verify the optimum design points of RBB. Figure 2.23 shows the total energy saving of 65nm C1355 at 13.1ns (energy breakeven time of PG) with different bias voltage $V_P$. As predicted by the model, setting $V_P$ as 1.1v ($V_{DD}=0.9v$) yields the maximum energy saving.

Figure 2.23: RBB Optimum Bias Voltage on 65nm C1355 at Idle Time 13.1ns

Figure 2.24 shows the total energy saving model as a function of time with different $V_P$ value. As can be observed, if $V_P$ value is smaller than 1.1v (1.0v, the green line), the overhead is smaller but the leakage reduction is not effective enough. If $V_P$ value is larger than 1.1v (1.2v-1.4v), the leakage reduction is improved. However the overhead will be too high such that at 13.1ns, their net energy saving is less than using 1.1V.

Figure 2.24: Total Energy Saving Model with Different $V_P$ on 65nm C1355

2.4 Conclusion and Future Work

In this chapter, we study the optimum control policy problem of RTLC. We emphasize that EBT modeling is more important than WUT modeling because EBT have much larger value than WUT.
in both PG and RBB techniques. Hence the main part of this chapter focuses on the EBT model derivation for both PG and RBB. Our approach first builds the model for the circuit during mode transition. Then based on the circuit model, we derive the energy saving model and then EBT model. Furthermore, we propose the concept of average EBT for a circuit during runtime, receiving multiple input vectors. In RBB modeling, we propose a new optimum design point for RBB, which factors in the energy overhead for mode transition.

Our models in this chapter not only answer the question of optimum control policy, they also provide the foundation for further quantitative study in this proposal. They will be used in all the following chapters to perform optimization, or to determine design methodologies. Regarding design automation, we do not necessarily use the full-fledged models due to its high computational complexity. Instead, we can use a regression model to make a trade-off between speed and accuracy. The whole modeling process also provides insights of leakage reduction by using PG and RBB. Later it becomes crucial to help us to choose one technique for RTLC in circuit active mode.

Since we assume fixed temperature and no process variation in the above study, one possible future work is to consider the temperature and process variation impact on the models.
Chapter 3

Optimum Granularity for Run-time Leakage Control

The granularity of RTLC has been an open question. Most of the current designs adopts block level, or coarse-grained approach [44]. In order to achieve more aggressive leakage reduction, fine-grained approach has been proposed recently [45, 46, 47, 48, 49, 36, 50, 51, 52]. Figure 3.1 shows an example of coarse-grained PG versus fine-grained PG. As shown in Figure 3.1, coarse-grained PG uses a single footer (or multiple distributed footers with connected virtual ground) to control the leakage of the whole circuit. On the contrary, fine-grained PG has an individual footer for each gate. The footer of each gate can be turned on and off separately. Fine-grained PG has several advantages over coarse-grained PG. It is easy for synthesis, and causes less ground bouncing problem. Most importantly, it is supposed to allow better circuit slack utilization [45], since each individual footer is controllable.

![Coarse-Grained Power Gating vs Fine-Grained Power Gating](image)

**Figure 3.1: Coarse-grained VS. Fine-Grained Power Gating**

Various aspects of the fine-grained approach have been studied, such as sizing [45], wake-up
time [46], state retention [47], control schemes [48, 49, 36] and robustness issues [50, 51]. However to the best of our knowledge, a key problem has not been quantitatively studied yet: Does the fine-grained approach yield the optimum design? If not, what is the optimum granularity?

To answer this question, we first introduce the definition of energy breakeven time (EBT). EBT is defined as the minimum time for a circuit (or a gate) to stay in low-leakage mode, such that leakage saving compensates the energy penalty for mode transition [37]. Ideally, the fine-grained RTLC yields maximum net leakage saving, if each gate in the circuit is put into low-leakage mode whenever the idleness of the gate exceeds its EBT. However to implement this, each gate requires an individual RTLC control signal. This will cause unaffordable area and power overhead. The power consumption of all the control signals can even overwhelm the leakage saving. Therefore although the fine-grained approach yields maximum leakage saving, it does not necessarily yield the optimum design due to its huge control cost.

In general, control cost increases when the granularity reduces. The trade-off is whether the improvement of leakage saving with finer granularity overcomes the extra control cost. So for a RTLC solution, there should exist an optimum granularity, which balances leakage saving and control cost. Figure 3.2 illustrates the trade-off among coarse-grained, fine-grained and optimum-grained approaches.

![Figure 3.2: Coarse-grained, Fine-grained and Optimum-grained](image)

Given the problem of control cost, although recent researches target the fine-grained approach [48, 49, 36], they are actually implemented with medium granularity, on gate clusters. Bhunia et al. [48] proposed a Shannon expansion based clustering method to apply PG. Leinweber et al. [49] improved the method in [48] by performing hypergraph partitioning before Shannon expansion. Usami et al. [36] proposed to use clock gating signals as references to partition a circuit into several PG domains. Each domain belongs to a clock gating signal. When clock gating is applied, PG will be applied as well to the corresponding domain. There are two common issues of these three studies: 1) They cannot guarantee the net leakage saving to be positive, because they did not
consider the energy penalty for PG mode transition when performing circuit clustering. In [48, 49], the energy penalty for switching PG can exceed leakage saving if input activity is high enough (Their experiments used slow clock frequency as 200MHZ with 20% switching activity). In [36], same problem occurs if the clock gating signal flips frequently. So the first problem makes these studies less practical. 2) None of these studies quantifies the leakage saving and control cost of their schemes. So they are not able to make trade-off between leakage saving and control cost, and thus cannot answer the question of the optimum granularity.

This paper aims at seeking the optimum granularity to apply RTLC on a combinational circuit, such that leakage saving and control cost reach a balanced point. As shown in Figure 3.3, for any given circuit, we first analyze its potential idleness based on the circuit topology and input statistics. Next, we analyze how to exploit the potential idleness with finer granularity. More importantly, we build models to quantify the leakage saving and control cost with different granularity. Finally based on model, we will try to determine the optimum granularity. In our scheme, to address the first common problem of [48, 49, 36], a workload prediction is used to guide the mode switching of RTLC. Detailed mechanisms of this predictor are explained in [1]. In brief, the workload predictor sends out RTLC enable signals only when the slackness in circuit workload is predicted to be larger than the EBT of the circuit, or a circuit cluster. The rate of successful prediction depends on the regularity of the workload. Here we assume that the workload is very regular so that the successful prediction rate is 100%.

Both power gating (PG) and reverse body biasing (RBB) are effective techniques to implement RTLC. In this paper, we choose RBB to be the target technique because PG has the output-floating problem. Isolation cells need to be inserted at the interface of different PG domains [53]. This adds significant area and power overhead for applying PG with finer granularity. However, although only RBB is demonstrated in this paper, our proposed methods are also applicable for PG. The only difference is that to PG, the overall control cost must include the cost for isolation cells.
This chapter is organized as follows. Section 3.1 derives the basic leakage saving model of RBB as a foundation for quantitative study. Section 3.2 proposes two methods to exploit temporal and spatial idleness by applying RBB with finer granularity. Section 3.3 formulate the optimum granularity problem and presents an heuristic algorithm to obtain a near-optimum granularity. Section 3.4 shows the experimental results. Finally Section 3.5 concludes the paper and explains the future work.

3.1 Leakage Saving Modeling OF RBB

RBB can be implemented in a discrete manner using \( V_{th} \)-hopping, or in a continuous manner using dynamic \( V_{th} \) scaling [6]. Here we choose \( V_{th} \)-hopping since it is easy to implement. A basic \( V_{th} \) hopping scheme is illustrated in Figure 3.4. When the controller decides to apply RBB on the circuit, it connects the circuit body to bias voltage sources (\( V_P \) for PMOS, \( -V_N \) for NMOS) by turning on the switch transistors \( S_1 \) and \( S_3 \) and turning off the switch transistors \( S_2 \) and \( S_4 \). So the \( V_{th} \) of each transistor in the circuit increases due to the body effect. As a result, subthreshold leakage reduces due to higher \( V_{th} \). Meanwhile, the BTBT tunneling leakage slightly increases as a side effect. In the following, we show the modeling process of leakage saving by applying PMOS-RBB. NMOS-RBB can be modeled in a similar way.

![Figure 3.4: \( V_{th} \) Hopping Implementation](image)

For a particular gate (\( g \)) in the circuit, assume that its original subthreshold leakage current with zero body biasing is \( \tilde{I}_s \), and its original BTBT leakage is \( \tilde{I}_b \). When PMOS-RBB is applied, the PMOS body voltage switches to \( V_P \). This results in a reduction in the subthreshold leakage of \( g \) [43]:

\[
I_s = \tilde{I}_s e^{K_s(V_{DD}-V_P)}
\] (3.1)
Meanwhile, the BTBT leakage increases [43]:

\[ I_b \approx \hat{I}_b e^{-K_s(V_{DD} - V_P)} \]  (3.2)

where \( K_s \) and \( K_b \) are technology dependent parameters. For a idle period of length \( T \), the leakage energy (\( S_g \)) saving of \( g \) by applying RBB is:

\[ S_g(T) = (\hat{I}_s(1 - e^{K_s(V_{DD} - V_P)}) + \hat{I}_b(1 - e^{-K_b(V_{DD} - V_P)}))V_{DD}T \]  (3.3)

The RBB mode transition incurs an energy overhead (\( O_g \)) for charging the PMOS body capacitance (\( C_g \)) of gate \( g \):

\[ O_g = C_g(V_P - V_{DD})^2 \]  (3.4)

Thus, the net leakage saving (\( E_g(T) \)) is:

\[ E_g(T) = S_g(T) - O_g \]  (3.5)

Equation 3.5 is applicable when \( g \) has only one state when RBB is applied. In a runtime environment, the circuit can receive different input patterns and thus \( g \) can have multiple states when RBB is applied. In [54], Xu et al. have presented a method to factor in the impact of input patterns for PG. Similar idea can be used for RBB. Assume that during runtime, the circuit receives \( U \) typical input patterns. The probability of occurrence for each input pattern is \( P_u \) (\( u = 1..U \)). For each input pattern \( u \), the net leakage saving of gate \( g \) is \( E_g^u(T) \). Then according to [54], the average net leakage saving (\( \tilde{E}_g(T) \)) of \( g \) for all \( U \) input patterns is:

\[ \tilde{E}_g(T) = \sum_u P_u E_g^u(T) \]  (3.6)

Similarly, the EBT (\( B_g \)) of \( g \) can be calculated by:

\[ B_g = \frac{O_g}{\sum_u P_u S_g^u(T)|_{T=1}} \]  (3.7)

For the whole circuit, its net leakage saving is the summation of the saving of each gate:

\[ \tilde{E}_{ckt}(T) = \sum_g \tilde{E}_g(T) \]  (3.8)

The EBT of the whole circuit can be obtained by:

\[ B_{ckt} = \frac{\sum_g O_g}{\sum_g \sum_u S_g^u(T)|_{T=1}} \]  (3.9)
By using the workload predictor, the circuit will only be put into RBB mode when the expected idleness is larger than $B_{swt}$. So the leakage saving before $B_{swt}$ is zero. Hence Equation 3.8 needs to be modified into a discontinuous function:

$$
\tilde{E}'_g(T, B_{swt}) = \begin{cases} 
0 & (T < B_{swt}) \\
\tilde{E}_g(T) & (T \geq B_{swt})
\end{cases}
$$

(3.10)

3.2 Leakage Saving and Control Cost With Finer Granularity

During runtime, not all the gates in a circuit are functioning at all time. The idleness of some gates gives us opportunities to apply RTLC on them to save leakage. The amount of idleness that we can exploit determines how much leakage we can save. In this section, we will introduce two methods to aggressively exploit the temporal and spatial idleness of a runtime circuit.

3.2.1 Temporal Idleness Exploitation

Temporal idleness is caused by the slackness in the circuit workload. To exploit temporal idleness, the conventional coarse-grained approach will apply RBB to the whole circuit when the slackness is larger than the EBT of the circuit. Temporal idleness exploitation can be improved by applying RBB with finer granularity. To explain this, we start from the following observation. When RBB is applied, the leakage saving and energy penalty for mode transition of each gate in the circuit can vary significantly. Some gates have high leakage saving and low penalty, while others have low leakage saving but high penalty. To quantify this variation, we analyze the EBT of each gate in ISCAS85 C880. As shown in Figure 3.5, the X-axis is EBT value, and the Y-axis is the count of the gates, whose EBT falls into the same range.

![Figure 3.5: EBT Variation of Each Gate in C880](image)
In Figure 3.5, the EBT of each gate varies up to 70 times, from 4 to 280 clock cycles. This indicates that by applying RBB, some gates which have high leakage saving but low penalty, can achieve net energy saving after just 4 cycles. But some other gates which have low leakage saving but high penalty, can only save energy after 280 cycles. The vertical line ($B_{ckt}=14$ cycles) in Figure 3.5 is the EBT of the whole circuit. The coarse-grained approach will put the whole circuit into RBB mode only when the slackness is larger than $B_{ckt}$. However, there are two disadvantages in doing so:

1) For those gates ($G_L$) whose EBT is larger than $B_{ckt}$, entering RBB mode at $B_{ckt}$ yields negative net leakage saving, and thus causes more energy consumption.

2) For those gates ($G_S$) whose EBT is smaller than $B_{ckt}$, entering RBB mode at $B_{ckt}$ yields positive net leakage saving. However their leakage saving potentials are not fully exploited, since each gate in $G_S$ can enter RBB mode earlier.

These two disadvantages can be reduced if RBB is applied on $G_L$ and $G_S$ separately. $G_S$ can enter RBB mode earlier than $B_{ckt}$ to exploit more idleness, while $G_L$ can enter RBB mode later than $B_{ckt}$ to avoid unnecessary mode switching. Thus, we can see the potential of improving temporal idleness exploitation with finer granularity. Ideally, each gate in the circuit should enter RBB mode individually, whenever the circuit idleness is larger than its EBT value. This is essentially the fine-grained approach. It maximally exploits the temporal idleness of each gate and thus yields the upper bound. Based on Equation 3.10, we can quantify this upper bound as:

$$E_{tmpU}(T) = \sum_g \tilde{E}_g'(T, B_g)$$

(3.11)

The coarse-grained approach yields the lower bound:

$$E_{tmpL}(T) = \sum_g \tilde{E}_g'(T, B_{ckt})$$

(3.12)

If the circuit is partitioned into $M$ clusters, denote the EBT of each cluster as $B_m$ ($m = 1..M$). Then the net leakage saving of $M$-way partitioning is:

$$E_{tmpM}(T) = \sum_g \tilde{E}_g'(T, B_m) \quad (g \in m, \ m = 1..M)$$

(3.13)

Figure 3.6 shows the net leakage saving as a function of idle time, for 2/3/4-way temporal partition of C880. (Temporal partition will be explained in Section 4.2.) As shown in Figure 3.6, leakage saving can be significantly improved when granularity reduces. However for a 4-way partition, the
leakage saving is very close to the theoretical upper bound. Further reducing granularity yields minor improvement on temporal idleness exploitation, while control cost will keep increasing.

![Energy Saving Impv. by Temporal Partitioning on C880](image)

**Figure 3.6: Leakage Saving Impv. by Temporal Partitioning on C880**

3.2.2 **Spatial Idleness Exploitation**

Spatial idleness refers to the cases where even when a circuit has continuous workload, a subset of the circuit can still be idle. This phenomenon is due to two features in circuit topology: partial input dependency and mutual exclusion.

Partial input dependency means that some parts of a circuit only have dependency on a subset of the circuit primary inputs, instead of all of them. For example, Figure 3.7 shows a channel interrupt controller (ISCAS85 C432). This circuit is divided into five modules ($M_1$ to $M_5$). As we can see,

![Spatial Idleness Due To Partial Input Dependency](image)

**Figure 3.7: Spatial Idleness Due To Partial Input Dependency**

$M_1$ does not have dependency on channels B and C. $M_2$ does not have dependency on channel C. During runtime, it is possible that channels A and E are continually switching, while channels B and C remain static. In this case, although the circuit workload is full, module $M_1$ and $M_2$ are not functioning. They have spatial idleness due to their partial input dependency.

To exploit this spatial idleness, we can partition the circuit into three clusters, as shown in Figure
3.7. The improvement of leakage saving by performing spatial partitioning can be reflected by the variable $T$ in Equation 3.10. When the whole circuit is partitioned into clusters, each cluster has less input dependency than the whole circuit. So the probability for them to be idle increases. To quantify this, we denote that for each primary input $n$, the probability for it to be idle is $R_n$. Assume that the circuit has $N_{ckt}$ primary inputs, and $R$ for each primary input is equal for simplicity. Then the probability for all primary inputs, or the whole circuit to be idle is $R_{ckt}$. For a workload period of length $T_W$, the total idle time of the circuit is $T_W R_{ckt}$. Based on Equation 3.10, we have the net leakage saving during $T_W$ of the whole circuit without partitioning:

$$E_L = \sum_g \tilde{E}'_g(T_W R_{ckt}, B_{ckt})$$

Equation 3.14 essentially factors in both temporal (by controlling variable $B_{swt}$ of Equation 3.10) and spatial (by controlling variable $T$ of Equation 3.10) idleness exploitation. It is our overall optimization goal in Section 4. However here, in order to study pure spatial idleness, we fix the variable $B_{swt}$ as $B_{ckt}$ to disable temporal idleness exploitation. Equation 3.14 also gives the lower bound of spatial idleness exploitation. Again, the upper bound is given by the fine-grained approach. Assume that for each gate $g$, its number of dependant inputs is $N_g$, then the upper bound can be obtained by:

$$E_U = \sum_g \tilde{E}'_g(T_W R_{ckt}, B_{ckt})$$

For $M$-way partitioning, assume that the number of dependant inputs of each cluster $m$ is $N_m$. Its net leakage saving is:

$$E_M = \sum_g \tilde{E}'_g(T_W R_{ckt}, B_{ckt}) \quad (g \in m, \ m = 1..M)$$

Figure 3.8 shows the net leakage saving for 2/3/4-way spatial partition of C432. The X-axis is the probability ($R_{ckt}$) for all primary inputs to be idle. As shown in Figure 3.8, the leakage saving improvement is significant when the inputs has medium activities (between 0.4 and 0.8).

The second type of spatial idleness, mutual exclusion means that two parts of a circuit do not work concurrently in any conditions. A simple example is a MUX. When one channel of the MUX is selected, the output of unselected channels is useless. If the select signals remain the same for a period longer than the EBT of the unselected channels, the unselected channels can be put into low-leakage mode. However to do this, we need to monitor and predict the steering signals (select signals for MUXes). This turns out to be difficult for the gates inside a circuit. For example, Figure 3.9 shows an inside cluster ($M_1$), which does not receive any primary inputs.
To predict the mutual exclusion of $M_1$, we need to monitor the internal steering signals. This is difficult to implement for two reasons: 1) Predictors need to be inserted inside the circuit. This causes large overhead and disturbs the original design. 2) The steering signals inside a circuit may not have regularity, so the prediction may not be accurate. Hence in our method, we only exploit mutual exclusion for the clusters, whose steering signals are primary inputs, such as $M_2$ in Figure 3.9.

The studies in [48, 49] essentially exploit the mutual exclusion in a circuit. But their schemes are not aware of this prediction problem of steering signals. In their scheme, mode transition is only dependant on the values of steering signals in the next clock cycle. As discussed in Section 1, this can result in negative net leakage saving if steering signals flip frequently.

### 3.2.3 Control Cost With Finer Granularity

The control cost mainly consists of three parts: the cost of the predictor(s), the cost of switch transistors ($S_1$ to $S_4$ in Figure 3.4), and the routing area for bias voltage rails. [1] reported that the predictor has small area and power consumption. So we ignore the predictor cost in this paper.

**Area and Power Cost of Switch Transistors**

The area of switch transistors ($A_{swt}$) are determined by the required speed for mode transition:
wake-up time. In [55], wake-up time is calculated by the charge stored in the body capacitances \( C_{body} \) divided by the \( I_{on} \) of switch transistors:

\[
T_{WUT} = \frac{C_{body} \Delta V_P}{I_{on}}
\]  

(3.17)

\( I_{on} \) is proportional to \( A_{swt} \). So Equation 3.17 tells that for a fixed \( T_{WUT} \), \( A_{swt} \) is a linear function of the body capacitance of the circuit. When the circuit is partitioned into \( M \) clusters, the total body capacitance of each cluster does not change. Hence the summation of the area of all switch transistors does not change either. So the area and power cost of RBB switch transistors are roughly constants, despite of granularity.

**Routing Cost for Bias Voltage Rails**

Since each cluster enters RBB mode separately, it requires a pair of virtual bias rails. For a standard cell design, this will just be a pair of metal strips between each row of cells [45]. For \( M \)-way clustering, \( M \) pairs of metal strips are required between each row of cells. For example, Figure 3.10 demonstrates that for a 2-way partition, the routing area is simply doubled. (Only NMOS is shown here.) Denote the routing area cost for a pair of virtual bias rails as \( A_{ral} \). The total area cost of a \( M \)-way partition can be estimated as:

\[
A_M = A_{swt} + M \cdot A_{ral}
\]  

(3.18)

![Figure 3.10: Routing Cost of Bias Volt. Rails for A 2-way Partition](image)

**3.3 Design Automation For Determining The Optimum Granularity**

In this section, we first formulate the optimum granularity problem. Then we implement a heuristic algorithm to obtain a near-optimum granularity for any given circuit.
3.3.1 Problem Formulation

As discussed in Section 3.3, the power consumption of control circuits remains roughly the same, despite of granularity, while the area consumption is a linear function of granularity. So the trade-off between leakage saving and control cost finally comes down to the trade-off between energy saving and area consumption. In order to make this trade-off, weights need to be assigned to energy and area. Assume that the cost of unit energy (area) consumption is \( W_E \) \( (W_A) \). The optimum granularity \( (M_{opt}) \) is determined by:

\[
\text{Maximize} \quad \frac{\text{MAX}(E_{M_{opt}+1}) \cdot W_E}{A_{M_{opt}+1} \cdot W_A} < 1 \leq \frac{\text{MAX}(E_{M_{opt}}) \cdot W_E}{A_{M_{opt}} \cdot W_A}
\] (3.19)

where \( \text{MAX}(E_{M_{opt}+1}) \) \( (\text{MAX}(E_{M_{opt}})) \) is the maximum net leakage saving that can be achieved with granularity \( M_{opt}+1 \) \( (M_{opt}) \). \( A_{M_{opt}+1} \) \( (A_{M_{opt}}) \) is the area cost of granularity \( M_{opt}+1 \) \( (M_{opt}) \) in Equation 3.18. Equation 3.19 guarantees that the optimum granularity is the finest granularity, whose weighted leakage saving is larger than its weighted area cost. To obtain \( \text{MAX}(E_M) \), we need to know how to achieve the maximum net leakage saving for a given granularity \( M \). We call this as the optimum partition problem.

We present an ILP formulation for the optimum partition problem based on Equation 3.16. Assume that the circuit is partitioned into \( M \) clusters. Use variable \( X_{gm} \) to denote the assignment of gate \( g \) in cluster \( m \). Specifically,

\[
X_{gm} = \begin{cases} 
1 & \text{if } g \in m \\
0 & \text{if } g \notin m 
\end{cases}
\] (3.20)

The optimization goal is to maximize \( E_M \) in Equation 3.16:

\[
\text{Maximize} \quad \sum_g \sum_m \tilde{E}_g(TWR^{N_m}, B_m) \cdot X_{gm}
\]

\[
B_m = \sum_g S_g(T)X_{gm} | T = 1
\]

\[
N_m = \sum_n \bigcup_g D_g(n) \cdot X_{gm}
\]

\[
\text{Constraint} : \quad \sum_g X_{gm} = 1
\]

\[
\text{Determine} : \quad X_{gm}
\]

where \( B_m \) is the EBT of cluster \( m \). \( N_m \) is the number of primary inputs that cluster \( m \) is dependent on. It can be calculated by \( D_g(n) \). \( D_g(n) \) is a dependent vector of gate \( g \):

\[
D_g(n) = \begin{cases} 
1 & \text{if } g \text{ is dependent on P.I. } n \\
0 & \text{if } g \text{ is not dependent on P.I. } n 
\end{cases}
\] (3.22)
3.3.2 Algorithm to Find A Near-optimum Granularity

ILP formulation is guaranteed to produce the optimal solution. But it is time consuming for large circuits. Instead, a simulated annealing (SA) based heuristic algorithm can be used to obtain near-optimum results. The maximization goal in Equation 3.21 can be simply used to generate the cost function. However, one problem in doing so is that Equation 3.21 tries to optimize both temporal and spatial idleness exploitation. This may cause SA long time to converge, since a good partition for temporal idleness exploitation may not be suitable for spatial idleness exploitation. To address this issue, our algorithm will only optimize either temporal or spatial idleness exploitation at a time. Then whichever partition yields the better leakage saving will be accepted. The second feature of our algorithm is using the divide-and-conquer strategy. Once a \( M \)-way partition is generated, it will be fixed. The \( M+1 \) partitioning will be performed based on the existing \( M \)-way partition. Finally, our algorithm to obtain a near-optimum granularity is as follows.

The first and second parameters of SA are the initial partition and the best partition found, respectively. The third and forth parameters are the idle probability and EBT of each gate in Equation 3.21. By fixing the third parameter as \( R_{ckt}^{N} \), SA maximizes temporal idleness exploitation. By fixing the forth parameter as \( B_{ckt} \), SA maximizes spatial idleness exploitation.

3.4 Experimental Results

We conduct HSPICE on ISCAS85 benchmark circuits with 32nm predictive technology [41]. The clock speed is 1GHz. The bias voltages for \( V_{th} \)-hopping are set to enable \( 25 \times \) total leakage reduction. The switch transistors are sized to ensure that the wake-up time of each circuit is 0.5 clock cycle. For each benchmark, 1000 random input patterns are applied. These patterns are generated with 70% idleness probability for all primary inputs \( (R_{ckt}^{N}=0.7) \). The temperature is set to be 110 Celsius to emulate the runtime temperature.

Table 3.1 shows the experimental result of 10 benchmark circuits with different granularity \( (M=1-5) \). For each granularity \( M \), ‘T’ means that a partition optimizing temporal idleness exploitation (temporal cut) is accepted as the \( M \)th partition. ‘S’ means that a partition optimizing spatial idleness exploitation (spatial cut) is accepted. The percentages shown in Table 3.1 is the net leakage saving percentage. It is calculated by dividing the simulated net leakage saving by the simulated original leakage. The last column calculates the improvement of net leakage saving for
Algorithm 2: Determine A Near-optimum Granularity

INPUT: Cell Library; Technology; Gate-level Design; Input Statistics; $W_E, W_A$

OUTPUT: The Optimum Granularity ($M_{opt}$) and the Optimum Partition

———Derive RBB Leakage Saving Model———-
Characterize $K_s, K_t$ from Technology;
Characterize $\hat{I}_s, \hat{I}_b, C_b, B_g$ from Cell Library;
Obtain $P_u, R$ from Input Statistics;
Obtain $D_g$ from Gate-level Design;
Derive $E'_g(T, B_{swt})$;

———Determine the Optimum Granularity———-

$M = 1$;
$MAX(E_M) = \sum_g E'_g(T, R^{N_{ckt}}, B_{ckt})$;
$A_M = A_{swt}$;
InitialPar $\rightarrow$ AcceptedPar;
while $\frac{MAX(E_M) \cdot W_E}{A_M \cdot W_A} \geq 1$ do

$M + 1$;
$MAX(E_M^{tmp}) = S. A. (AcceptedPar, TmpPar, R^{N_{ckt}}, B_m)$;
$MAX(E_M^{spa}) = S. A. (AcceptedPar, SpaPar, R^{N_{m}}, B_{ckt})$;
if $MAX(E_M^{tmp}) \geq MAX(E_M^{spa})$ then

TmpPar $\rightarrow$ AcceptedPar;

$MAX(E_M) = MAX(E_M^{tmp})$;
end if
else
SpaPar $\rightarrow$ AcceptedPar;
$MAX(E_M) = MAX(E_M^{spa})$;
end if

$A_M = A_{swt} + A_{ral} \times M$;
end while

FOUND: $M_{opt} = M - 1$;

5-way partition ($M=5$).

In Table 3.1, 5-way partition can achieve 14% to 47% improvement on the net leakage saving. A single temporal cut can achieve up to 10% improvement (C6288). A single spatial cut can achieve
Table 3.1: Experimental Results of $V_{th}$-hopping with $M=1-5$

<table>
<thead>
<tr>
<th>BM. Cnts.</th>
<th>Gate</th>
<th>M=1  Cnts.</th>
<th>M=2</th>
<th>M=3</th>
<th>M=4</th>
<th>M=5</th>
<th>M=5</th>
<th>Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>160</td>
<td>11.4% S 24.8%</td>
<td>11%</td>
<td>32.0%</td>
<td>34.8%</td>
<td>37.7%</td>
<td>39.4%</td>
<td>21%</td>
</tr>
<tr>
<td>C499</td>
<td>202</td>
<td>23.5% T 32.3%</td>
<td>33.6%</td>
<td>35.2%</td>
<td>37.6%</td>
<td>14%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C880</td>
<td>383</td>
<td>17.7% S 28.5%</td>
<td>32.4%</td>
<td>34.8%</td>
<td>36.6%</td>
<td>19%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1355</td>
<td>546</td>
<td>9.2% T 13.9%</td>
<td>17.4%</td>
<td>20.6%</td>
<td>22.9%</td>
<td>14%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1908</td>
<td>880</td>
<td>18.4% S 29.1%</td>
<td>34.7%</td>
<td>39.3%</td>
<td>41.4%</td>
<td>23%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2670</td>
<td>1193</td>
<td>20.3% S 34.8%</td>
<td>42.2%</td>
<td>43.3%</td>
<td>47.5%</td>
<td>27%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3540</td>
<td>1669</td>
<td>19.2% S 32.0%</td>
<td>34.8%</td>
<td>37.7%</td>
<td>39.4%</td>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5315</td>
<td>2507</td>
<td>18.8% S 45.2%</td>
<td>60.1%</td>
<td>65.3%</td>
<td>47%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6288</td>
<td>2406</td>
<td>14.5% T 24.5%</td>
<td>29.9%</td>
<td>32.1%</td>
<td>33.6%</td>
<td>19%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7552</td>
<td>3521</td>
<td>22.7% S 37.4%</td>
<td>50.1%</td>
<td>55.5%</td>
<td>57.8%</td>
<td>35%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7552</td>
<td>Area</td>
<td>10%</td>
<td>14%</td>
<td>25%</td>
<td>34%</td>
<td>45%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

up to 26% improvement (C5315). For a runtime circuit, the actual amount of temporal idleness and spatial idleness that we can exploit depends on its input characteristics. To demonstrate this, we change the input idleness probability ($R_{N_{ckt}}$), and observe the variation of improvement of a single cut. As shown in Figure 3.11, the improvement varies significantly with inputs idleness probability. Figure 3.11 also shows that applying RTLC with finer granularity is especially efficient for medium input activities ($0.4 < R_{N_{ckt}} < 0.8$). This is because when the idle probability is high (> 0.8), the circuit already has idleness for the most time. Reducing granularity will not yield significant improvement. When the idle probability is low (< 0.4), the circuit has such a heavy workload that there is not much idleness that can be exploited with finer granularity.

![Figure 3.11: Improv. of Temporal and Spatial Cut VS. Idleness Prob.](image)

The last row shows the total area cost ($A_{M}$) of each granularity of C7552. The area costs of
other benchmark circuits are similar. Figure 3.12 clearly shows the leakage saving improvement versus extra area cost with different granularity on C7552. Given $W_E$ and $W_A$, we can calculate the optimum granularity using Equation 3.19. However due to the heuristic nature of our algorithm, the results can only be near-optimum.

![Figure 3.12: Imprv. of Leakage Saving VS. Extra Area Cost -C7552](image)

3.5 Conclusion and Future Work

In this chapter, we studies aggressive runtime leakage control for combinational circuits. Different from conventional coarse-grained and fine-grained approaches, our method aims at seeking the optimum granularity to balance the leakage saving and control cost. The key to make this trade-off is the quantification of leakage saving and control cost with different granularity. So our first contribution is the model derivation, as well as the formulation of the optimum granularity problem. Our second contribution is the proposition of two methods to aggressively exploit the temporal and spatial idleness of a runtime circuit. Experimental results have proven the effectiveness of these two methods. An algorithm has been implemented to automate these two methods and obtain a near-optimum granularity. Future work may also include how to apply this method to sequential circuits.
Chapter 4

Run-time Leakage Control for Active-mode Circuit

Most of the current runtime leakage control techniques change the circuit state only when they are in standby mode. However as the technology scales down, more aggressive leakage reduction techniques are required. As a promising technique, runtime active leakage power reduction (RALC) has drawn more attention recently [38, 36, 33, 56, 57, 58, 55, 59]. As shown in Figure 4.1, the RALC switches the circuit into a low-leakage state once it detects sufficient idleness in circuit workload, even when the circuit is in the active mode. In this way, it is able to exploit more circuit slackness, and thus reduce more leakage power.

Figure 4.1: Run-time Active Mode Leakage Reduction

Specifically, there are two reasons that signify the importance of RALC: 1) With the technology moving ahead, the ratio of leakage versus dynamic power consumption increases quickly. In the latest technologies, leakage even catches up with dynamic power consumption. Right now, leakage energy consumption, even in a few clock cycles, can be quite significant. This makes RALC quite profitable in terms of energy saving. 2) Because subthreshold leakage has a super linear dependency
on temperature, leakage in the active mode is significantly larger due to the higher die temperature in active mode [38]. This makes RALC even more important.

RALC is usually implemented at a finer granularity than existing leakage control techniques. It may cost noneligible area and power overhead. However, there are two other reasons make RALC feasible: 1) Because of the abundant on-chip transistor resources in deep sub-micro technologies, we can now afford to use some area to perform runtime monitoring and control function. 2) Since the dynamic power consumption shrinks with newer technologies, leakage power can be the major source of power overhead. However, since the control of RALC is usually not a timing-critical event, high-$V_{th}$ transistors can be used. This alleviates the power overhead problem of RALC.

Before start to design RALC, we look back to the existing dynamic power reduction techniques. Many of them have been proven to be effective for an active-mode circuit. Examples as clock gating, precomputation and operand isolation, all prevent the circuit from switching when it is considered to be idle in the next clock cycle. Therefore in essence, these techniques and leakage reduction techniques have the common nature: they all look for circuit idleness to enforce the circuit into a low-power mode. This leads us to consider a question: Since dynamic and leakage power reductions have the same nature, can they be applied together on an active-mode circuit?

Researchers have been studying joint dynamic and leakage reduction. For example, Abddollahi et al. [56] proposes the scheme of applying power gating with pre-computation on a n-bit comparator, as shown in Figure 4.2.

![Figure 4.2: Pre-computation and Power Gating](image)

When the control circuit determines that the LSB of the comparator will not be used in the next clock cycle, it disables the flip-flops feeding to LSB to reduce dynamic power. Meanwhile, power gating is applied to the combinational circuit of LSB to reduce leakage power. This scheme appears
to be reasonable in principle, but is not feasible in practices. The dynamic power reduction of the comparator works well. However, for leakage power, this scheme ignores the fact that PG usually require high energy overhead and long time for mode transition. Tschanz et al. [37] have shown that in 130nm technology for an ALU circuit, the EBT and WUT of power gating are 100 clock cycles and 2 cycles, respectively. (Clock speed is 4GHz.) For reverse body biasing, its EBT and WUT are around 100 cycles and 4 cycles, respectively. Now go back to [56]. Assume that the EBT and WUT of power gating are E and W cycles, respectively. Their scheme has the following two problems:

1) Energy overhead problem: In order to gain energy saving, LSB should not be used in the next E+W cycles. This is because LSB requires N cycles to recover from sleep mode, and M cycles to guarantee that energy saving is larger than energy overhead. So if LSB is used within M+N cycles, net energy saving is negative and more energy will be consumed.

2) Wake-up delay problem: Since the WUT of LSB is W cycles, LSB should be notified W cycles before it is used. This imposes an extra requirement for designers. We need to know the workload information of LSB beforehand. Otherwise, the circuit needs to wait for W cycles to be active again.

Thus, we can see that there is a gap between dynamic and leakage power reduction techniques. This gap is caused by the energy overhead problem and wake-up delay problem of current leakage reduction techniques. These two problems make the joint dynamic and leakage reduction on active-mode circuits not feasible. Both of them need to be addressed for a practical joint reduction scheme. In [56], Abdollahi et al. considered the energy overhead problem, and proposed a method to avoid power gating switching for short idleness. But their method did not address the wake-up delay problem. And since their method requires series of D-flipflop to be inserted before the power gating controller, it incurs large area and power overhead. Min et al. [57] studied the joint application of clock gating and power gating. The proposed “ZSCCMOS” targets to solve the wake-up delay problem. However, their methods did not consider the energy overhead problem. Similar studies attacking wake-up delay problem can be found in [58, 55, 59].

In this chapter, we aim at forging a practical solution for RALC. In the first section, we will compare the candidate techniques for RALC. Especially, we focus on PG and RBB and derive four figures-of-merit to quantitatively compare them. Finally, we find out RBB to be a better candidate technique than PG due to multiple reasons, such as energy overhead problem, isolation cell problem and data retention problem. In Section Two, we propose a new leakage reduction technique called
“Selective Light $V_{th}$ Hopping” (SLITH), which overcomes the energy overhead problem and wake-up time problem, while maintaining the effectiveness of leakage reduction. SLITH bridges the gap between dynamic and leakage power reduction. It enables RALC and can be applied together with most dynamic power reduction techniques during runtime.

4.1 Candidate Techniques Comparison for RALC

There are several runtime leakage reduction techniques available: input vector control, power gating and reverse body biasing. We will compare them from various aspects to identify the best one for RALC.

It is evident that input vector control is the least effective one among all runtime leakage reduction techniques [11]. Furthermore, applying input vector to a circuit incurs large dynamic energy overhead. So it is first excluded from our candidate list.

Both PG and RBB are considered to be effective leakage reduction techniques [11]. They should be compared to see which one is more suitable for RALC. As shown in [37], the EBT values of both PG and RBB are much larger than their WUT values. This indicates that energy overhead problem is more stringent for both of them. So our comparison will focus on study PG and RBB from energy perspective. Our comparison is based on the energy saving models derived in Chapter Two.

4.1.1 Four Figures-of-merit to Compare PG and RBB

Since RALC performs state transition more frequently, the technique that implements it should have small energy overhead and quick leakage reduction ability. To precisely compare the efficacy of PG and RBB, we define four figures-of-merit of RALC.

1) Energy overhead

The energy overhead of PG is caused by switching the footer: $C_F V_{DD}^2$. For RBB, this overhead has two components. One is for switching the control transistors: $C_C V_p^2$. The other is for charging the substrate capacitance: $E_b$ in Equation 2.62. The values of $C_F$, $C_C$ and $C_b$ and their corresponding energy consumption of benchmark circuit C1355 [12] is shown in Table 4.1. $V_p$ and $C_C$ of RBB are chosen to be the optimum values. We can observe that a) $C_C$ is usually very small
because substrate charging does not require high current. b) Although \( C_b \) is much larger than \( C_F \), \( E_b \) is smaller than \( E_F \) because \( C_b \) is only charged up for 0.2v in this case. c) In general, RBB has less overhead than PG if optimum design points are met.

<table>
<thead>
<tr>
<th>BM.</th>
<th>PG ( V_{P}(V) )</th>
<th>RBB ( V_{D,D}=0.9V )</th>
<th>PG ( C_F ) ( \mu )fF</th>
<th>RBB ( C_F ) ( \mu )fF</th>
<th>PG ( E_F ) (E-15J)</th>
<th>RBB ( E_F ) (E-15J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32nm</td>
<td>1.08v</td>
<td>5.4 327</td>
<td>6.3 19.9 26</td>
<td>56</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>45nm</td>
<td>1.12v</td>
<td>7.7 459</td>
<td>9.6 27.7 37</td>
<td>107</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>65nm</td>
<td>1.11v</td>
<td>3.4 725</td>
<td>4.3 42.6 46</td>
<td>179</td>
<td>145</td>
<td></td>
</tr>
</tbody>
</table>

2) Current Injection Speed

Both PG and RBB essentially inject current to one of the transistor terminals and change the terminal voltage. Leakage current is then reduced as a result of terminal voltage changes. So the current injection speed impacts the energy saving speed.

For PG, the subthreshold leakage current is injected into the circuit and raises the internal nodes voltages. Its injection speed is controlled by the value of leakage current \( I_{sub} \). For RBB, the bias voltage source injects the current into substrate. Its injection speed is controlled by the value of on-current \( I_{on} \) of the control transistor \( P_1 \), which is usually larger than \( I_{sub} \). However with PG, as we describe in Section II.C, an important phenomenon, circuit self-discharge, causes instant leakage reduction, and thus boosts its energy saving. According to our experiments, this instant reduction can be 30% to 50%. This phenomenon partially compensates the slow current injection speed of PG.

3) Leakage Reduction Rate

With the change of transistor terminal voltages in the low-leakage state, the leakage current reduces. According to Equation 2.16, the PG leakage reduction rate relies on the exponent \( K \). According to Equation 2.48, the RBB leakage reduction rate relies on \( B_s \). Hence \( K \) and \( B_s \) represent the efficiency of turning current injection to leakage reduction. Table 4.2 compares the \( K \) and \( B_s \) for a single PMOS or NMOS.

4) Stabilized Leakage Reduction Ratio

If the circuit stays in the low-leakage state for a long period of time, the final stabilized leakage
value becomes critical for energy saving. For PG, this stabilized value depends on the $V_{th}$ of the footer. For RBB, this value depends on the value of the bias voltage $V_P$. In our experiments, since we only use the PMOS RBB, the leakage reduction ratio is roughly $2 \times$, while PG is over $30 \times$.

In conclusion, Table 4.3 compares the four figures-of-merit of PG and RBB. It can be concluded from the table that RBB is suitable for short idleness exploitation due to lower overhead and faster current injection speed, while PG is more suitable for long idleness exploitation due to its high leakage reduction rate and stabilized leakage reduction ratio.

### Table 4.3: Four Figures-of-merit of PG and RBB

<table>
<thead>
<tr>
<th></th>
<th>Short Term</th>
<th>Long Term</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Energy Overhead</td>
<td>Current Injection Speed</td>
</tr>
<tr>
<td>PG</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>RBB</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

### 4.1.2 Experiments to Compare PG and RBB

In our experiments, footers are inserted into the benchmark circuits to implement ground gating. The footer size is designed to be equal to the total NMOS width of the circuit. Since the footer is large, it requires an non-negligible driving circuit. To emulate the energy overhead of the driving circuit, we double the switching energy of the footer to be conservative.

The PMOS RBB is implemented as illustrated in Figure 2.16. The bias voltage and size of the control transistors are set to enable the optimum energy saving as discussed in Chapter One. $P_1$ and $P_2$ are of the same size for simplicity. Similar to PG, we double the switching energy.

We conducted experiments to compare the energy saving of PG and RBB for the same circuit in the same state. RBB is designed to have maximum energy saving at the EBT of PG. Figure 4.3
shows the comparison of 65nm C1355. As shown, RBB has better energy saving ability than PG before 19.1ns, due to its small energy overhead. However after 19.1ns, the energy saving of PG catches up with RBB, because PG has higher stabilized leakage reduction ratio. The EBT of PG and RBB in this case is 13.1ns and 5.8ns, respectively. Table 4.4 shows the EBT comparison for each benchmark circuit. We can observe that RBB with optimum design point has 18% (32nm), 34% (45nm) and 52% (65nm) of improvement on EBT over PG. This makes RBB more suitable for short idleness exploitation.

![Figure 4.3: Energy Saving Comparison of PG and RBB on 65nm C1355](image)

**Table 4.4: Energy Breakeven Time (ns) of PG and RBB.**

<table>
<thead>
<tr>
<th>Ckt.</th>
<th>32nm PG RBB</th>
<th>45nm PG RBB</th>
<th>65nm PG RBB</th>
<th>Ckt.</th>
<th>32nm PG RBB</th>
<th>45nm PG RBB</th>
<th>65nm PG RBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>2.3 1.1</td>
<td>4.4 2.7</td>
<td>16.2 7n.3</td>
<td>C2670</td>
<td>1.4 1.2</td>
<td>2.7 1.8</td>
<td>13.0 6.0</td>
</tr>
<tr>
<td>C499</td>
<td>1.6 1.5</td>
<td>4.0 2.6</td>
<td>14.4 7.0</td>
<td>C3540</td>
<td>1.5 1.2</td>
<td>3.7 2.2</td>
<td>13.6 6.4</td>
</tr>
<tr>
<td>C880</td>
<td>1.4 1.2</td>
<td>3.5 2.2</td>
<td>13.2 6.1</td>
<td>C5315</td>
<td>1.5 1.2</td>
<td>3.7 2.2</td>
<td>14.1 6.4</td>
</tr>
<tr>
<td>C1355</td>
<td>1.4 1.2</td>
<td>3.3 2.3</td>
<td>13.1 5.8</td>
<td>C6288</td>
<td>2.3 2.2</td>
<td>4.5 3.7</td>
<td>20.0 11.2</td>
</tr>
<tr>
<td>C1908</td>
<td>1.3 1.2</td>
<td>3.3 2.3</td>
<td>12.6 6.5</td>
<td>C7552</td>
<td>1.4 1.2</td>
<td>3.5 2.2</td>
<td>13.1 6.3</td>
</tr>
<tr>
<td>PG-RBB</td>
<td>18%</td>
<td>34%</td>
<td>52%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.1.3 Overall Considerations

Conclusively, RBB is more suitable for RALC than PG for the following reasons:

1) As we have demonstrated, RBB is more suitable for short-term, active leakage reduction because of its low energy overhead and fast leakage reduction speed.

2) PG has the output floating problem. To prevent this, isolation cells need to be inserted at the
interface of different PG domains [53]. This adds additional cost to implement RALC.

3) Unlike PG, RBB does not have the state retention problem. This makes RBB especially suitable to be applied on active-mode circuit.

There are two types of RBB implementation. One is $V_{th}$ hopping, as we have shown in the previous discussion. Another is Dynamic $V_{th}$ scaling, which tunes the body voltage in a continuous fashion to change $V_{th}$. Dynamic $V_{th}$ scaling can be used to reduce both dynamic and leakage power [53]. However, it requires the knowledge of future workload, which is not always available. So in our study, we focus on $V_{th}$-hopping to be our primary technique to implement RALC.

4.2 Selective Light $V_{th}$ Hopping (SLITH)

In order to solve the energy overhead problem and the wake-up delay problem, any practical RALC implementation needs to reduce its EBT and WUT. If the sum of EBT and WUT can be reduced to one clock cycle, the two problems will be completed solved. A joint dynamic and leakage power reduction scheme is then possible. In this section, we will show how SLITH helps to achieve the reduction of EBT and WUT, to as low as one clock cycle. We call the summation of EBT and WUT as E&W.

SLITH is based on $V_{th}$ Hopping. We have modeled the basic $V_{th}$ Hopping scheme in Chapter Two and Three. In order to different parts of the model, and to make this chapter self-contained, we explain the modeling briefly in the following.

4.2.1 Basic $V_{th}$ Hopping Modeling

The basic $V_{th}$ Hopping is modeled in Chapter One as shown in Figure 4.4. In Figure 4.4, $V_P$ ($V_{DD} + \Delta V_P$) is the PMOS biasing voltage source, and $V_b$ is the PMOS body voltage. $R_1$ and $R_2$ are the equivalent resistors of transistor $S_1$ and $S_2$, respectively. $C_{BW}$ and $C_{BG}$ are the equivalent capacitances of PMOS body-to-VDD and body-to-GND, respectively. $I_{TV}$ and $I_{TG}$ are the BTBT leakage. $I_S$ is the subthreshold leakage. $C_1$ and $C_2$ (not shown in Figure 4.4) are the equivalent gate capacitances of transistor $S_1$ and $S_2$, respectively.

When $V_{th}$ hopping is applied, $R_1$ will be switched on, and $V_P$ starts to charge up the body
capacitances $C_bV$ and $C_bG$. The energy overhead for switching $S_1$ and $S_2$ is:

$$E_{swt} = V_P^2(C_1 + C_2)$$  \hspace{1cm} (4.1)

The charging overhead to fully charge up the PMOS body is:

$$E_{charging} = (C_bG + C_bV)\Delta V_P^2 + C_bG V_{DD} \Delta V_P$$  \hspace{1cm} (4.2)

When the body is fully charged up to $V_P$, the total leakage ($I$) will be reduced to:

$$I = e^{B_s \Delta V_P} \tilde{I}_S + e^{-B_t \Delta V_P} \tilde{I}_T$$  \hspace{1cm} (4.3)

where $\tilde{I}_S$ and $\tilde{I}_T$ are the original subthreshold leakage and BTBT leakage of the circuit without $V_{th}$ hopping. $B_s$ and $B_t$ are the reduction exponents for subthreshold and BTBT leakage, respectively [43]. The energy saving per unit time is:

$$E_{saving/ut} = (\tilde{I}_S + \tilde{I}_T - I) V_{DD}$$  \hspace{1cm} (4.4)

Thus, the EBT of $V_{th}$ hopping can be calculated by:

$$T_{EBT} = \frac{E_{swt} + E_{charging}}{E_{saving/ut}}$$  \hspace{1cm} (4.5)

According to [55], The WUT can be calculated by the total charge stored in the body capacitance divided by the on-current ($I_2$) of the discharging transistor $R_2$:

$$T_{WUT} = \frac{(C_bG + C_bV)\Delta V_P}{I_2}$$  \hspace{1cm} (4.6)

Finally E&W can be calculated by:

$$T_{E&W} = T_{EBT} + T_{WUT}$$  \hspace{1cm} (4.7)
4.2.2 Light $V_{th}$ Hopping

Although reverse body biasing reaches the optimum biasing voltage when the sum of subthreshold and BTBT leakage is minimized [43], this voltage is usually too high for active leakage reduction, for which the switching overhead is not negligible. For active leakage reduction, in Chapter Two we have demonstrated that choosing small biasing voltage causes slight degradation on leakage reduction, but significantly reduces energy overhead. Thus, it is able to reduce $E$ & $W$. We call the $V_{th}$ hopping with small biasing voltage as “light” $V_{th}$ hopping.

Figure 4.5 shows our experiment of applying light $V_{th}$ hopping on an inverter chain with 32nm technology. Table 4.5 compares its results with transitional body biasing technique. We can observe that the EBT of light $V_{th}$ hopping is reduced by nearly 20 times. WUT is reduced by 2.5 times. The trade-off is that leakage reduction is degraded by 14%. However, 85% is still effective for active leakage reduction. The optimum biasing voltage will be discussed in Section II.C.

![Figure 4.5: Light $V_{th}$ Hopping on Inverter Chain](image)

<table>
<thead>
<tr>
<th>V$_P$ (V)</th>
<th>V$_N$ (V)</th>
<th>-Leak%</th>
<th>EBT (ns)</th>
<th>WUT (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>3.0</td>
<td>-2.0</td>
<td>99%</td>
<td>10.9</td>
</tr>
<tr>
<td>Light</td>
<td>1.3</td>
<td>-0.4</td>
<td>85%</td>
<td>0.48</td>
</tr>
</tbody>
</table>

4.2.3 Selective $V_{th}$ Hopping

When $V_{th}$ hopping is applied, the leakage energy saving and body charging overhead of each gate in the circuit can vary significantly. Due to this variation, some gates in the circuit, which have high saving and low overhead, are suitable for both active and standby leakage reduction. On the contrary, some gates with low saving and high overhead, are only suitable for standby leakage reduction.

In order to demonstrate the variation of leakage saving versus body charging overhead, we apply
$V_{th}$ hopping on an ALU (C880 of ISCAS85) with 64 typical input patterns. Then the EBT value of each gate in the ALU is calculated. Figure 4.6 shows the statistics of the EBT value of each gate. The X-axis is EBT value. The Y-axis is the count (logarithmic) of the gates, whose EBT falls into the same range.

![Gate EBT Variation on C880 with 32nm Technology](image)

In Figure 4.6, the EBT of each gate can vary up to 80 times, from 3 to 230 clock cycles. This indicates that by applying $V_{th}$ hopping, some gates can save energy after 3 cycles. They are suitable for short-term, active leakage reduction. But some gates can only save energy after 230 cycles. They are suitable for long-term, standby leakage reduction.

We can apply $V_{th}$ hopping to a subset of the circuit only including gates with small EBT values. In this way, the EBT value of this selected potion will be significantly reduced. The WUT will be reduced as well. This is because the gates with small EBT values usually have small body area so that their discharging time will be short. We call this method as “selective” $V_{th}$ hopping. The key of this method is to select more gates as possible, while keep the E&W of the selected portion within the constraint.

Another phenomenon in a runtime circuit that can help to reduce E&W is called: “determinability of state”. During runtime, when a dynamic power reduction technique is applied to a circuit, the states of some gates (transistors) in the circuit will be determined. If the state of a gate is known, we can selectively apply $V_{th}$ hopping to a specific part of the gate to reduce the leakage current of the whole gate.

Take a clock-gated D-flipflop for example. Since the clock is gated during the clock low period, we know the states of some transistors in the D-flipflop, as shown in Figure 4.7. Instead of applying $V_{th}$ hopping to all the transistors, we only apply it on a few selected ones (the shade ones in Figure 4.7). The body of the remaining transistors is connected to normal body voltages. Experimental
results show that after this meticulous selection, the total body area is reduced by 67%. The leakage reduction ratio decreases by 28%. However, this scheme results in 2.2X reduction in EBT, and 3X reduction in WUT of the D-flipflop.

For a standard cell design, the D-flipflop with selective $V_{th}$ hopping in Figure 4.7 can be added into the cell library and replace the original D-flipflop to improve its ability for active leakage reduction.

4.2.4 Selective Light $V_{th}$ Hopping (SLITH)

Combining “selective” and “light” together, SLITH aims at reducing $E$&$W$, to as low as one clock cycle, while keeps the effectiveness of leakage reduction. Figure 4.8 illustrates the diagram of SLITH based on the basic $V_{th}$ hopping. There are four design parameters in the diagram need to be determined:
1) Discharging transistors \((S_2, S_4)\): \(S_2\) and \(S_4\) are designed with large size to guarantee low WUT. However if they are too large, EBT will increase due to large overhead.

2) Charging transistors \((S_1, S_3)\): \(S_1\) and \(S_3\) control the speed of body charging and also affect EBT and WUT.

3) Biasing voltages \((V_P, -V_N)\): Biasing voltages are the key of SLITH. They control the effectiveness of leakage reduction. They are also the dominating factors affecting EBT and WUT.

4) Gate selection: we want to apply SLITH on more gates to reduce more leakage, meanwhile keep the E\&W of the selected gates within constraint. Note that for a single gate, its PUN and PDN can receive SLITH separately, as shown in the D-flipflop example in Figure 4.7. However, the difference with the D-flipflop example is that handcrafting SLITH transistor by transistor is not allowed here. To facilitate the design automation for standard cell designs, the minimum granularity of applying SLITH is the PUN or PDN of a gate.

Hence if SLITH is applied to a circuit block, there exists an optimum design point which determines these four parameters. This optimum design point problem can be defined as:

\[
\text{Determine : } S_1, S_2, S_3, S_4, V_P, -V_N, \quad G^N_k = [0, 1], G^P_k = [0, 1] \quad (k = 0..K) \\
\text{Maximize : } E_{\text{saving/ut}} \\
\text{Constraint : } T_{E\&W} \leq M \cdot T_{\text{clk}}
\]

(4.8)

where \(G^N_k\) and \(G^P_k\) represent the selection of the PDN and PUN of gate \(k\), respectively. The PDN of gate \(k\) will be selected if \(G^N_k\) equals to 1. \(K\) is the total number of the gates in the circuit. \(E_{\text{saving/ut}}\) and \(T_{E\&W}\) are modeled in Equation 4.4 and 4.7, respectively. \(T_{\text{clk}}\) is the clock period. So \(M \cdot T_{\text{clk}}\) sets the constraint of E\&W to be \(M\) clock cycles. This optimum design point problem can be solved by a heuristic solver.

A CAD tool has been developed to implement SLITH on any given circuit with any technology. The tool flow is shown in Figure 4.9. The tool first takes cell library to characterize the leakage current \((I_S, I_T\) in Equation 4.4) and body capacitances \((C_{G}, C_{BV}\) in Equation 4.2) of each gate type. Then given a circuit netlist and its typical input patterns, the tool calculates the statistics of the leakage saving and overhead of each gate. Next, given the E\&W constraint \(M\), the tool selects gates to apply SLITH and determines the optimum design point. If \(M\)-cycle E\&W is not achievable, the tool will gradually increase \(M\) \((M = M + 1)\) and perform optimization again.
4.2.5 Experimental Results of SLITH

We conducted HSPICE simulation to verify the effectiveness of SLITH. The simulation runs with 32nm and 45nm predictive technology [41]. The temperature is set to be 110°C to emulate runtime temperature.

We emphasize that best expectation is to achieve one-cycle E&W to enable seamless collaboration of dynamic and leakage power reduction. However, experimental results show that depending on the technology, one-cycle E&W is not always feasible. We will show in Section IV that SLITH with multiple-cycle E&W still works, with the cost of extra area and delay.

Inverter Chain Results

Table 4.6 shows the results of applying SLITH on an inverter chain with both 32nm and 45nm technologies. The first line of the table shows the optimum design point when the E&W constraint is set to be one clock cycle ($M=1$) with 32nm technology. PDN% (PUN%) is the percentage of PDN (PUN), which receives SLITH. +A% is the extra area cost of the charging and discharging transistors $S_1$ to $S_4$. As we can see from the first line, the total active leakage reduction is up to 37%, with 1.7% extra area overhead for one-cycle SLITH.

Figure 4.10.left shows the waveform of N-well voltage in one clock cycle. Figure 4.10.right shows the waveform of subthreshold leakage current in one clock cycle. The pikes are caused by the N-well charging and discharging currents.

The first line of Table 4.6 shows that SLITH is applied to all PDN of the inverter chain, but not
Table 4.6: Optimum SLITH on An Inverter Chain with 32/45nm Technologies

<table>
<thead>
<tr>
<th>Tech.</th>
<th>M</th>
<th>$\Delta V_T$ (V)</th>
<th>$\Delta V_N$ (V)</th>
<th>PDN%</th>
<th>PUN%</th>
<th>+A%</th>
<th>-Leak%</th>
</tr>
</thead>
<tbody>
<tr>
<td>32nm</td>
<td>1</td>
<td>0</td>
<td>-0.26</td>
<td>0%</td>
<td>100%</td>
<td>1.7%</td>
<td>37%</td>
</tr>
<tr>
<td>32nm</td>
<td>2</td>
<td>0.21</td>
<td>-0.31</td>
<td>100%</td>
<td>100%</td>
<td>5.5%</td>
<td>56%</td>
</tr>
<tr>
<td>45nm</td>
<td>2</td>
<td>0</td>
<td>-0.33</td>
<td>0%</td>
<td>100%</td>
<td>0.9%</td>
<td>38%</td>
</tr>
</tbody>
</table>

Figure 4.10: Waveforms of N-well Voltage and Subthreshold Leakage Current of Inverter Chain by Applying One-cycle SLITH

PUN due to the large charging overhead of PMOS transistors. PUN will be included when $M=2$, as shown in the second line of Table 4.6. This is because when $M=2$, the leakage saving in 2 clock cycles is able to compensate the large overhead of PUN.

The first line of Table 4.6 also indicates that total leakage reduction ratio should be 37% with -0.26V NMOS reverse body biasing. However, it is not true if the inverter chain stays in SLITH mode for only one cycle. Our optimization goal is to maximize the leakage reduction ratio, while maintains one-cycle E&W. Hence at the first cycle, the energy saving will be dedicated to compensate the energy overhead of mode transition. The net energy saving of the first cycles is close to zero (9% in this case). The strength of SLITH will be shown after two cycles. Figure 4.11 shows the net energy saving percentage after SLITH is applied. The net energy saving at the second cycle is 25%. Similarly in the following experiments, if SLITH is optimized for M-cycle E&W, it yields significant net energy saving only after $M + 1$ cycles.

The second line in Table 4.6 shows the optimum design point when the E&W constraint is set to be two clock cycles ($M=2$). This time SLITH will be applied to both PDN and PUN. The biasing voltage of PUN is still lower than PDN due to the large charging overhead of PUN. The total active leakage reduction is improved to up to 56%, with 5.5% extra area overhead.
With 45nm technology, no energy saving can be achieved when M=1. This is because the dynamic to leakage power consumption ratio is high for 45nm technology. Applying SLITH for only one cycle causes larger energy overhead than leakage saving. SLITH is still promising with 45nm technology when \( M = 2 \). It can achieve up to 38% leakage reduction, with 0.8% extra area overhead, as shown in the third line in Table 4.6.

**Extensive Experiments Results**

Table 4.7 and 4.8 show the results of applying SLITH on different circuits with 32nm and 45nm technology, respectively. As we can observe, one-cycle SLITH in 32nm can achieve 9% to 37% leakage reduction. Two-cycle SLITH can achieve 40% to 63% leakage reduction. With 45nm technology, one-cycle SLITH is not able to achieve net energy saving. However two-cycle SLITH with 45nm technology can achieve 16% to 42% leakage reduction.

### 4.3 Joint Dynamic and Leakage Power Reduction With SLITH

One-cycle SLITH can seamlessly collaborate with any dynamic power reduction technique, such as clock gating, pre-computation and operand isolation etc.. Whenever a dynamic power reduction technique identifies circuit slackness during runtime, it triggers the corresponding mechanism to prevent circuit switching. Meanwhile, SLITH can be applied to the circuit to reduce its leakage power. Since the sum of EBT and WUT is controlled within one cycle, the two problems stated in Section I are no longer hurdles. Even if the circuit is reactivated after one clock cycle, the net energy saving is guaranteed to be positive. And its body voltage is guaranteed to have recovered to the normal voltage. Figure 4.12 shows the diagram of joint power reduction by pre-computation and one-cycle SLITH.

One issue of this scheme is using the clock enable signal with normal swing voltage (GND to
Table 4.7: One-cycle and Two-cycle SLITH with 32nm Technology

<table>
<thead>
<tr>
<th>Circuit</th>
<th>M</th>
<th>$\Delta V_P$</th>
<th>$\Delta V_N$</th>
<th>PDN%</th>
<th>PUN%</th>
<th>$+A%$</th>
<th>-Leak%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv. Chain</td>
<td>1</td>
<td>0</td>
<td>-0.26</td>
<td>0%</td>
<td>100%</td>
<td>1.7%</td>
<td>37%</td>
</tr>
<tr>
<td>D-flipflop</td>
<td>1</td>
<td>0</td>
<td>-0.2</td>
<td>0%</td>
<td>95%</td>
<td>1.2%</td>
<td>26%</td>
</tr>
<tr>
<td>P. Decoder(C432)</td>
<td>1</td>
<td>0</td>
<td>-0.18</td>
<td>0%</td>
<td>27%</td>
<td>0.3%</td>
<td>9%</td>
</tr>
<tr>
<td>ECAT(C499)</td>
<td>1</td>
<td>0</td>
<td>-0.29</td>
<td>0%</td>
<td>72%</td>
<td>0.6%</td>
<td>30%</td>
</tr>
<tr>
<td>ALU(C880)</td>
<td>1</td>
<td>0</td>
<td>-0.29</td>
<td>0%</td>
<td>73%</td>
<td>0.6%</td>
<td>27%</td>
</tr>
<tr>
<td>Multiplier(C6288)</td>
<td>1</td>
<td>0</td>
<td>-0.22</td>
<td>0%</td>
<td>83%</td>
<td>1%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Table 4.8: Two-cycle SLITH with 45nm Technology

<table>
<thead>
<tr>
<th>Circuit</th>
<th>M</th>
<th>$\Delta V_P$</th>
<th>$\Delta V_N$</th>
<th>PDN%</th>
<th>PUN%</th>
<th>$+A%$</th>
<th>-Leak%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv. Chain</td>
<td>2</td>
<td>0.21</td>
<td>-0.31</td>
<td>100%</td>
<td>100%</td>
<td>5.5%</td>
<td>56%</td>
</tr>
<tr>
<td>D-flipflop</td>
<td>2</td>
<td>0.26</td>
<td>-0.35</td>
<td>63%</td>
<td>95%</td>
<td>2.5%</td>
<td>50%</td>
</tr>
<tr>
<td>P. Decoder(C432)</td>
<td>2</td>
<td>0.19</td>
<td>-0.25</td>
<td>41%</td>
<td>93%</td>
<td>1.7%</td>
<td>40%</td>
</tr>
<tr>
<td>ECAT(C499)</td>
<td>2</td>
<td>0.22</td>
<td>-0.36</td>
<td>27%</td>
<td>97%</td>
<td>1.4%</td>
<td>51%</td>
</tr>
<tr>
<td>ALU(C880)</td>
<td>2</td>
<td>0.19</td>
<td>-0.30</td>
<td>86%</td>
<td>98%</td>
<td>3.7%</td>
<td>63%</td>
</tr>
<tr>
<td>Multiplier(C6288)</td>
<td>2</td>
<td>0.20</td>
<td>-0.36</td>
<td>46%</td>
<td>100%</td>
<td>1.9%</td>
<td>63%</td>
</tr>
</tbody>
</table>

$V_{DD}$) to control the charging and discharging transistors, which have higher swing voltage (GND to $V_P$ for $S_1$ and $S_2$, $-V_N$ to $V_{DD}$ for $S_3$ and $S_4$). Our solution is to insert a voltage level converter between the clock enable signal and the SLITH control circuit. The voltage level converter is small in size [60] and incurs minor area and power penalty.

Figure 4.13 shows the joint power reduction on a finite-state machine (FSM) by clock gating and one-cycle SLITH. As we can see in dash area 1, SLITH is applied to D-flipflops to reduce the leakage of D-flipflops. Since SLITH does not alter the circuit states, the logic values stored in the D-flipflops are retained. In dash area 2, if the clock-enabling signals from all state registers are zero,
it means the whole FSM will not be active in the next clock cycle. SLITH is then applied on the combinational logic to further reduce leakage power.

Two-cycle SLITH can only be applied when the circuit has at least two cycles of idleness. The key is to know when the circuit will not be used in the next two cycles. This can be easily implemented in a pipeline datapath as shown in Figure 4.14. We use an activity monitor at the datapath input and the first two stages to detect any two-cycle idleness in the workload, and pass it directly to the third stage. Whenever there is two-cycle idleness in the workload, the clock gating and SLITH will be triggered and applied to the pipeline stage by stage, starting from the third stage.
4.4 Conclusion and Future Work

In this chapter, we study the importance and feasibility of RALC. We first compare the effectiveness of all candidate techniques for implementing RALC. Especially, we derive four figures-of-merit to quantitatively compare the energy saving of PG and RBB. Our finally choose one type of RBB technique, $V_{th}$ hopping as our target technique to implement RALC.

In order to reach a practical solution for RALC, we consider the possibility of having joint dynamic and leakage reduction on active-mode circuit. To do so, we identify that the obstacle is caused by the energy overhead problem and wake-up time problem of current leakage reduction technique. Therefore we propose a new leakage reduction technique based on $V_{th}$ hopping: SLITH. It is able to reduce the EBT and WUT, to as low as one clock cycle, while maintaining the effectiveness of leakage reduction. Therefore, it bridges the gap and enables joint dynamic and leakage power reduction. SLITH can be used together with most of the dynamic power reduction techniques. Experimental results show that it yields significant active leakage power reduction.

Future research may include more extensive experimenters on SLITH. Then the next step is to verify the effectiveness of joint dynamic and leakage power reduction using SLITH. Finally if all experimental results turn out to be promising, we should consider how to insert SLITH into the current design flow.
Power mode transition is sometimes accompanied with reliability problems. Delay overhead or rush current can be incurred when the target circuit switches from one mode to another. For leakage control techniques, this usually occurs when the sleeping circuit is woken up [27]. Take a recent power gating application in an AMD multicore design [61] for example. The accumulated charges during sleep period are released to the power grid of the chip at wake-up, forming large rush currents. In a multicore design, it is normal that some cores are active while others are performing power mode transition. The large rush current incurred by the waking-up cores (CoreS in Figure 5.1) can cause temporary voltage drop on the adjacent active cores (CoreA in Figure 5.1), and ground bounce on the whole power grid. The solution to alleviate the rush current problem is to turn on power gates gradually, so that to limit the current injected to the power grid. However, this essentially prolongs the power mode transition time. Thus, a good wake-up scheme makes a trade-off between delay overhead and the rush current.

The wake-up problem for power gating has been extensively studied in recent years. The initial focus was on optimal circuit clustering to minimize wake-up delay given a certain rush current constraint. However, it is evident that DSTN and shared virtual ground design approach have been adopted in most design practice [62]. Since there is only one common virtual ground for all gates, circuit clustering does not alter the wake-up process any more. Hence later on, researchers shifted their focus from circuit clustering to power gates partition and activation sequencing [24, 25, 26, 27, 63, 64]. Almost all previous literatures only studied the wake-up problem for a circuit block, instead of a full chip. Just recently, Jiang et al. [65] presented a wake-up scheduling algorithm for a number
of circuit blocks in a chip with a certain rush current constraint. However, their approach requires a sophisticated central control state machine to arrange the wake-up sequencing of each block at runtime, rendering the scheme less practical. Furthermore, very few existing research analyzed or optimized the physical implementation of wake-up circuit in detail.

In this chapter, we present the first study to consider fast and reliable power mode transition from the chip level, with analysis on detailed physical models and implementations. The study uses the recent ring style power gating for multicore designs as an example, and introduces the concepts and physical designs of two novel techniques, namely current shaping and multi-thread activation. A CAD flow is also presented to automatically design an optimized power gating control circuit using the two novel techniques. The two techniques introduced in this paper are applicable to fast and reliable designs of other on-chip power switches, such as body biasing control or $V_{DD}$ hopping.

This chapter is organized as following. Section 5.2 analyzes the rush current and voltage drop problem from chip level in the context of multicore design. A design rule is proposed for fast power mode transition with guaranteed power integrity. A full chip per-core power gating model is also presented as the foundation for further analysis. Section 5.3 analyzes the three phases of the wake-up process, and presents the ideas and physical optimization of the two novel techniques. The CAD flow is also presented in this section. Section 5.4 compares the experimental results between our approach and two other approaches on generated cores with different sizes. Finally Section 5.5 concludes the chapter.
5.1 Reliable Power Mode Transition

The diagram of ring style implementation of AMD per-core power gating [61] is illustrated in Figure 5.1. Figure 5.2 gives further details on the control circuits of power gates (footers for ground gating). In particular, the control signal \( S_c \) is routed along with the footers. Repeaters are inserted to boost the propagation of the control signal round the core. Drivers are required to drive the footers, which are usually designed to be very large to reduce voltage drop across them. When central power management unit decides to wake up a power-gated core, it releases a wake-up signal. Assume that this signal arrives at the point A in Figure 5.2. Then it propagates towards the arrow direction, and turns on the footers one by one along its way. The control signal returns to A after it has traversed every edge of the core.

![Figure 5.2: Control Circuit of Ring Power Gating](image)

Power grid is usually designed to accommodate average current drawn from the circuit in active mode [66]. Voltage drop on the power grid is caused by two phenomena. The IR drop is caused by delivering current \( I \) through resistive power grid \( R \). The inductive voltage drop is because the high \( di/dt \) of fast switching excites the parasitic inductance \( L \), and thus increases impedance and causes an additional voltage drop. The total voltage drop is the sum of both:

\[
V_{drop} = RI + L \frac{di}{dt}
\]  

(5.1)

In a multicore design, assume that the core under activation (denoted as CoreS) shares a common ground path to the ideal ground with adjacent cores (denoted as CoreA). The resistance of this common path is \( R_{return} \). Denote the average current of CoreA as \( I_{adj} \), and the average current of CoreS in active mode as \( I_{avg} \). By assuming a chip-wide voltage drop threshold as \( \alpha V_{DD} \), the voltage drop equation for CoreA and CoreS in active mode is:

\[
\alpha V_{DD} = R_{return}(I_{avg} + I_{adj}) + L \frac{d(I_{avg} + I_{adj})}{dt}
\]

(5.2)
Footers of CoreS are designed to be large in size to avoid performance lost. When CoreS is sleep mode, its virtual ground voltage is elevated to a certain value ($V_{ini}$). When CoreS is woken up, the virtual ground discharges through the large-size footers. Assume that the footers are designed to incur $\alpha V_{DD}$ voltage drop, and the peak current of CoreS is $I_{peak}$. Then its equivalent resistance ($R_F$) is:

$$R_F = \frac{\beta V_{DD}}{I_{peak}}$$ (5.3)

Hence when the footers are turned on during wake-up, the worst case rush current incurred ($I_{worst}$) is:

$$I_{worst} = \frac{V_{ini}}{R_F} = \frac{V_{ini}}{\beta V_{DD}} I_{peak}$$ (5.4)

Our experiments show that $V_{ini}$ is usually about 50% to 60% $V_{DD}$ value. Assume that $\alpha$ and $\beta$ are both 5%. Thus, Equation 5.4 yields an approximate value of the worst case rush current as $10 \times I_{peak}$. The worst case voltage drop at the common return path of CoreS and CoreA is:

$$V_{worst} = R_{return}(10I_{peak} + I_{adj}) + L \frac{d(10I_{peak} + I_{adj})}{dt}$$ (5.5)

Subtract each side of Equation 5.5 by Equation 5.2:

$$V_{worst} - \alpha V_{DD} = R_{return}(10I_{peak} - I_{avg}) + L \frac{d(10I_{peak} - I_{avg})}{dt}$$ (5.6)

Equation 5.1 gives the extra voltage drop caused by waking up CoreS in the worst case. Since $10I_{peak}$ is much higher than $I_{avg}$, the voltage drop of this worst case is unacceptable, and is very likely to cause the malfunction of CoreA.

One method to guarantee the power integrity of CoreA is to confine the rush current of CoreS to be under $I_{avg}$. In this way, the first term in the right side of Equation will be zero, and the second term will also be negligible if footer is controlled to avoid sharp slope. Calimera et al. propose to limit the footer current to be within $I_{peak}$ [64]. However, we consider that this limit cannot guarantee power integrity. This is because the power grid is usually designed to accommodate average current consumption, while decaps are used to alleviate instantaneous peak current in active mode. So $I_{peak}$ in active mode is provided by both power grid and decaps. However, the wake-up process of a power-gated circuit usually takes multiple clock cycles. In this case, decaps do not have chances to recharge themselves every cycle, and become useless once depleted [67]. Since decaps are ineffective in providing charges during circuit wake-up, using $I_{peak}$ as a rush current constraint is not safe, and can lead to unaffordable voltage drop.
5.2 Fast Activation Techniques

We analyze an ideal wake-up process in three phases: initial boost, current shaping and multithread activation.

5.2.1 Initial Boost

Before \( I_{on} \) reaches the limit of \( I_{avg} \), the footers should be turned on as fast as possible. Denote on-current and equivalent resistor of each footer as \( i_f \) and \( r_f \), respectively. The ideal number (\( N_{ideal} \)) of footers allowed to be turned on is \( I_{avg}/I_{on} \). However, due to the delay in propagating the control signal, when the control signal reaches \( N_{ideal} \), the virtual ground voltage decreases below its original value due to discharge, so that \( I_{on} \) at that point also decreases below \( I_{avg} \), as shown in Figure 5.4. Thus, a few more footers need to be turned on to compensate this current drop. Denote the real number of footers required for \( I_{on} \) to reach \( I_{avg} \) as \( (N_{ini}) \).

![Figure 5.4: Initial Boost](image)

To achieve the goal of turning on \( N_{ini} \) footer as fast as possible, repeaters and drivers before
\( N_{ini} \) should be properly sized. Denote the repeater width of stage \( k \) as \( w^k \), and the delay of stage \( k \) as \( t^k \). Denote the width of the first layer driver at stage \( k \) as \( w^k_{d} \), and the total driver delay of stage \( k \) as \( t^k_{d} \). The optimal sizing problem is:

\[ t^k = 0.69 \frac{r_{\text{rep}} c_{\text{wire}}}{2 w^k} + 0.69 \left( \frac{r_{\text{rep}}}{w^k} + r_{\text{wire}} \right) \left( \frac{1}{2} c_{\text{wire}} + c_{\text{rep}} w^{k+1} \right) \]  

\( 5.7 \)

\[ t^k_{d} = P_{opt} t_{p0} \left( 1 + P_{opt} \sqrt{F^k_d} \right) \]

where \( F^k_d = \frac{W_f}{3 w^k_d} \); \( P_{opt} = \ln(F) \);

\[ t^k_{o} = t^k_{d} + \sum_{n=1}^{k} t^n_{r} \]  

\( 5.9 \)

\[ \text{Minimize} : T_{ini} = \text{Max}(t^1_{o}, t^2_{o}, \ldots, t^{N_{ini}}_{o}) \]  

\( 5.10 \)

\[ v^k_{d} = \frac{(T_{ini} - t^k_{o}) \cdot i_f}{c_{\text{veg}}} \]  

\( 5.11 \)

\[ \text{Constraint} : \frac{V_{ini} - \sum_{k=1}^{N_{ini}} v^k_{d}}{V_{ini}} \cdot N_{ini} \cdot i_f = I_{avg} \]  

\( 5.12 \)

Equation 5.7 calculates the Elmore delay of each repeater stage. \( r_{\text{rep}} \) and \( c_{\text{rep}} \) are the resistance and capacitance of repeater per unit width. \( r_{\text{wire}} \) and \( c_{\text{wire}} \) are the resistance and capacitance of control signal at each stage. Note that Equation 5.7 uses \( \pi \) wire model. Equation 5.8 calculates the total driver delay using optimal inverter chain sizing rule. \( t_{p0} \) denotes the intrinsic delay of a minimal sized inverter. \( P_{opt} \) denotes the optimal number of driver layers. \( F^k_d \) is the effective fanout of the driver chain at stage \( k \). \( t^k_{o} \) in Equation 5.9 represents the actual turn-on time of the footer at stage \( k \). \( T_{ini} \) in Equation 5.10 is the longest turn-on time of all footers before the \( N_{deal} \) stage. \( T_{ini} \) is essentially our optimization goal. \( v^k_{d} \) in Equation 5.11 calculates the virtual ground voltage decrease caused by footer \( k \) before \( T_{ini} \). \( c_{\text{veg}} \) represents the total virtual ground capacitance of CoreS. Finally, Equation 5.12 calculates the total decrease of virtual ground voltage, and calculates the actual total on-current for \( N_{ini} \) footers, and then equate it to the limit of \( I_{avg} \). At the time when \( N_{ini} \) footers is fully turned on, the new virtual ground voltage \( (V_{p1}) \) equals to:

\[ V_{p1} = r_f \cdot \frac{I_{avg}}{N_{ini}} \]  

\( 5.13 \)

5.2.2 Current Shaping

When \( I_{on} \) reaches the limit of \( I_{avg} \), the design goal switches to keep \( I_{on} \) as close to \( I_{avg} \) as possible. To avoid rush current surpassing the limit, the activation of footers should be slowed
down. This can be realized by downsizing repeaters and drivers. The challenge is how to optimally size them to obtain a constant rush current of $I_{avg}$. We introduce a Current Shaping technique to solve this problem.

Group the footers that has been turned on in initial boost together, and denote the current of them as $I_a$. If no more footers are turned on, $I_a$ will drop quadratically, since CoreS can be roughly considered as a huge capacitance discharging through a single resistor. However, suppose we continue to open the remaining footers, and keep the total $I_{on}$ nearly as $I_{avg}$, then the virtual ground voltage ($V_{vg}(t)$) decreases at a linear rate:

$$V_{vg}(t) = \frac{c_{vvg} \cdot V_{p1}}{c_{vvg}} - \frac{I_{avg} \cdot t}{c_{vvg} \cdot r_f}$$

Thus, $I_a$ can also be modeled as a linear function of time.

$$I_a(t) = \frac{V_{vg}(t)}{r_f/N_{ini}} = I_{avg} - \frac{I_{avg} N_{ini}}{c_{vvg} \cdot r_f} \cdot t$$

The total time ($T_{alt}$) for discharging is:

$$T_{alt} = \frac{c_{vvg} \cdot V_{p1}}{I_{avg} \cdot N_{ini}} = \frac{c_{vvg} \cdot r_f}{N_{ini}}$$

In order to shape a flat rush current curve, we need to construct a new current $I_b(t)$, which has an opposite slope to $I_a(t)$, as shown in Figure 5.5.a. In this way, $I_b(t)$ compensates the current decrease of $I_a(t)$, and the total current $I_{on}(t)$ will remain a constant. The shaping of $I_b(t)$ is done by controlling another group of footers, denoted as Group B. As shown in Figure 5.5.a, when $I_b(t)$ converges with $I_a(t)$ at time $T_{alt}/2$, Group B is fully turned on, and $I_b(t)$ starts to decrease together with $I_a(t)$. Since after fully turned on, the current of Group B equals to the current of Group A, it can be deduced that the number of footers in Group B equals to the number of footers ($N_{ini}$) in Group A. Another requirement is that the gate voltage switching of Group B should take $T_{alt}/2$ to complete. This is the basic idea of current shaping.

Once Group A and B converge at $T_{alt}/2$, they merge as Group AB with current $I_{ab}(t)$, and start another round of current shaping. This time, the current decrease rate is doubled, since the on-state footers are doubled:

$$I_{ab}(t) = I_{avg} - \frac{2I_{avg} N_{ini}}{c_{vvg} \cdot r_f} \cdot t$$

Similarly, we need to construct a new current $I_c(t)$, which has an opposite slope to $I_{ab}(t)$, as shown in Figure 5.5.b. $I_c(t)$ converges with $I_{ab}(t)$ at time $3/4 T_{alt}$. And the number of footers in Group C equals to the number of footers ($2N_{ini}$) in Group AB. The requirement on the gate voltage switching
of Group C is that it should start at time $T_{all}/2$, and end at time $3/4T_{all}$. Similarly, by performing current shaping iteratively, a near-constant rush current can be obtained until the virtual ground voltage decreases to the safe threshold $\beta V_{DD}$. ($\beta$ is 5% in our experiments.) Figure 5.5.d illustrates the current of each footer group for a complete current shaping. Ideally, the total number of iterations required before reach $\beta V_{DD}$ is 5, because:

$$
\log_{0.5}(\beta) = 4.32 \ (\text{when } \beta = 0.05)
$$

(5.18)

![Current Shaping Diagram](image)

**Figure 5.5: Current Shaping**

The physical implementation of current shaping is realized by tuning the size of drivers and repeaters. Take the current shaping of Group C for example. The activation of Group C starts at time $T_{all}/2$. Hence the control signal should arrive at the very first footer ($2N_{ini} + 1$) of Group C at $T_{all}/2$. This can be realized by sizing the repeater at stage $2N_{ini} + 1$, as shown in 5.5.e. Group C is fully activated at time $3/4T_{all}$. And the turned-on time for each footer in Group C is $1/4T_{all}$. This can be realized by sizing the leaf node driver of each footer in Group C. Precisely, the sizing algorithms are shown in Equations 5.19 to 5.25.

$T_B$ in Equation 5.19 is the minimal total delay of the control signal for passing the Group B, calculated by the optimal repeater insertion rule. Equation 5.20 guarantees that the control signal arrives footer $2N_{ini} + 1$ at time $T_{all}/2$. Solving Equations 5.19, 5.20 and 5.21 yields the sizing for the repeater at stage $2N_{ini} + 1$. $T_C$ in Equation 5.22 is the minimal total delay of the control
signal for passing the Group C. $t_{d-1}^k$ represents the total driver delay except the leaf node driver for stage $k$. The leaf node driver is specifically sized to enable current shaping. Equation 5.24 defines the required delay of the leaf node driver. The first term of Equation 5.24 represents the repeater delay before stage $k$. Equation 5.25 calculates the delay of the leaf node driver given a certain size.

Solving Equations 5.24 and 5.25 together yields the sizing for the leaf node driver at stage $k$.

$$T_B = (2 + \sqrt{2})N_{ini}\sqrt{\frac{r_{rep}c_{rep}r_{wire}c_{wire}}{w_{rep}}}(5.19)$$

$$T_{ini} + T_B + t_r^{2N_{ini}+1} = \frac{T_{all}}{2}$$  \hspace{1cm} (5.20)

$$t_r^{2N_{ini}+1} = 0.69\left(\frac{r_{rep}}{w_{rep}} + r_{wire}\right)\left(\frac{1}{2}c_{wire} + c_{rep}\frac{2N_{ini}+2}{w_{rep}}\right)$$  \hspace{1cm} (5.21)

$$T_C = (2 + \sqrt{2})(2N_{ini} - 1)\sqrt{\frac{r_{rep}c_{rep}r_{wire}c_{wire}vspace}{w_{rep}}}-1mm$$  \hspace{1cm} (5.22)

$$t_{d-1}^k = (P_{opt} - 1)t_0(1 + P_{opt}\sqrt{F_k})$$  \hspace{1cm} (5.23)

$$F_k = \frac{W_f}{3w_d}; \ P_{opt} = ln(F);$$

$$\frac{k}{2N_{ini}}T_C + t_{d-1}^k + t_{leaf}^k = \frac{1}{4}T_{all}$$  \hspace{1cm} (5.24)

$$t_{leaf}^k = 2.2r_{inv}/w_{leaf}^k$$  \hspace{1cm} (5.25)

### 5.2.3 Multi-thread Activation

Current shaping addresses the fast discharge problem. In addition to complete discharge, all the footers have to be turned on to ensure normal operation of the core. This turns out to be challenging. With ideal discharging current $I_{avg}$, the virtual ground voltage is:

$$V_{vg}(t) = \frac{c_{vg} \cdot V_{ini} - I_{avg} \cdot t}{c_{vg}}$$  \hspace{1cm} (5.26)

To keep the rush current under $I_{avg}$, the maximum number of footer allowed to be turned on at any time $t$ is:

$$N_{MAX} = \frac{V_{vg}(t)}{I_{avg}}r_f = \frac{r_fI_{avg}c_{vg}}{V_{ini}c_{vg} - I_{avg}t}$$  \hspace{1cm} (5.27)

The curve line in Figure 5.6 demonstrates the trend of $N_{MAX}$. As can be observed, $N_{MAX}$ remains a small number for 80% of the time. $N_{MAX}$ only increases drastically when the discharge proceeds to the end. This requires large number of footer to be turned on in a short time. However, it is challenging to implement this due to the slow propagation speed of the control signal. The linear
line in Figure 5.6 shows the number of footers that the control signal passes at time $t$. Before Point 1 in Figure 5.6, the propagation speed is slower than required, so the delay should be minimized at the initial boost. Between Point 1 and 2, the propagation speed is faster than required, so the repeaters and drivers should be downsized during current shaping. After Point 2, the propagation speed is much slower than required.

Figure 5.6: Required Activation Speed VS. Control Signal Propagation Speed for $0.4 \times 0.4\mu m$ Core

We propose the technique of multi-thread activation to accelerate the propagation of the control signal. Figure 5.7 illustrates the idea. The control signal propagates through three threads, instead of one, from Point A. T1 and T2 travel through the edge of the core. They are responsible for the initial boost and current shaping. However, since the footer activation speed are essentially doubled, the repeaters and drivers sizing in the initial boost (Equations 5.7 to 5.12) and current shaping (Equations 5.19 to 5.25) need to be recalculated. TTmp1 travels across the core to reach Point B, and then spreads into two new threads, T3 and T4. In addition, TTmp1 spawns two child threads, TTmp2 and TTmp3 before reaches Point B. TTmp2 and TTmp3 stretch to Point C and D. Once they reach, they spread into 4 threads, T5 to T8. At this point, the propagation speed is improved to eight times, since there are 8 threads concurrently propagating.

Figure 5.7: Multi-thread Activation

In each iteration in Figure 5.5, current shaping requires different number of footers to be activated
within different time. The required activation speed (number of footers per unit time) in iteration 
\((j)\) can be calculated by:

\[
s_j = \frac{N_{ini} \cdot 2^{j-1}}{T_{all}/2^j} = \frac{N_{ini} \cdot 2^{2j-1}}{T_{all}} \tag{5.28}
\]

The propagation speed of two threads (T1 and T2) is:

\[
s_p = \frac{2}{2 + \sqrt{2/T_{rep}}} \tag{5.29}
\]

where the denominator represents the minimal repeater delay of one stage. At any iteration \(j\), if 
\(s_j\) is smaller than \(s_p\), only T1 and T2 are needed to activate footers. And they should be properly 
slowed down to avoid over-limit rush current. Contrarily, if \(s_j\) is larger than \(s_p\), more threads should 
be running for fast activation. To give an example, assume that at iteration \(j\), \(s_j\) equals to \(3.5s_p\). 
And \(s_j\) was smaller than \(s_p\) in \(j - 1\) iteration. It is desirable to have \(7 (3.5 \times 2)\) threads running at \(j\) 
iteration. However, our multi-thread activation only have the power of 2 as the number of threads. 
So the actual number \((y_j)\) of threads should be:

\[
y_j = 2^{\text{integ}(\log_2(s_j))} + 1 \tag{5.30}
\]

In this case, 8 threads are used to facilitate the needs of \(s_j\).

### 5.2.4 Design Automation Flow

A CAD flow is introduced in this section to solve the fast and reliable wake-up problem for 
ring style power gating, as shown in Algorithm 1. This algorithm first generates the core model as 
presented in Section 2.1. Then it calculates the maximum threads \((Y_{max})\) required for all 5 iterations 
of current shaping using the methods explained in Section 3.3. This maximum number is used to 
generate the control signal topology. However, the precise landing points of each thread cannot be 
determined yet. The algorithm then enters the actual 5 iterations of current shaping. One highlight 
is that in each iteration, the algorithm analyzes the core model to calculate the accurate virtual 
ground voltage distribution on the footer ring, in order to perform accurate sizing for footer drivers. 
The analysis can be done by any numerical algorithm or tool, such as SPICE. In each iteration, the 
algorithm treats the existing threads and newly created threads differently as described in Section 
3.3. But it guarantees two conditions for all threads: 1) guarantee the thread arrival time by repeater 
sizing 2) guarantee the current slope by sizing the leaf node drivers.
Algorithm 3: Fast and Reliable Activation Algorithm

INPUT: Circuit netlist; Technology; Footer total width, \( I_{avg}, V_{ini} \)

OUTPUT: Control signal topology; Repeater sizes and positions; Driver sizes;

--- Generate Core Model ---
Divide the core into 76um \( \times \) 76um blocks;
Obtain the \( \pi \) model of the circuit in each block: \( c_b, c_{vg}, r_b \);
Calculate virtual ground parasitics of each block: \( r_{gnd} \) and \( c_{gnd} \);
Partition the footer ring into footers of 76um width;
Calculate \( r_f \);

--- Control Signal Topology Generation ---
Optimize repeater and driver sizing for initial boost, calculate \( N_{ini} \);
Calculate \( T_{all} \);

\[
\text{foreach } j = 1 \text{ to } 5 \text{ do}
\]
\[
\text{Calculate required footer activation speed } s_j;
\]
\[
\text{Calculate the start time and end time for current shaping } T^s_j, T^e_j;
\]
\[
\text{if } s_j > s_p \text{ then}
\]
\[
\quad y_j = \text{integ}(\log_{sp}(s_j)) + 1;
\]
\[
\text{end if}
\]
\[
\text{end foreach}
\]
\[
Y_{max} = \text{MAX}(y_j);
\]
Design control signal topology with \( Y_{max} \) threads;

--- Current Shaping ---
\[
Y_{current} = 2;
\]
\[
\text{foreach } j = 1 \text{ to } 5 \text{ do}
\]
\[
\text{Calculate current virtual ground voltage distribution using core model;}
\]
\[
\text{foreach } i = 1 \text{ to } Y_{current} \text{ do}
\]
\[
\quad \text{Sizing the subsequent repeater to start thread until } T^s_j;
\]
\[
\text{end foreach}
\]
\[
\text{if } y_j > Y_{current} \text{ then}
\]
\[
\quad \text{foreach } i = Y_{current} \text{ to } y_j \text{ do}
\]
\[
\quad \quad \text{Calculate the landing point base on the existing frontiers;}
\]
\[
\quad \quad \text{Adjust the arrival time to } T^s_j \text{ by repeater sizing;}
\]
\[
\quad \quad \text{Tune the propagation speed to } s_j/y_j;
\]
\[
\quad \text{end foreach}
\]
\[
\text{end if}
\]
\[
\text{foreach } i = 1 \text{ to } y_j \text{ do}
\]
\[
\quad \text{Sizing the leaf node driver } w_{leaf} \text{ for current shaping;}
\]
\[
\quad \text{Calculate the frontier of each thread;}
\]
\[
\text{end foreach}
\]
\[
Y_{current}=y_j;
\]
\[
\text{end foreach}
\]
5.3 Experimental Results

The experiments were conducted using HSPICE with 32nm predictive model [41]. We generate 8 cores with different sizes from 1.8mm×1.8mm down to 0.14mm×0.14mm. The core circuit is replaced with numbers of C7552 circuits from ISCAS85 benchmark, since the function of the circuit does not affect the experiments. For example, the 0.14mm×0.14mm core contains 16 C7552 circuits, while the 1.8mm×1.8mm core contains 2304 C7552 circuits. The control signal parasitics are derived from ITRS 2007, with resistance as 5.6ohm/um and capacitance as 0.002pF/um. The virtual ground parasitics vary with the scale of the core and the power grid sizing. We sized the power grid such that the IR drop occupies 5% $V_{DD}$. Footers are also sized to incur a voltage droop of 5%$V_{DD}$.

We compared three approaches for footer activation. One is the naive single thread approach with uniform repeater and driver sizing. Another is the technique presented in [64]. The third one is our current shaping and multi-thread activation. All three approaches have the same rush current limit of $I_{avg}$. We mainly compare the wake-up time, area and power consumptions of three approaches. The area and power consumptions include footer, repeater and driver consumptions. The results are shown in Table 5.1. As can be observed, our approach reduces the wake-up time for 5 to 11 times compared with the naive approach, and 1.5 to 3 times compared with the technique in [64]. The area and power consumption of our technique is also slightly smaller than two other techniques. This is because the leaf node drivers for each footer consume a large portion of area and power, while our current shaping technique uses very small transistors for leaf node drivers. (Leaf node drivers can consume up to 25% of the total area, and 60% of the total power consumption). The area overhead in Table 5.1 shows the percentage of the overall area versus pure footer area.

Figure 5.8 demonstrates the repeater and footer sizing for the 0.86mm×0.86mm core with our approach. The repeaters that are passed by the control signal in different iteration are marked with different colors. 16 threads are running at the last iteration. Figure 5.8 also demonstrates the leaf node driver sizing for implementing current shaping. This design achieves a wake-up delay of only 0.341ns.

The repeaters network in Figure 5.8 consumes 118% extra area of the original repeater ring. However, drivers consume most part of the total area and power. Figure 5.9 shows the area and power breakdown for the design in Figure 5.8. It can be observed that the cost of repeaters are almost
Table 5.1: Area/Power/Wake-up Time of Three Approaches

<table>
<thead>
<tr>
<th>core (mm²)</th>
<th>Naive</th>
<th>Tech. in [64]</th>
<th>Our Tech.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T(ns)</td>
<td>Area (%)</td>
<td>Power</td>
</tr>
<tr>
<td>0.14 x 0.14</td>
<td>2.841</td>
<td>1.68%</td>
<td>4.37</td>
</tr>
<tr>
<td>0.22 x 0.22</td>
<td>2.856</td>
<td>1.68%</td>
<td>9.75</td>
</tr>
<tr>
<td>0.29 x 0.29</td>
<td>2.848</td>
<td>1.67%</td>
<td>17.2</td>
</tr>
<tr>
<td>0.43 x 0.43</td>
<td>2.880</td>
<td>1.67%</td>
<td>38.6</td>
</tr>
<tr>
<td>0.58 x 0.58</td>
<td>2.880</td>
<td>1.67%</td>
<td>68.6</td>
</tr>
<tr>
<td>0.86 x 0.86</td>
<td>2.880</td>
<td>1.67%</td>
<td>154</td>
</tr>
<tr>
<td>1.30 x 1.30</td>
<td>2.880</td>
<td>1.66%</td>
<td>346</td>
</tr>
<tr>
<td>1.80 x 1.80</td>
<td>2.880</td>
<td>1.66%</td>
<td>615</td>
</tr>
</tbody>
</table>

Figure 5.8: Design of a 0.86mmX0.86mm Core

negligible. And the reduction in leaf node driver cost by using our approach also compensates the extra repeater cost.
5.4 Conclusion and Future Work

This chapter analyzes the power mode transition problem of power gating from the chip level, with consideration on physical designs. A design rule is proposed for fast wake-up design with guaranteed power integrity. Current shaping and multi-thread activation techniques have been proposed and analyzed to significantly accelerate the wake-up process. A CAD flow is presented to implement automatic design of wake-up circuit. The technique of current shaping and multi-thread activation are applicable to other on-chip power switch designs as well. Future works include studying the effectiveness of the two techniques under temperature and process variation.
Chapter 6

Microarchitectural Level Adaptive Leakage Control

Modern CPUs have a large number of advanced FUs to accelerate performance, for example multiple ALUs, FPUs (floating point unit), LSU (load-store unit), BTB (branch target buffer). When different types of instructions are executed, some FUs are used, while some are not. This mutual exclusion creates idleness at MA level and thus opportunities to reduce FUs’ leakage. Figure 6.1 demonstrates the ALU activities when running benchmark program of 64 points FFT [68] on a Alpha CPU [69]. The dynamic energy consumption (red) of one calculation is averaged and normalized upon the leakage energy consumption (blue) of one clock cycle. It can be observed that even when the CPU is in active mode, significant idle leakage exists, demanding better solutions for more aggressive leakage control.

Figure 6.1: Dynamic Power VS. Idle Leakage on ALU 32nm 110°C
Research has been conducted to study leakage control at MA level for an active CPU [39, 70, 71, 72, 35]. Hu et al. [39] discussed using PG to turn off function units (FUs) when they are idle or branch mis-prediction occurs. [72] pointed out that idleness prediction is highly dependent on program regularity and large mis-prediction can lead to large energy penalty instead of saving. He et al. [71] emphasized the importance of designing leakage control system considering the inter-dependency of temperature and leakage. Kannan et al. [35] proposed the scheme of using leakage sensor to determine the runtime leakage of FUs, and turn off the FU with the highest leakage. Roy et al. [70] studied power gating control by inserting control instructions during compilation.

These studies have four common problems. First, their approaches did not perform quantitative study. They were not able to determine the optimum design point, such as when to apply leakage control and what the granularity of control should be. Second, as will be shown in Section 2, PG and BB are actually inefficient for MA level leakage control due to their large overhead. Third, most previous studies did not consider the temperature and process variation (TV/PV) impact on leakage control. Kannan et al. [35] was the first one to introduce the consideration of TV/PV. However, their scheme simply chooses the FU with the highest leakage and applies PG on it. This scheme cannot guarantee the net energy saving by applying PG to be positive. Finally, the previous studies were based on two approaches to generate the leakage control signals: idleness prediction at runtime or control instruction insertion during compilation. The problem of prediction is its dependency on regularity in a program. With low regularity, it might cause large mis-prediction [72]. Complicated mechanisms are needed to ensure the correctness of prediction [70]. The problem of compiler-directed detection is the large amount of control instructions inserted to the original codes.

This chapter introduce an adaptive Light-Weight Vth Hopping (LW-VH) scheme to stretch the limit of MA level leakage control. Our study has the following contributions. First, we derive a systematic and quantitative approach for leakage control at MA level leakage using LW-VH. Second, we propose the usage of leakage sensor to calculate the runtime MIT, and a technique to dynamically adjust leakage control policy at runtime for TV/PV compensation. Finally, we introduce the scheme of instruction-queue monitoring to generate the leakage control signals.

This chapter is organized as follows. Section 6.1 quantitatively compares the overhead of PG, BB and LW-VH with emphasis on energy overhead, and in the presence of TV/PV. Section 6.2 analyzes the potential of MA level leakage control, and proposes a systematic approach to maximize the leakage saving. Section 6.3 introduces the idea of adaptive LW-VH to address the TV/PV problem,
as well as the technique of instruction queue monitoring. Section 6.4 discusses the experimental results.

6.1 Leakage Control Techniques with the Presence of TV/PV

To perform a quantitative study on CPU leakage power reduction, the EBT of each leakage control technique must be specified. In this section, we will study the EBT and WUT of each technique we have introduced so far in the context of CPU leakage control, and in the presence of TV/PV.

6.1.1 Power Gating, Body Biasing and Light-Weight $V_{th}$ Hopping

The basic PG diagram (ground gating) is shown in Figure 6.2.a. Since footers are responsible for providing active currents for the target circuit, they should be designed in large size to minimize performance penalty. Buffers are needed to drive these large-scale footers. Denote the sum of both capacitance as $C_F$ for convenience. For each mode transition of PG, large energy overhead ($O_{PG}$) is incurred for switching the footers and buffer capacitance:

$$O_{PG} = C_F V_{DD}^2$$  \hspace{1cm} (6.1)

EBT of PG can then be calculated by its definition:

$$B_{PG} = \frac{O_{PG}}{S_{PG/time}}$$  \hspace{1cm} (6.2)

where $S_{PG/time}$ is the leakage energy saving per unit time by applying PG. For PG, although its $S_{PG/time}$ is high, its large $C_F$ leads to large overhead ($O_{PG}$). Consequently, long EBT time ($B_{PG}$) is required to compensate the overhead. The typical value of $C_F$ and $B_{PG}$ is shown in Table 6.1.

The basic BB design is shown in Figure 6.2.b. For each mode transition of BB, energy overhead ($O_{BB}$) is incurred for charging the body capacitance ($C_B$), and for switching the control transistors ($C_S$):

$$O_{BB} = (C_B + C_S) \Delta V_{bias}^2$$  \hspace{1cm} (6.3)

where $\Delta V_{bias}$ is the offset of body voltage when BB is applied. EBT of BB can then be calculated by:
\[ B_{BB} = \frac{O_{BB}}{S_{BB/time}} \]  

where \( S_{BB/time} \) is the leakage energy saving per unit time by applying BB. For BB, its large \( C_B \) also leads to large overhead \( (O_{BB}) \), and thus long EBT \( (B_{BB}) \) for BB. The typical value of \( C_B \) and \( B_{BB} \) is shown in Row 2 of Table 6.1.

\[ VDD \]

**Virtual Ground**

**a. Ground Power Gating**

**b. Body Biasing**

Figure 6.2: Power Gating and Body Biasing

Conventionally, the optimum body biasing voltage of \( V_{th} \) Hopping is determined when the sum of subthreshold and BTBT leakage is minimized. This voltage achieves high leakage reduction ratio, but incurs large energy overhead due to the quadratic dependency of \( O_{BB} \) on \( \Delta V_{bias} \) (Equation 6.3). Figure 6.3 demonstrates the energy overhead versus leakage saving of different biasing voltages on an inverter chain (our studies [73]).

We further conducted experiments on an ALU. As shown in the Row 2 and 3 of Table 6.1, instead of using large \( \Delta V_{bias} \) (0.7V), we can choose small \( \Delta V_{bias} \) (0.15V) to reduce EBT for 8×, while maintaining 51% leakage reduction \( (Z_{min}) \). 51% might not be high enough for standby leakage reduction, but is sufficient for short idleness exploitation. This is the basic idea of LW-VH.
6.1.2 Leakage Control with TV/PV Impact

Leakage of the target circuit varies significantly with temperature and process in scaled technologies. When designing leakage control policy, worst case scenario should be considered. Assume that leakage current of the target circuit varies from \( I_{\text{min}} \) to \( I_{\text{max}} \) (3\( \sigma \) values). According to its definition, EBT (\( B_{\text{norm}} \)) of the circuit varies:

\[
\frac{E_{\text{overhead}}}{Z \cdot I_{\text{max}} V_{DD}} = B_{\text{min}} \leq B_{\text{norm}} \leq B_{\text{max}} = \frac{E_{\text{overhead}}}{Z \cdot I_{\text{min}} V_{DD}} \tag{6.5}
\]

where \( Z \) is the leakage reduction ratio. A robust leakage control policy should guarantee positive energy saving in all dies with all possible temperatures. Thus, the worst case EBT (\( B_{\text{max}} \) in Equation 6.5) must be chosen to represent EBT of the target circuit. We call this phenomenon as “corner case leakage control”. We conducted Monte Carlo simulation on an ALU with 32nm technology to demonstrate this phenomenon. The 3\( \sigma \) variations of gate length and \( T_{ox} \) are 12\% and 4\%, respectively. The temperature variation is set to be from 70\(^\circ\)C to 110\(^\circ\)C, with normal temperature as 90\(^\circ\)C. Figure 6.4 shows the leakage variation induced by TV and PV. The corner case leakage can be as low as 1/6 of the nominal value. According to Equation 6.5, runtime EBT of the target circuit can be magnified 6 times, as shown in the last column of Table 6.1. This dramatically weakens the
effectiveness of any leakage control technique.

Figure 6.4: TV/PV Induced Leakage Variation

6.2 Microarchitectural Level Leakage Control

The potential of MA level leakage control is dependant on three factors: 1) How much idleness there exists at MA level. 2) How much idleness can be detected. 3) How much idleness can be exploited by leakage control techniques to yield leakage saving. We will answer the first question in Section 3.1, the second question in Section 4, and the third question in Section 3.2.

6.2.1 Mutual Exclusion, Locality and Exceptions

The idleness at MA level is caused by three mechanisms: mutual Exclusion, exceptions and locality. Mutual exclusion refers to the case that when CPU is executing instructions in sequence, some FUs are activated by certain types of instructions, while others are not. Here we classify all instructions into four basic types: control, memory manipulation (load/store), integer arithmetic, floating point arithmetic [69]. Table 6.2 summarizes the busy/idle FUs when different instructions are executed. The leakage of each FU is also shown in the last column [15]. We can see that ALU, FPU, and branch target buffer (BTB) are the three major sources of CPU leakage.

The second type of idleness is created by the locality in programs. A typical example is the memory access locality. At runtime, it is common that a subset of cache system will be accessed frequently during a short period. So the rest of cache system exhibits very long idleness (> 100 cycles), and creates perfect applications for leakage reduction. Cache leakage reduction has been a well-studied research topic, and will be ignored in our study.
### Table 6.2: Leakage Power and Idleness of FUs (32nm)

<table>
<thead>
<tr>
<th>FU</th>
<th>Idleness</th>
<th>Leakage [15]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Control</td>
<td>Load/Store</td>
</tr>
<tr>
<td>ALU *3</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>FPU</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>BTB</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Integer Reg.</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>FP. Reg.</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>RUU</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>LSQ</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Decode</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

The third type of idleness at MA level is caused by the exceptions during CPU runtime. A typical example is branch mis-prediction. When branch mis-prediction occurs, the current pipeline execution will be flushed, and the correct instruction will be fetched from the instruction memory, decoded and executed. This event usually creates 10 to 20 cycles of idleness for ALU, FPU and BTB, and can be exploited for leakage reduction.

#### 6.2.2 Leakage Saving Potentials

Assume that all idleness are detectable. A suitable leakage control technique is needed to efficiently exploit these idleness, and maximize leakage saving. Denote the EBT, WUT, leakage reduction ratio and energy overhead of a certain leakage control technique as $B$, $W$, $Z$ and $O$ respectively. Also denote the length of idle period as $T_i$, and the occurrence of $T_i$ as $U_i$. The net energy saving ($E_i$) for idleness of length $T_i$ is:

$$E_i = (T_i - W)V_{DD}I_{leak}Z - O$$  \hspace{1cm} (6.6)

Equation 4.4 assumes that there is no energy saving during wake-up period. The first term represents the leakage saving. By subtracting the second term (energy overhead), we have the net energy saving. By mutating Equation 6.2 or 6.4, the second term can be further expressed as:

$$O = V_{DD}I_{leak}ZB$$  \hspace{1cm} (6.7)

Thus, Equation 4.4 can transform into:

$$E_i = (T_i - B - W)V_{DD}I_{leak}Z = (T_i - M)V_{DD}I_{leak}Z$$  \hspace{1cm} (6.8)
where $M$ denotes MIT ($B + W$). Finally, we can determine the overall efficacy of leakage control by:

$$P = \frac{\sum U_i E_i}{V_{DD} I_{leak} T_{runtime}} = \frac{\sum U_i ((T_i - M))Z}{T_{runtime}}$$

(6.9)

where the numerator represents that total net energy saving for all idle events, the denominator represents the total leakage energy consumption during program runtime. $P$ is the percentage of energy saving by leakage control. From Equation 6.9, it can be observed that the MIT (M) and leakage reduction ratio (Z) determine the efficacy of leakage control. For PG and BB, their M and Z values are fixed. For LW-VH, its M and Z are tunable, depending on the biasing voltages.

We conducted experiments on the net leakage saving of an ALU with different leakage control techniques. The experimental settings will be introduced in Section 5. As shown in Figure 6.5, although the Z value of PG and BB is high, their high M value leads to large overhead and thus low leakage saving. LW-VH can squeeze its M to a very small value, and achieve better leakage saving. (Maximum is 19% when $M = 5$.)

![Figure 6.5: Net Leakage Saving VS. Variation of M and Z](image)

Now assume that we implement a mechanism for TV/PV compensation, such that LW-VH can be applied according to the actual runtime MIT of the target circuit, instead of the “corner case” MIT. As such, leakage saving can be further improved. We call this technique as “adaptive LW-VH”. The effectiveness of adaptive LW-VH on ALU is shown in Figure 6.5, where it achieves 30% maximum leakage saving when $M = 6$. The implementation of it will be introduced next.
6.3 The Whole Scheme

In many CPU architectures, an instruction queue (IQ) exists between the fetch unit and decode unit. This IQ provides us the ability to monitor future workloads, and apply leakage control accordingly. A counter can be used to monitor instruction types in IQ, and decides which FUs to be turned off. For example, if there is no FP instructions in the next N cycles, and MIT of FPU is larger than N, FPU can be turned off. Similarly, ALU and BTB can be turned off based on monitoring integer arithmetic and control instructions, respectively. The whole scheme is illustrated in Figure 6.6.a, where each major FU (ALU, FPU and BTB) has a dedicated counter to monitor the consecutive idleness of its corresponding instructions. When the counter value exceeds MIT of the FU that it is monitoring, LW-VH will be triggered and applied to that particular FU.

Figure 6.6: Microarchitectural Level Adaptive LW-VH
To address the “corner case leakage control” problem, leakage sensor [74] is used. The diagram of this sensor is shown in Figure 6.6.b. The output of the single-channel sensor is a one-bit digital signal, indicating whether the actual leakage is larger than the reference. Multiple-channel sensor can be used to improve the resolution, as shown in Figure 6.6.c. The sensed leakage value will be used to determine the runtime MIT (RMIT) of the FU, and change the control policy dynamically. Take ALU for example. The HSPICE simulation on 1024 die samples (Table 6.1) suggested that the RMIT of ALU spans from 1 to 18 cycles. So the counter for ALU should have at least 5 bits \(2^5 > 18\). Next, we use a sensor with 5 channels \((C_0 \text{ to } C_4\) in Figure 6.6.d) to represent 5 different leakage levels, which yield 1, 2, 4, 8 and 16 cycle RMIT respectively. Whenever the actual leakage value is larger than the reference value, the corresponding channel will give output ‘1’. The output of the sensor, together with the counter value, are used to generate the control signal for adaptive LW-VH, as shown in Figure 6.6.d. For example, when runtime leakage of ALU yields 6 cycle RMIT, the output of sensor channel \(C_3\) (corresponding to 8-cycle RMIT) and \(C_4\) (corresponding to 16-cycle RMIT) will be ‘1’, so the \(B_3\) and \(B_4\) of the counter will be passed to the OR5 gate. So if the counter detects more than 8 (binary ‘01000’) cycles of idleness, the output of OR gates will be logic ‘1’, and LW-VH will be triggered.

To exploit the idleness created by exceptions, we program the idle length created by the exception into counters. Assume that branch mis-prediction creates 10 cycles of idleness for each FU. Then we can program the preset value of all counters as 10 (binary ‘01010’). Whenever branch mis-prediction occurs, each counter will be preset to 10, as shown in Figure 6.6.d. If the sensed RMIT of any FU is larger than 10 cycles, LW-VH will be applied to that FU.

### 6.3.1 Leakage Control Resolution and Granularity

In Section 4.1, the resolution of the sensor is determined arbitrarily, leading to sub-optimal leakage control. Increasing counter and sensor resolution can address this problem, but incurs more area for leakage sensor. The granularity of leakage control is another open question. In the previous study, we arbitrarily decided that the granularity is at the FU level. An FU is considered as a whole for sleep control. Leakage control with finer granularity is possible, if we consider the sub-circuits inside each FU. As shown in Figure 6.7, an ALU is separated into three sub-circuits: adder, multiplier and logic units. Each sub-circuit has a dedicated counter, a leakage sensor and LW-VH control switches. Each counter needs to counter the corresponding instruction types (add instruction
for adder; multiply instruction for multiplier; other ALU instructions for logic operations), and
decides the application of adaptive LW-VH on its sub-circuit. Even finer granularity is also possible.
For example, we can divide logic operations into or/shift/and, and separately control the circuit for
each operation. Another example is that the adder can be divided into LSB and MSB. By monitoring
the operands, we can decide whether to turn off the MSB or not. However, finer granularity also
demands for more control and monitoring circuits, and thus consumes more area and power.

![Diagram]

**Figure 6.7: Leakage Control With Finer Granularity**

### 6.4 Experimental Results

We conducted experiments to verify the effectiveness of LW-VH with adaptive control for MA level leakage control. The CPU layout is illustrated in Figure 6.8.a. 1024 CPU die samples with TV/PV were generated with 32nm predictive technology. On each die, there are three high-leakage FUs: ALU, FPU and BTB. For each FU, LW-VH was designed and optimized in nominal conditions. HSPICE simulations were conducted to determine the RMIT span of each FU for all dies. Then the RMIT span is used to design the adaptive control scheme. Finally, we modified SimpleScalar to estimate the net leakage saving of the whole CPU. The experiment flow is shown in Figure 6.8.b.

Figure 6.9 and Table 3 show experimental results on the overall leakage reduction for different benchmark programs. We compare the results of four techniques: PG, BB, Adaptive LW-VH and Adaptive LW-VH with PG. In Figure 6.9, PG and BB have minor savings on most benchmark programs for each FU (ALU, FPU, BTB) and for branch mis-prediction (BRA), while adaptive LW-VH achieve 36.3% total CPU leakage reduction. That is 23.3% improvement over PG, and 25.3% over BB.
Figure 6.8: Experiment Settings

Figure 6.9: Average Net Leakage Saving Percentage of Four Techniques on 1024 CPU Dies with 32nm technology

Table 3. Total CPU Net Leakage Saving

<table>
<thead>
<tr>
<th></th>
<th>PG</th>
<th>BB</th>
<th>LW-VH1</th>
<th>LW-VH2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>18.4%</td>
<td>15.6%</td>
<td>36.4%</td>
<td>44.3%</td>
</tr>
<tr>
<td>iFFT</td>
<td>17.3%</td>
<td>14.4%</td>
<td>35.6%</td>
<td>43.1%</td>
</tr>
<tr>
<td>susan</td>
<td>4.2%</td>
<td>2.6%</td>
<td>31.9%</td>
<td>33.4%</td>
</tr>
<tr>
<td>basic math</td>
<td>15.3%</td>
<td>12.6%</td>
<td>38.6%</td>
<td>44.5%</td>
</tr>
<tr>
<td>qsort</td>
<td>14.7%</td>
<td>14.0%</td>
<td>32.2%</td>
<td>38.1%</td>
</tr>
<tr>
<td>adpcm</td>
<td>9.8%</td>
<td>9.6%</td>
<td>43.0%</td>
<td>47.0%</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>13%</td>
<td>11%</td>
<td>36.3%</td>
<td>41.7%</td>
</tr>
</tbody>
</table>
Figure 6.10 demonstrates the improvement of ALU leakage saving by using finer resolution and granularity. The improvement is significant when resolution is low, and is minor when resolution is larger than 5. Leakage control of finer granularity yielded 26% average improvement, since most benchmark programs have little to no multiplication operations (except ‘susan’). Hence putting Multiplier into sleep separately can yield substantial leakage saving.

![Graph showing leakage saving vs. resolution/ granularity](image)

Figure 6.10: ALU Leakage Saving VS. Resolution/Granularity

### 6.5 Conclusion

This chapter targets active leakage control at MA level. The obstacle is the energy overhead problem of PG and BB for mode transition. When TV/PV are considered, PG and BB can hardly achieve leakage saving for an active CPU. Adaptive Light-Weighted Vth Hopping (LW-VH) technique has been introduced for overhead reduction and TV/PV compensation. We systematically analyze the leakage control potentials at MA level, and propose a comprehensive scheme to maximize leakage control at MA level.
Chapter 7

Summary and Future Work

The continuous technology scaling of CMOS Integrated Circuit technology has evoked numerous challenges in physical designs due to the change of device property in scaled technology nodes. This phenomena is significantly aggravated when we enter the deep sub-micron era, and manipulate molecules on the scale of a few hundreds. A successful IC design nowadays needs to address many physical issues: stringent power density and heat removal requirement, the emerging electro-migration, NBTI and aging problem, the process variation problem, the voltage drop and power/signal integrity problem.

Leakage power control is among the most compelling physical problems, due to its rapid rising percentage in the total power consumption, while power is identified as the number one obstacle for further integration. Runtime leakage control is especially promising, since there exists huge amount of idleness for most runtime circuits. However, aggressive leakage saving is hardly achievable through current coarse-grained and ad-hoc design style. A set of modeling, optimization, design methodology and design automation methods is in urgent need of being invented.

In summary, our whole study has established a sound mathematic basic by performing leakage saving and overhead modeling from the transistor level to the circuit level. Based on the modeling, we explored trade-offs at physical, RTL and microarchitecture levels, and proposed various novel techniques. Two popular leakage control techniques, namely power gating and body biasing, have been investigated, with the emphasis on their energy and delay overhead during mode transition. Our research has resulted in eleven publications in premier conferences and journal publications. Our major contributions are as following.
7.1 Our Contributions

1) Accurate modeling

An aggressive RTLC system performs mode transition frequently. Therefore, accurate modeling of energy and delay overhead incurred by mode transition is essential to ensure its efficacy. Conventional modeling methods incur large errors. The demanded models should be fast, accurate and temperature-aware. These goals have been achieved in our studies on power gating (PG) [75, 54, 76] and body biasing (BB) [73, 77]. These accurate models can be used to determine the optimum control policy.

2) Physical Design

To reduce leakage control overhead, we proposed two novel physical design techniques, namely workload-adaptive $V_{th}$ hopping and hierarchal $V_{th}$ hopping, to drastically reduce both delay and energy overhead of the basic $V_{th}$ hopping technique [78, 79]. To address the power noise and delay overhead problem, we proposed two novel wake-up techniques, namely current shaping and multi-thread activation, to minimize the wake-up delay with guaranteed power integrity [80].

3) Circuit Optimization

Granularity of leakage control has been an open question for researchers in this field. To realize aggressive RTLC, fine-grained approach has been an active research area. However, we have observed that control cost by adopting finer granularity increases faster than the improvement of leakage saving. Therefore, our research proposes the concept of optimum granularity to make a trade-off between leakage saving and control cost [81]. We have also proposed two novel circuit clustering techniques, namely clustering for temporal and spatial idleness exploitation, to systematically improve leakage saving with finer granularity [81]. Given any target circuit, A heuristic algorithm has been developed to analyze the idleness exploitation potentials, estimate leakage saving and control costs by circuit clustering, and finally decide the optimum granularity and optimum partition of the circuit.

4) Architectural Design

The effectiveness of a RTLC system is determined by two factors: the system ability of idleness exploitation and leakage reduction ratio of the technique used. Current leakage control systems based on PG or BB excel in the latter, but perform poorly in the former factor due to their large
overhead for mode transition. It is observed that by using light-weight biasing voltage sources for $V_{th}$ hopping, the overhead can be reduced by $4-6 \times$ in our studies [73, 82, 77]. The trade-off is that leakage reduction ratio drops to 60%. The overhead can be further reduced by “selective leakage control” to a very small value, such that joint dynamic and leakage power reduction at RTL level is possible [82]. Based on this observation, we introduce a 3-tier aggressive leakage control system (figure below). PG or BB will be used to exploit system level idleness; Light-weight $V_{th}$ hopping (LW-VH) will be used to exploit microarchitectural-level medium idleness [77, 83]; Selective leakage control will be used to exploit RTL-level short idleness [82]. A leakage sensor is used to tackle the TV/PV problem. It dynamically tunes the microarchitectural-level control policy, and activates RTL-level selective leakage control in a high-leakage runtime environment.

5) Design Automation

A CAD framework has been proposed by us to automate modeling, circuit optimization and architecture generation process. The bottom figure shows the diagram of this framework. The inputs of this framework are technology file, cell library, chip design, activity profile and TV/PV. The output will be the optimized 3-tier leakage control system design with estimated leakage saving and area costs. Our preliminary experimental results [77] shows that 35% average leakage saving can be achieved at microarchitectural level, and up to 15% of leakage saving at RTL level with the presence of TV/PV.

7.2 Future Work

As we have explained previously, a successful design in the deep sub-micron era requires careful analysis of problems in various domains. There are certain questions remain open towards the end of forging a practical leakage control solution.
7.3 Power Gating and Body Biasing Modeling with the Presence of Process Variation

Our modeling on energy breakeven time in Chapter 1 and wake-up time in Chapter 5 ignores the runtime temperature and process variation. However, TV/PV has a significant impact on both parameters. An accurate trade-off has to be made with the consideration of both. In Chapter 6, we conducted an approximate estimation on EBT with the presence of TV/PV. What we did not consider is the variation of energy overhead with TV/PV. The EBT estimation will be different when that is taken into account. An accurate estimation will be meaningful for designers.

7.4 Low Power Versus Reliability

In order to achieve ultra low power design, aggressive power management (PM) is applied pervasively and aggressively at runtime for design nowadays. Most PM techniques, such as DVS, Body Biasing and Power Gating, change the terminal voltage of each transistor in the circuit to control power consumption, leading to significant charge/discharge when performing mode transition. The consequent rush current causes IR drop and inductive noise on the power grid, and results in soft error in adjacent active circuits. So PM techniques have to be tuned carefully to achieve a balance between power, robustness and performance. This problem can be especially severe for leakage power control techniques due to their heavy manipulation of transistor terminal voltages.

There can be two layers of solutions to tackle this problem. First, design time optimization can be realized to place the noise source away from vulnerable components. For example, a power gating aware floorplan or task scheduling can be studied to place the block with frequent power
gating mode transition away from those blocks, which are sensitive to voltage drop. The second layer of solutions falls into the category of runtime techniques. A runtime protection mechanism can be designed to quarantine active circuit from noise. For example, we can temporarily increase the supply voltage of the active block to counteract the voltage drop impact from the adjacent noise sources.

7.5 Future Idleness Detection

Most of the previous studies have been focused on manipulating the circuit when there is certain period of idleness in the circuit workload. On the other end, another key question is how to obtain the workload information. More importantly, we need to know the workload information of future. Knowing the future workload information (FWI) can be implemented at different levels:

a) System level. At system level, we usually have good information of future circuit status. For example, the GSM cell phone receives paging signal every 20ms. So the circuit dedicated to manipulate the paging signals can be designed to perform mode transition every 20ms. Another example is the real-time system, where in some cases the workload is pre-scheduled during design-time. In this case, we know the FWI very well and can fully utilize it to guide runtime leakage control.

b) Architecture level. At architecture level, the acquisition of FWI is more difficult. But there are a variety of techniques available. [84] introduces the method of monitoring the instructions of microprocessor to guide the mode transition of function units. [39] presents the method of using branch prediction to guide the mode transition. Our studies in Chapter 6 have preliminarily demonstrated the microarchitectural level FWI. However, more work needs to be done in this area.

c) Circuit Level. Circuit level provides the least FWI. We have demonstrated the scheme of workload monitor in Chapter Four. It is a completely run-time monitoring scheme. For a FSM-controlled circuit, there is already hidden FWI information in the design. For example for a simple FSM in Figure 7.1, It takes at least six clock cycles for $S_3$ to transit to $S_6$. This means if the EBT is less than six cycles, the FSM circuit dedicated to state six and the corresponding datapath can be turn into low-leakage mode.

Although circuit level FWI detection is not as effective as higher level approaches, it is more
pervasive. It can be effective, since it can be applied to any part of a design.
Bibliography


[66] Q. K. Zhu, POWER DISTRIBUTION NETWORK DESIGN FOR VLSI. Wiley InterScience.


