Ph.D. DISSERTATION

Voltage and Retention Storage Allocation Problems for SRAMs and Power Gated Circuits

정적 램 및 파워 게이트 회로에 대한
전압 및 보존용 공간 할당 문제

BY

KIM TAEHWAN

AUGUST 2021

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Voltage and Retention Storage Allocation Problems for SRAMs and Power Gated Circuits

정적 램 및 파워 게이트 회로에 대한
전압 및 보존용 공간 할당 문제

지도교수 김 태 환
이 논문을 공학박사 학위논문으로 제출함

2021년 7월

서울대학교 대학원

전기·정보 공학부

김 태 환

김태환의 공학박사 학위 논문을 인준함

2021년 7월

위 원 장: _____
부위원장: _____
위 원: _____
위 원: _____
위 원: _____

Abstract

Low power operation of a chip is an important issue, and its importance is increasing as the process technology advances. This dissertation addresses the methodology of operating at low power for each of the SRAM and logic constituting the chip.

Firstly, we propose a methodology to infer the minimum operating voltage at which SRAM failure does not occur in all SRAM blocks in the chip operating on near threshold voltage (NTV) regime through the measurement of a monitoring circuit. Operating the chip on NTV regime is one of the most effective ways to increase energy efficiency, but in case of SRAM, it is difficult to lower the operating voltage because of SRAM failure. However, since the process variation on each chip is different, the minimum operating voltage is also different for each chip. If it is possible to infer the minimum operating voltage of SRAM blocks of each chip through monitoring, energy efficiency can be increased by applying different voltage. In this dissertation, we propose a new methodology of resolving this problem. Specifically, (1) we propose to infer minimum operation voltage of SRAM in design infra development phase, and assign the voltage using measurement of SRAM monitor in silicon production phase; (2) we define a SRAM monitor and features to be monitored that can monitor process variation on SRAM blocks including SRAM bitcell and peripheral circuits; (3) we propose a new methodology of inferring minimum operating voltage of SRAM blocks in a chip that does not cause read, write, and access failures under a target confidence level. Through experiments with benchmark circuits, it is confirmed that applying different voltage to SRAM blocks in each chip that inferred by our proposed methodology can save overall power consumption of SRAM bitcell array compared to applying same voltage to SRAM blocks in all chips, while meeting the same yield target.

Secondly, we propose a methodology to resolve the problem of the conventional retention storage allocation methods and thereby further reduce leakage power con-

sumption of power gated circuit. Conventional retention storage allocation methods have problem of not fully utilizing the advantage of multi-bit retention storage because of the unavoidable allocation of retention storage on flip-flops with mux-feedback loop. In this dissertation, we propose a new methodology of breaking the bottleneck of minimizing the state retention storage. Specifically, (1) we find a condition that mux-feedback loop can be disregarded during the retention storage allocation; (2) utilizing the condition, we minimize the retention storage of circuits that contain many flip-flops with mux-feedback loop; (3) we find a condition to remove some of the retention storage already allocated to each of flip-flops and propose to further reduce the retention storage. Through experiments with benchmark circuits, it is confirmed that our proposed methodology allocates less retention storage compared to the state-of-the-art methods, occupying less cell area and consuming less power.

keywords: SRAM, on-chip monitoring, process variation, power gating, state retention, leakage power

student number: 2016-20884

Contents

Abstract **i**

Contents **iii**

List of Tables **vi**

List of Figures **viii**

1 Introduction **1**

1.1 Low Voltage SRAM Monitoring Methodology 1

1.2 Retention Storage Allocation on Power Gated Circuit 5

1.3 Contributions of this Dissertation 8

2 SRAM On-Chip Monitoring Methodology for High Yield and Energy Efficient Memory Operation at Near Threshold Voltage **13**

2.1 SRAM Failures 13

2.1.1 Read Failure 13

2.1.2 Write Failure 15

2.1.3 Access Failure 16

2.1.4 Hold Failure 16

2.2 SRAM On-chip Monitoring Methodology: Bitcell Variation 18

2.2.1 Overall Flow 18

2.2.2 SRAM Monitor and Monitoring Target 18

2.2.3 V_{fail} to $\hat{V}_{ddmin}$ Inference 22
2.3 SRAM On-chip Monitoring Methodology: Peripheral Circuit IR Drop and Variation 29
2.3.1 Consideration of IR Drop 29
2.3.2 Consideration of Peripheral Circuit Variation 30
2.3.3 V_{ddmin} Prediction including Access Failure Prohibition 33
2.4 Experimental Results 41
2.4.1 $\hat{V}_{ddmin}$ Considering Read and Write Failures 42
2.4.2 $\hat{V}_{ddmin}$ Considering Read/Write and Access Failures 45
2.4.3 Observation for Practical Use 45

3 Allocation of Always-On State Retention Storage for Power Gated Circuits - Steady State Driven Approach 49
3.1 Motivations and Analysis 49
3.1.1 Impact of Self-loop on Power Gating 49
3.1.2 Circuit Behavior Before Sleeping 52
3.1.3 Wakeup Latency vs. Retention Storage 54
3.2 Steady State Driven Retention Storage Allocation 56
3.2.1 Extracting Steady State Self-loop FFs 57
3.2.2 Allocating State Retention Storage 59
3.2.3 Designing and Optimizing Steady State Monitoring Logic . . 59
3.2.4 Analysis of the Impact of Steady State Monitoring Time on the Standby Power 63
3.3 Retention Storage Refinement Utilizing Steadiness 65
3.3.1 Extracting Flip-flops for Retention Storage Refinement 66
3.3.2 Designing State Monitoring Logic and Control Signals 68
3.4 Experimental Results 73
3.4.1 Comparison of State Retention Storage 75
3.4.2 Comparison of Power Consumption 79

3.4.3 Impact on Circuit Performance 82
3.4.4 Support for Immediate Power Gating 83

4 Conclusions **89**
4.1 Chapter 2 89
4.2 Chapter 3 90

Abstract (In Korean) **97**

List of Tables

2.1 Process variation on each part of the circuit considered 17
2.2 Types of non-systematic process variation considered. 17
2.3 Size, count, and other design parameters for target SRAM 21
2.4 Dies and $\hat{V}_{ddmin}$ distributions by V_{fail} 44
2.5 Savings on leakage power, read energy, and write energy of SRAM bitcell array over those by the conventional flow [31, 32] for read/write operation. 44
2.6 Dies and $\hat{V}_{ddmin}$ distributions by V_{fail} and L_{WL} 46
2.7 Savings on leakage power, read energy, and write energy of SRAM bitcell array over those by the conventional flow [31, 32] for read/write/access operation. 46

3.1 The number of self-loop FFs in circuits from IWLS2005 benchmarks and OpenCores. 52
3.2 Changes of the number of steady self-loop flip-flops as γ changes. . . 55
3.3 Changes of $prob_f$ as γ changes. 58
3.4 Comparison of total number of flip-flops deploying state retention storage (#RFFs) and total bits of retention storage (#Rbits) used by [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs). . . . 72

3.5 Comparison of cell area occupied by flip-flops(FF), always-on control logic(Ctrl) and combinational logic including state monitoring logic and excluding always-on control logic(Comb) in [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs). Wakeup latency l is 2. 76
3.6 Same as Table 3.5, with wakeup latency $l = 3$. 77
3.7 Comparison of the active power (= dynamic + leakage in active mode) and standby power (= leakage in sleep mode) consumed by [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs). 80
3.8 f_{max} comparison of No-Opt [24] and Full-Opt2 82
3.9 Power state table of powers in Fig. 3.19 84
3.10 Total number of flip-flops deploying state retention storage (#RFFs) and total bits of retention storage (#Rbits) used by ours supporting immediate power gating 85
3.11 Active power and standby power in each of sleep modes consumed by ours supporting immediate power gating. 86

List of Figures

1.1 Probability of read, write, and overall operation failures on 14nm HC (High-Current) and HD (High-Density) bitcells [4]. V_{dd} is normalized to nominal voltage. 2

1.2 Dies with different global corners exhibit different rates of SRAM failure, though they have an identical local random variation. 3

1.3 The structure of circuit with power gating. 5

1.4 The structure of multi-bit retention flip-flop (MBRFF) that can save $l > 1$ retention bits [22]. 7

1.5 Standard flows for low power design, which support retention with power gating. 11

2.1 Waveform of SRAM bitcell failures: (a) read failure, (b) write failure, (c) access failure, (d) hold failure. V_{dd} of peripheral circuit and bitcell are 0.6V and 0.7V, respectively. 14

2.2 6T SRAM bitcell storing data "1" 15

2.3 Overall flow of our proposed SRAM on-chip monitoring methodology: (a) building-up V_{fail}-V_{ddmin} correlation table at design infra development phase, (b) deriving an SRAM $\hat{V}_{ddmin}$ on each die at silicon production phase. 19

2.4 The changes of die count distribution in each V_{fail} group (0.56V$\sim$0.64V) as the size of SRAM monitor increases. 20

2.5 The changes of the number of bitcells with failure in the monitored test SRAM as the applied voltage V_{dd} ($V_{dd1} > V_{dd2} > \cdots > V_{dd8}$) goes down. 22
2.6 (a) Probability distribution function near $t\sigma$, (b) *failure sigma* for N-bit SRAM monitored, k, and probability P_t. 24
2.7 Our modified ADM/WRM flow for generating V_{ddmin} values, in which V_{th} skew offset is reflected on the ADM/WRM flow. 26
2.8 An illustration of V_{fail}-V_{ddmin} correlation table. 27
2.9 Example of an SRAM block structure and waveform of word line pulse affected by IR drop. Word line pulse is generated from control module, and propagated to selected word lines according to address bits. The pulse delivers to the cells one by one, from the first cell (red) to the last (blue). 28
2.10 Required sigma increases as word line pulse length decreases. 31
2.11 Histograms of all dies (blue) and dies with write failure (orange) according to word line pulse length. Each histogram is associated with V_{fail} group: (a) 0.56V, (b) 0.58V, (c) 0.60V, (d) 0.62V. 32
2.12 Histograms of all dies (blue) and dies with access failure (orange) according to word line pulse length. Each histogram is associated with V_{fail} group: (a) 0.56V, (b) 0.58V, (c) 0.60V, (d) 0.62V. 34
2.13 Extended flow of our proposed SRAM on-chip monitoring methodology to cope with access failure: (a) building-up L_{WL}-V_{ddmin} correlation table at design infra development phase, (b) deriving an SRAM $\hat{V}_{ddmin}$ on each die from V_{fail}-V_{ddmin} and L_{WL}-V_{ddmin} correlation tables at silicon production phase. 35
2.14 Ring oscillator for word line pulse length monitoring. Transistors on the path generating word line pulse from control module are extracted to build reduced control module. 36

2.15 (a) Quadratic interpolation between 100 spice simulation results of ring oscillator frequency and word line pulse length. (b, c) 3σ local worst word line pulse length prediction results: (b) considering global variation only, and (c) considering local random variation induced noise in ring oscillator measurement. 38
2.16 An illustration of L_{WL}-V_{ddmin} correlation table that is added to V_{fail}-V_{ddmin} correlation table. 38
2.17 Comparison of the values of V_{ddmin} (② orange dotted lines) and $\hat{V}_{ddmin}$ (④ red lines and ⑤ purple line) computed by our prediction flow for 1000 dies for 99.9% yield constraint with the values of V_{ddmin} (③ gray dotted lines) and $\hat{V}_{ddmin}$ (⑥ black line) computed by the conventional flow using [31, 32]. 43

3.1 (a) An HDL verilog description. (b) The flip-flops with mux-feedback loop synthesized for the code in (a). (c) The logic structure for (b) supporting idle logic driven clock gating. (d) The logic structure supporting data toggling driven clock gating. (e) The structure of ICG(Integrated Clock Gating cell). 50
3.2 (a) Flip-flop dependency graph of circuit containing three FFs *with* one self-loop FF. (b) Minimal allocation of retention storage for (a). (c) Minimal allocation of retention storage for (a), assuming the self-loop FF as a FF with no self-loop. 51
3.3 Two signal flow paths to Q_t at cycle time t in the self-loop FFs, which are implemented with (a) mux-feedback loop and (b) idle logic driven clock gating. 53
3.4 The changes of the portion of steady self-loop FFs in simulation as the circuits gracefully move to sleep mode. 53

3.5 The normalized saving of total retention storage size and total number of retention FFs for wakeup latency l set to 1, 2, 3, 4, and 5, which shows that l = 2 or 3 suffices. 56
3.6 Classification and deployment of retention bits on flip-flops in the three steps of our strategy of retention storage allocation with $l = 3$. 57
3.7 State monitoring circuitry for the flip-flops in $\mathcal{F}_{loop}^{steady}$ with no retention storage (①), power gating controller (②), and resource sharing with clock gating logic (③). 60
3.8 Timing diagram showing the transition to sleep mode by monitoring (*pg_en*) in ① for l (= 3) clock cycles. 62
3.9 State transition diagram for the power gating controller in ②. 62
3.10 The changes of total energy consumption as the values of $prob_f$ and ρ vary. Energy consumption is normalized to that of [24]. Our simulation in Step 1 corresponds to energy curve between blue and purple curves, since we selected a set of self-loop FFs for every benchmark circuit so that the $prob_f$ value became nearly 0. 64
3.11 Retention storage in f_1 can be reduced from (a) 3-bit to (b) 2-bit if *retention storage refinement condition* is satisfied. 65
3.12 State monitoring logic insertion scheme for (a) 3-bit to 2-bit reduction and (b) 2-bit to 1-bit reduction. State monitoring logic is newly inserted only when there is no pre-existing state monitoring logic in the fanin path of last flip-flop (f_3 in (a), f_2 in (b)). 68
3.13 Timing diagram of control signals and states of each flip-flops after retention storage refinement in Fig. 3.11. 70
3.14 Flow of our retention storage allocation and state monitoring circuit generation methodology. 71

3.15 Layouts for MEM_CTRL. The colored rectangles represent flip-flops: flip-flops with no retention storage (white), flip-flops with 1-bit retention storage (yellow), and flip-flops with 2-bit retention storage (red). 74
3.16 Detailed comparison of cell area in each method for each design with (a)∼(d) $l = 2$ and (e)∼(h) $l = 3$. 78
3.17 Detailed comparison of normalized standby power in each method for each design with (a)∼(d) $l = 2$ and (e)∼(h) $l = 3$. 81
3.18 Spice simulation generating pg_en signal through state monitoring logic for circuit MEM_CTRL. 83
3.19 Power connection to flip-flops whose retention storage are allocated by proposed method supporting immediate power gating. 84
3.20 Detailed comparison of normalized standby power consumed by each cell type in each of power modes when wakeup latency l is 3. 87
3.21 The changes of total energy consumption as the values of r_I and ρ vary, while γ is fixed to 0.02. Energy consumption is normalized to that of [24]. 88

Chapter 1

Introduction

1.1 Low Voltage SRAM Monitoring Methodology

As CMOS technology entered the sub-micron era, supply voltage (V_{dd}) reduction becomes stagnant, whereas chip size reduction and performance improvement have been continued. This is due to the non-scalability of threshold voltage (V_{th}) and the underlying limits on the sub-threshold slope of transistors. As a result, energy and power dissipation becomes the biggest barrier of technology scaling. In order to resolve this issue, low power design by near-threshold voltage (NTV) operation becomes attractive recently. NTV (i.e., $V_{dd} \gtrsim V_{th}$) operation entails a reasonable trade-off between energy efficiency improvement and performance degradation in comparison with current super-threshold voltage (i.e., $V_{dd} \gg V_{th}$) operation and sub-threshold voltage (i.e., $V_{dd} < V_{th}$) operation. Therefore, NTV operation could be a more practical alternative to low power design. However, there are several barriers for the use of NTV operation, one of which is the significant increase of embedded static random-access memory (SRAM) functional failure, in short, SRAM failure.

Data may be flipped while performing read operation (read failure) and data may be fixed to a specific value while performing write operation (write failure). These are two major SRAM failures [1]. As shown in Fig. 1.1, the probability of read and

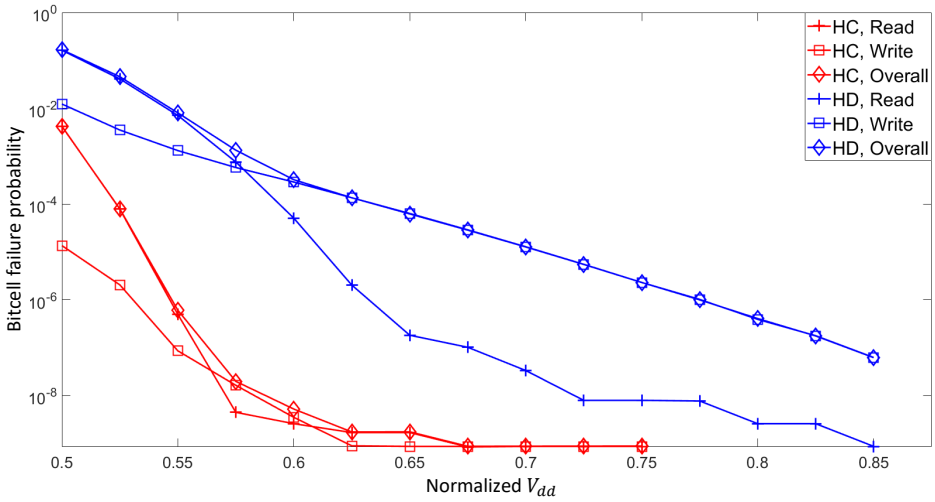


Figure 1.1: Probability of read, write, and overall operation failures on 14nm HC (High-Current) and HD (High-Density) bitcells [4]. V_{dd} is normalized to nominal voltage.

write failure on an SRAM bitcell increases dramatically as V_{dd} decreases, indicating that it is important to resolve SRAM failure issue in order to adopt NTV operation for low power design. Besides the read and write failures, SRAMs designed for high performance may experience failure while performing read operation due to insufficient timing margin (i.e., access failure). This can also limit the V_{ddmin} or operating speed on SRAM in NTV operation. The SRAM failure issue has been tackled in several research directions, including redesign of bitcell for NTV, read and write assistance scheme, and bitcell monitoring [2]. In addition, a simple but practical way to mitigate SRAM failure for NTV operation is to apply a higher V_{dd} to SRAM bitcell than that to logic circuit [3]. Two fundamental concerns regarding SRAM operation are (1) how much high V_{dd} is suitable to prohibit SRAM failure while logic circuit is operated on NTV regime? and (2) are there any systematic procedure that is able to achieve energy efficiency without sacrificing SRAM failure?

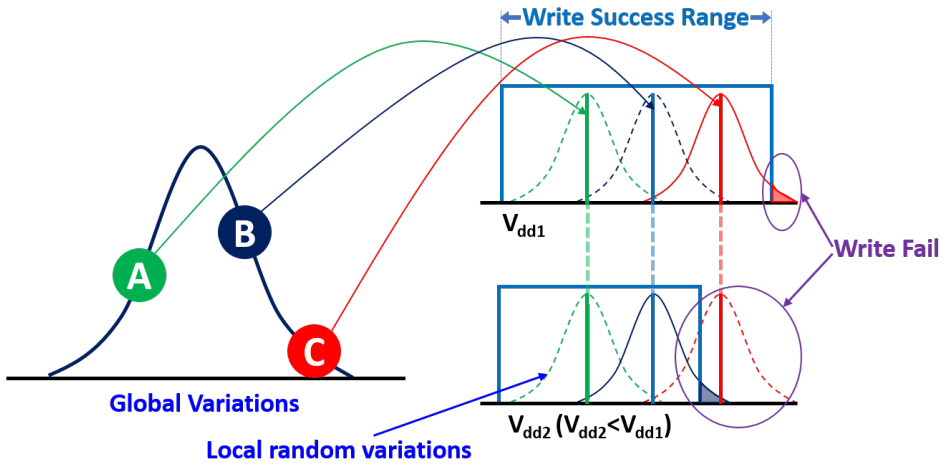


Figure 1.2: Dies with different global corners exhibit different rates of SRAM failure, though they have an identical local random variation.

The SRAM failure can be explained by process variations, which are usually classified as global variation and local random variation. Suppose that dies A, B, and C in Fig. 1.2 are located in different global corners and all three dies get the same amount of local random variation. Then, write fail will occur only in die C at voltage level V_{dd1}, since die C gets the global variation in the most vulnerable direction to write failure. When V_{dd} is lowered to V_{dd2}, an additional write failure occurs on die B, since the global variation on die B becomes vulnerable to write failure. This means die B can operate on a lower voltage than die C, and die A can operate on a lower voltage than both of dies B and C. The illustration in Fig. 1.2 indicates that V_{dd} *for SRAM bitcell with no SRAM failure depends on global variation.* Consequently, if we can estimate a minimum operation voltage, V_{ddmin}, to SRAM on each die under a tight confidence level, we can control SRAM bitcell V_{dd} for each die adaptively, like adaptive voltage scaling (AVS) scheme for logic, to achieve an energy saving on the die.

To control V_{dd} of SRAM bitcell on each die adaptively, it is necessary to be able to monitor and detect stability of SRAM blocks on each die. There has been no research on supplying V_{ddmin} of SRAM bitcell on each die by monitoring an SRAM block,

but there are research results that monitored individual SRAM block and controlled V_{dd} for those individual SRAM blocks for yield improvement. Mojumder *et al.* [5] designed a self-repairing SRAM with read stability and writability detectors to monitor an SRAM block. They improved yield by controlling word line voltage and bitcell voltage if failure is expected by the detectors. It is well suited for yield improvement of a few big SRAM blocks in microprocessor by adaptively controlling supply voltages of individual SRAM blocks. However, it is not suitable for finding V_{ddmin} of SRAM on each system-on-chip (SoC) die, where lots of SRAM blocks with different size and configuration (e.g. number of rows and columns) exist. Also, there are research results to monitor SRAM for resolving reliability issues. Ahmed and Milor [6] proposed an on-chip monitoring method that can monitor aging of bitcells in real time by modifying peripheral structure of SRAM. Wang *et al.* [7] showed an impact of peripheral circuit aging on SRAM read performance by designing monitoring circuit based on silicon odometer [8]. Jain *et al.* [9] proposed read and write sequence that can minimize the recovery during the accelerated aging test of SRAM. However, the monitoring methods proposed to solve reliability issue [6, 7, 9] can also only monitor and analyze one SRAM block which is being monitored. In summary, the above mentioned previous researches were focused on improving yield or resolving reliability issues for a targeted SRAM block. However, they are not efficient for monitoring an SRAM block to find V_{ddmin} of SRAM which can cover all different size and configuration of SRAM blocks in SoC die.

As somewhat related researches for seeking energy efficient SRAM operation, there has been other approaches including the charge recycling techniques for SRAM design. They modified peripheral circuit [10, 11, 12, 13] or bitcell [10, 13] to reduce the bit line voltage swing by reused charge. However, the charge recycling techniques are design methods that can be used by SRAM bitcell designers and circuit designers while designing SRAM architectures. Whereas, our work is a methodology that can be built by chip designers in design infra development phase and used it for optimizing

SRAM supply voltage of each die in silicon production phase.

1.2 Retention Storage Allocation on Power Gated Circuit

Regardless of supply voltage reduction coupled with process node shrinkage, which is stagnated recently, reducing the leakage power always been an important issue and has become more and more important for low power modern chips as semiconductor process node shrinks. Power gating, which is a technique to shut off the power on a chip when it's not in active (i.e., in sleep mode), is one of the most commonly used low power design techniques for saving leakage power [14]. Fig. 1.3 shows the structure of circuit with power gating, in which virtual VDD (VVDD) of circuit can be shut off by sleep signal. By turning off VVDD and only supplying VDD to cells that must operate during sleep mode, leakage power consumed by the power gated block can be saved. However, one reverse side of the benefit of power gating is that it requires the always-on high-Vth storage for retaining the state of flip-flops during the sleep mode, so that the circuit state can be restored when waking up [15].

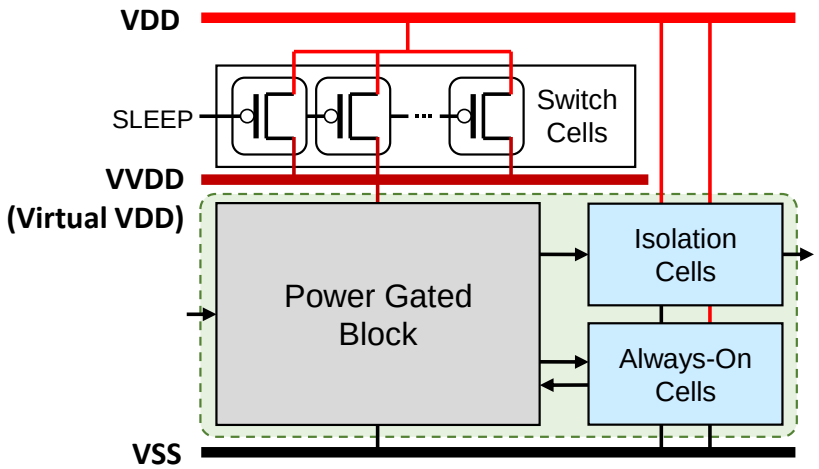


Figure 1.3: The structure of circuit with power gating.

It is shown in [16] that simply allocating a distinct single retention bit (i.e., 1-bit)

storage to every flip-flop in circuit is generally expected to have more than 10% area increase. (We call such flip-flops single bit retention flip-flops (SBRFFs).) Since the state retention storage consumes leakage power (called *standby* power) even when the circuit is in sleep mode, it is very important to minimize the total storage size.

The concept of *selective state retention* has been adopted by a number of works (e.g., [17, 18, 19, 20]), which retains only a minimal number of flip-flop states that are necessary to restore the circuit state when waking up. Sheets [17] defined *checkpoints* as the possible states when they do not change on the next clock cycle, which is given by circuit designer or figured out by analyzing the next state logic. From the analysis of read and write patterns, all states are classified according to whether they are reused after each checkpoint or not, thereby reducing the resource overhead for maintaining circuit state in sleep mode. On the other hand, Greenberg *et al.* [18, 19] used gate-level simulation [18] and formal verification [19] to extract the flip-flops, called *non-essential* flip-flops, whose states never help in recovering the circuit state. They searched for the flip-flops having always the same state value as that in the pre-standby phase, overwritten before read, or never being read in the post-standby phase. Chiang *et al.* [20] proposed to find non-essential registers by applying RTL symbolic simulation using real test sequences [21], for which they converted the circuit into a set of conjunctive normal forms (CNF) and formulated the problem into a satisfiability (SAT) problem.

On the other side, Chen *et al.* [16, 22] proposed a structure of **multi-bit retention flip-flop (MBRFF)** as shown in Fig. 1.4. They extracted flip-flops from circuit as minimal as possible and replaced them with l-bit MBRFFs while satisfying the constraint that the state restoration should be processed by shifting-out the data in the l-bit storage in MBRFFs through l-cycle execution of circuit when waking up. Lin and Lin [23] solved the problem of allocating a minimal number of l-bit MBRFFs by formulating it into an ILP (Integer Linear Programming).

To further reduce the total retention storage, Fan and Lin [24] allows every flip-flop

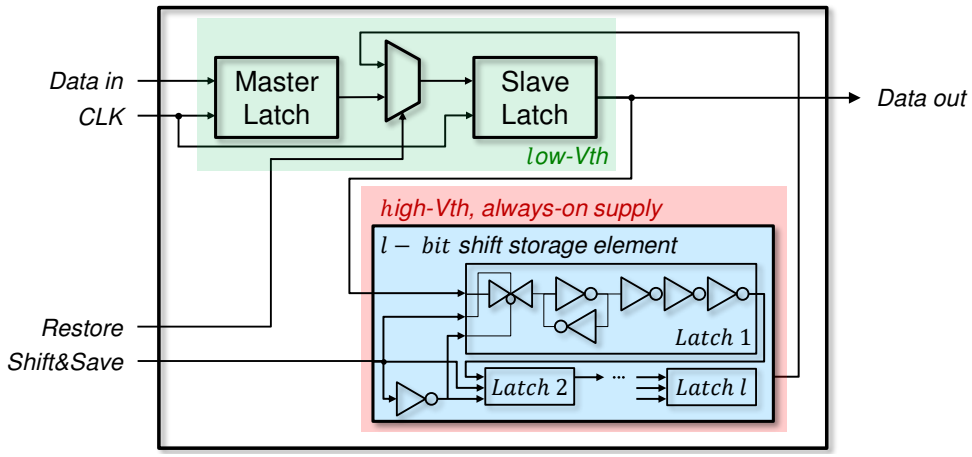


Figure 1.4: The structure of multi-bit retention flip-flop (MBRFF) that can save $l > 1$ retention bits [22].

to use any of none, 1-bit, 2-bit, $\cdots$, l-bit retention storages as opposed to constraining to none or l-bit storage only. They proposed an ILP based heuristic approach to the problem of *non-uniform* MBRFF allocation, in which starting from the SBRFF allocation to all flip-flops, they iteratively applied their ILP formulation to replace more than one short-bit SBRFF/MBRFF into a long-bit MBRFF with less total bits. Recently, Hyun and Kim [25, 26] elaborated the wakeup operation of SBRFF so that its state restoration can also be triggered in the second (i.e., one-cycle delayed) clock cycle to boost up the exploitation of 1-bit data in SBRFFs for the circuit state recovery. Kim and Kim [27] transformed the problem to unate covering problem to find optimal allocation with three different objectives: minimal retention storage, leakage power consumption, and area.

Though considerable efforts have been made by the prior works, the amount of reducing state retention storage is within a limited bound. The main reason is due to the abundant presence of flip-flops with *mux-feedback loop* in the circuit since each of them should have at least one bit of state retention storage to restore its state in wakeup mode. (It will be described in detail in Sec. 3.1.1).

1.3 Contributions of this Dissertation

It has always been an important issue to operate chip at low power while ensuring its functionality. In this dissertation, we propose low power design methodologies with different approaches for each of two parts of chip: SRAM (Chapter 2) and logic (Chapter 3).

In Chapter 2, we propose an SRAM on-chip monitoring methodology, in which V_{ddmin} for prohibiting SRAM failure on each die can be accurately derived by analyzing V_{fail} measured by the SRAM monitor on the same die [28, 29]. Monitoring is done only once per a chip to estimate $\hat{V}_{ddmin}$ of SRAMs under process variation. Then AVS is applied to each chips for energy efficient memory operation, while assuming the reliability issue caused by aging is handled by aging-aware signoff [30]. Note that *monitoring the chip performance and reducing energy consumption by applying AVS to logic circuits have been studied by many researchers, but to our best knowledge, this is the first work in the context of SRAM monitoring at NTV.* The contributions and features of our work are the following:

1. We propose to find SRAM $\hat{V}_{ddmin}$ of each die to prohibit SRAM failure while logic circuit is operated on NTV regime. As a result, energy efficient memory operation on NTV regime is possible without increasing SRAM failure.

2. We propose an SRAM monitor and a methodology to measure the highest voltage, V_{fail}, for incurring SRAM monitor failure with no modification of the structure of SRAM bitcells, which otherwise may distort the inherent variation characteristics of SRAM.

3. We develop a novel methodology to estimate V_{ddmin} that is the lowest V_{dd} for prohibiting SRAM read and write failures on the same die, in which we modify the ADM (Access disturb margin) and WRM (write margin) extraction flow [31, 32] to derive global and local random variations on target SRAM from the failure voltage data observed by the SRAM monitor.

4. We extend our methodology to take into account the effect of IR drop and process variation of peripheral circuit on SRAM bitcell operation, and the potential SRAM access failure as well as the SRAM read and write failures.

In Chapter 3, we overcome the inherent limitation of retention storage allocation for the flip-flops with mux-feedback loop by introducing a concept of *steady state driven allocation* [33, 34]. Through gate level simulation, we find a condition where retention storage allocation can not constrained by flip-flops with mux-feedback loop. Retention storage is minimally allocated by utilizing the condition, and state monitoring circuitry is inserted to detect the condition where power gating is available under the allocated retention storage. The contributions and features of our work are the following:

1. We identify a crucial observation regarding the circuit behavior when circuits are about to switch to sleep mode. To be a safe transition, power gating controller maintains a short grace time period during which steady (primary) inputs should be issued to the circuits. This behavior enables us to characterize and classify the state pattern of the flip-flops, which in turn provides a useful clue to break the bottleneck of minimizing the state retention storage.

2. We propose a novel state monitoring mechanism based on the analysis of the circuit behavior, by which we break down the barrier in power gating, which is invariably allocating the expensive retention storage to every flip-flop with mux-feedback loop.

3. We propose a novel retention storage refinement method, which can reduce the retention storage further after the initial retention storage allocation by utilizing state monitoring circuitry.

4. We propose a method of hardware resource sharing to minimize the implementation cost of our power gating by utilizing the implementation logic for data toggling driven clock gating.

It should be noted that the methods proposed in each chapter are applicable to standard chip design and production flows. SRAM on-chip monitoring methodology, which will be discussed in Sec. 2.2.1 and 2.3.3, creates a correlation table during chip design and uses the monitoring results to refer the table during chip production. The monitoring results are measured through memory BIST (built-in self test) logic and ring oscillator, all of which are already used for chip monitoring, and the subsequent correlation table referencing can be done in a short period. Therefore, the method proposed in Chapter 2 can be applied to the chip design and production flows in practice. Retention storage allocation in Chapter 3 is part of the standard flow for low power design. Retention storage allocation and subsequent retention cell mapping are performed in RTL synthesis stage as shown in Fig. 1.5(a). For fine-grained retention storage allocation, however, since it requires knowledge of the connections between flip-flops, it can be done in the re-synthesis stage of gate-level netlist after technology mapping, as shown in Fig. 1.5(b). Proposed method in Chapter 3 is compatible with the standard design flow because only the stages colored red in the figure are modified while not changing the overall flow.

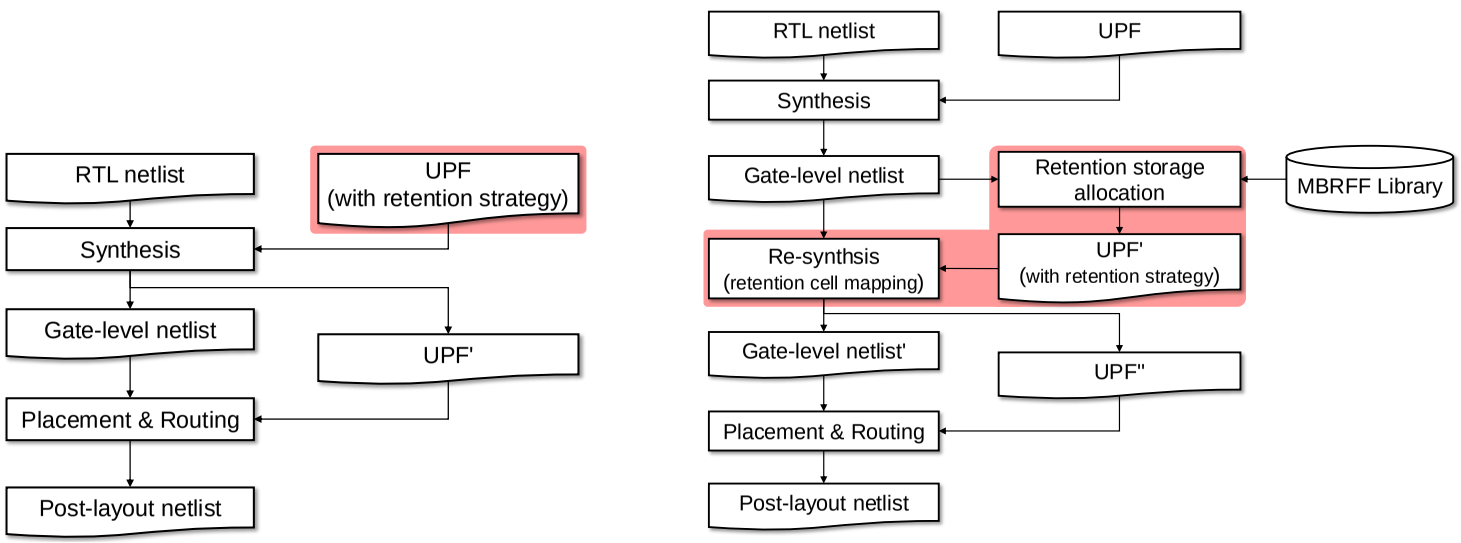


Figure 1.5: Standard flows for low power design, which support retention with power gating.

Chapter 2

SRAM On-Chip Monitoring Methodology for High Yield and Energy Efficient Memory Operation at Near Threshold Voltage

2.1 SRAM Failures

An SRAM bitcell consists of 6 transistors as shown in Fig. 2.2: two inverter pairs (PUL-PDL, PUR-PDR) and their access transistors(AXL, AXR). Within-die (local) variation causes mismatch between different transistors in an SRAM bitcell, degrading stability of bitcell and resulting in bitcell failure. SRAM bitcell failure can be classified into four categories: read failure, write failure, access failure, and hold failure.

2.1.1 Read Failure

Read failure, also referred to as destructive read or read flip, is the failure that data stored in a bitcell is lost on a read operation (Fig. 2.1(a)). For read operation, the bit line pair are precharged to V_{dd} and the word line is triggered to high state. Then, access transistor of the node storing "0" (AXR in Fig. 2.2) is turned on, and discharge the bit line $\overline{BL}$. AXR and PDR act as voltage divider during the read operation, making the voltage of node QB higher than 0. If the voltage of node QB becomes higher than

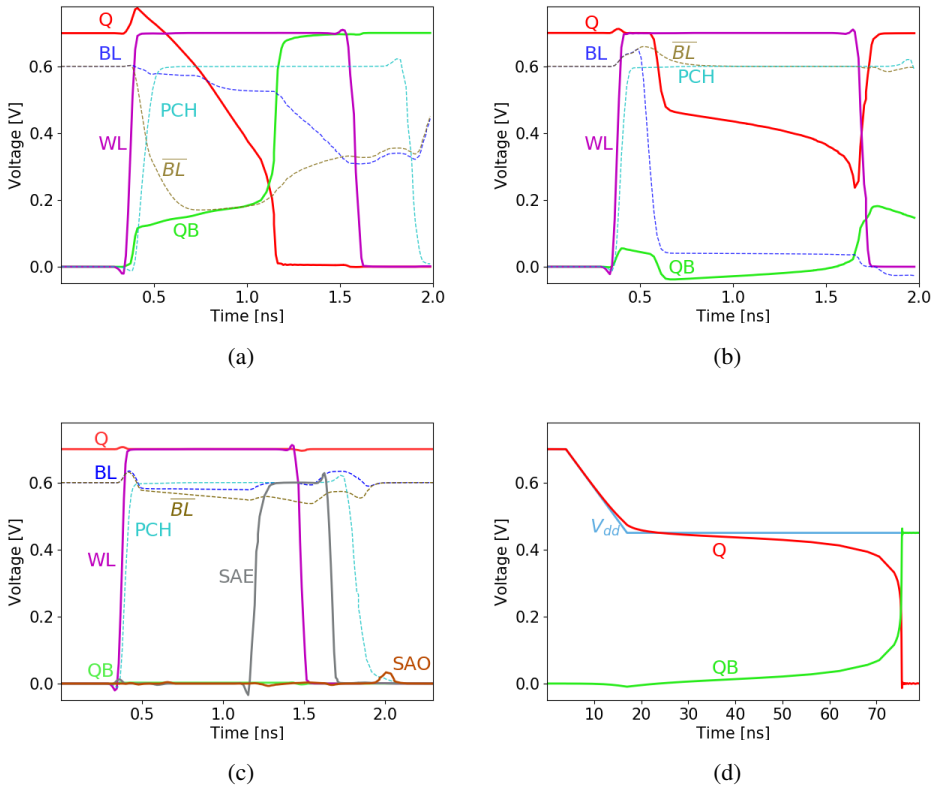


Figure 2.1: Waveform of SRAM bitcell failures: (a) read failure, (b) write failure, (c) access failure, (d) hold failure. V_{dd} of peripheral circuit and bitcell are 0.6V and 0.7V, respectively.

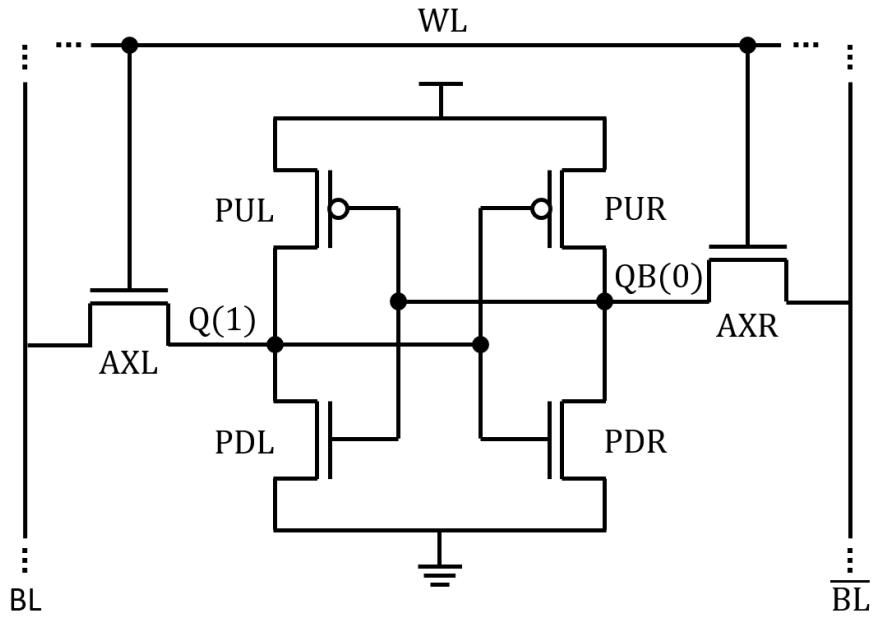


Figure 2.2: 6T SRAM bitcell storing data “1”

the tripping voltage of PUL-PDL inverter due to mismatch between bitcell transistors, voltage of node Q and QB are flipped, resulting in the destruction of data.

2.1.2 Write Failure

Write failure or unsuccessful write is the failure that data cannot be written to bitcell (Fig. 2.1(b)). For write operation, the bit lines are biased to V_{dd} or GND according to data to be written, and the word line is triggered to high. For example, to write “0” to node Q in Fig. 2.2, BL and $\overline{BL}$ are biased to 0 and V_{dd}, respectively, while WL is triggered to high. Then, the access transistors are turned on, pull down the voltage of node Q to GND through BL, and finally write data “0” to bitcell. However, mismatch in bitcell transistors can cause the write failure such that write operation is incompleted while the word line is high, or data cannot be written regardless of the word line pulse length.

2.1.3 Access Failure

For successful read operation, voltage difference between the bit line pair must be large enough to be detected by the sense amplifier. Access time is defined as the time taken to produce sufficient voltage difference between bit line pair, which is generally more than $0.1V_{dd}$. If access time is longer than maximum tolerable time due to process variation, it cannot be sensed by sense amplifier, causing access failure as shown in Fig. 2.1(c), in which voltage difference between BL and $\overline{BL}$ is not enough for sensing, causing voltage of SAO (sense amp. output) not being pulled up to V_{dd} though the bitcell is storing "1".

2.1.4 Hold Failure

Due to the high leakage power for always-turning-on SRAM, V_{dd} of SRAM is lowered in retention mode to reduce power consumption rather than staying on high V_{dd} for long stand-by cycles. However, bitcell margin becomes lower as the supply voltage is reduced. For example, if supply voltage of bitcell is reduced, then voltage of node Q in Fig. 2.2 becomes lower. It can be lowered further due to the leakage in PDL, even lower than tripping voltage of PUR-PDR inverter. In that case, data stored in the bitcell is lost as described in Fig. 2.1(d), which is referred to hold failure.

Among the four different SRAM bitcell failures, we focus on prohibiting read and write failures, which are majority (almost 100%) of bitcell failures in real world [35]. In addition, we extend the scope of our study to potential access failure which can be an additional issue for high-speed designs. However, since the voltage that incur hold failure is lower than retention mode voltage, SRAM bitcell on operating mode voltage is tolerant to process variation for hold failure. Thus, hold failure will not be covered in this paper.

Process variation that we considered to analyze SRAM failure are described in Table 2.1 and 2.2. Among FEOL part of SRAM block, only process variation on bitcell

Table 2.1: Process variation on each part of the circuit considered

process variation on...		**considered?**
FEOL	bitcell	*yes*
	word line pulse generating circuit	*yes*
	others	*no*
BEOL	-	*no*

Table 2.2: Types of non-systematic process variation considered.

types of process variation		**considered?**
Die-to-Die	-	*yes*
Within-Die	independent	*yes*
	spatial	*no*

transistors, which is the analysis target, and transistors in the word line pulse generating circuit, which directly affects bitcell operation, are considered. However, process variation on BEOL part is not considered because our target is the effect of process variation on bitcell margin at transistor level only.

Process variation is classified into die-to-die (global) variation that affects differently to transistors in different dies but identically to transistors in the same die, and within-die variation that affects differently to transistors in the same die. In addition, within-die variation consists of independent (local random) variation that affects each of transistors randomly, and spatial variation that is induced by geometric relation between transistors. In this paper, under the assumption of negligible spatial variation, we only considered (1) global and (2) local variation because (1) our target is to find SRAM $\hat{V}_{ddmin}$ of each die to prohibit SRAM failure, and (2) the stability of each bitcell is affected by the random variation of each bitcell transistors even on the same global variation basis.

2.2 SRAM On-chip Monitoring Methodology: Bitcell Variation

2.2.1 Overall Flow

Fig. 2.3 shows the overall flow of proposed methodology that finds SRAM $\hat{V}_{ddmin}$ of each die with the guidance of SRAM on-chip monitor, in which V_{fail}-V_{ddmin} correlation table is built-up at design infra development phase, and SRAM $\hat{V}_{ddmin}$ of each die is found at silicon production phase. The correlation table is built-up only once, and continuously referenced once per a chip to determine SRAM $\hat{V}_{ddmin}$.

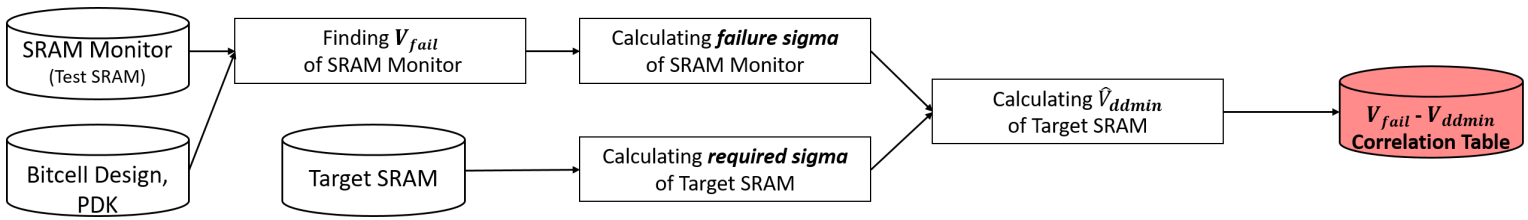
We assume a chip is designed at NTV regime, in which the supply voltage for logic is assumed to 0.6V and the supply voltage for SRAM bitcell is assumed to higher than 0.7V in 28nm process. The scheme of using higher supply voltage on SRAM bitcell than the voltage on logic is commonly used to mitigate SRAM functional failure at the low supply voltage regime [3]. In addition, we assume the SRAM peripheral uses the same voltage level as that on logic.

2.2.2 SRAM Monitor and Monitoring Target

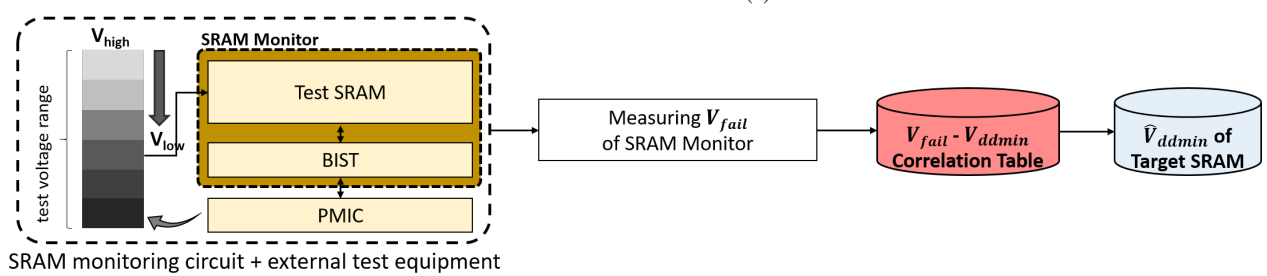
We use a normal SRAM block as an SRAM monitor (i.e., test SRAM), from which we infer $\hat{V}_{ddmin}$ of the SRAM blocks on a chip. Read and write failures of SRAM monitor can be monitored by using memory BIST (built-in self test) logic with test algorithm (e.g., MARCH[36]). From the SRAM monitor, we measure the failure voltage V_{fail}, which is the highest voltage that the number of bitcell failure exceeds pre-determined threshold value[1]. During the V_{fail} measurement in silicon production phase, voltage to be tested will be applied and swept through an off-chip test equipment.

An important concern is to determine the size of SRAM monitor. We observed that V_{ddmin} estimation result of proposed methodology increases reliability as the size of SRAM monitor increases, but there is a saturation point at which the V_{ddmin} estimation

[1]The determination of threshold value will be discussed in Sec. 2.2.3



(a)



(b)

Figure 2.3: Overall flow of our proposed SRAM on-chip monitoring methodology: (a) building-up V_{fail}-V_{ddmin} correlation table at design infra development phase, (b) deriving an SRAM $\hat{V}_{ddmin}$ on each die at silicon production phase.

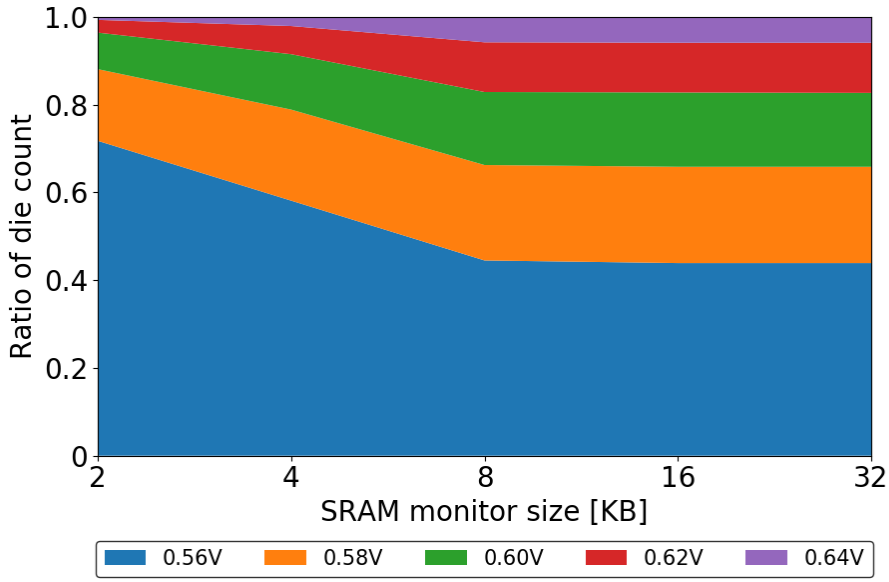


Figure 2.4: The changes of die count distribution in each V_{fail} group (0.56V∼0.64V) as the size of SRAM monitor increases.

result does not change beyond the point on increasing SRAM monitor size.

Our proposed methodology directly uses the measured V_{fail} of SRAM monitor in silicon production phase, and the V_{ddmin} decision is based on the V_{fail}-V_{ddmin} correlation table, which is constructed in design infra development phase. Since the V_{fail}-V_{ddmin} correlation table is based on statistical data from the SRAM monitor simulation results, the die count distribution for V_{fail} affects the final V_{ddmin} estimation result. Fig. 2.4 shows the changes of die count distribution in each V_{fail} group among 1000 dies as the size of SRAM monitor increases. In the figure, the die count distribution in each V_{fail} group starts to saturate when the SRAM monitor size exceeds 8KB. From the SRAM monitor simulation results, we decided the SRAM monitor size in our experiments to 16KB. Modern SoCs usually contain SRAM blocks of various sizes and total size exceeds 100Mb [37]. In addition, all SRAM blocks have their BIST circuits. Therefore, the area increased by 16KB SRAM monitor and its BIST circuit is negligible. Also, test time overhead induced by sweeping test voltage can be reduced by using

Table 2.3: Size, count, and other design parameters for target SRAM

size(bit)	count	CPW	RPB	APR	RDN
512	24	32	2	2	2
640	48	40	2	2	2
1040	69	65	2	2	2
1296	6	81	2	2	2
1440	24	45	4	2	2
2048	12	128	2	2	2
2560	207	80	4	2	2
3456	192	108	4	2	2
4864	24	76	8	2	2
6528	48	102	8	2	2
7680	48	64	15	2	2
9984	24	78	16	2	2
10240	72	80	16	2	2
46080	24	72	80	2	4
73728	24	128	72	2	4
139264	24	128	136	2	4
319488	288	128	156	4	4
344064	12	128	168	4	4

dual-rail voltage scheme [38] or testing multiple SRAM monitor simultaneously.

Target SRAM for V_{ddmin} estimation is all the SRAM blocks in a tested chip. In other words, V_{ddmin} is the lowest voltage that *all* SRAM blocks in the chip can operate without bitcell failures. We used OpenSPARC T1 processor [39] as a tested chip. However, we included new SRAM blocks so that the total SRAM size is close to 100Mb. Columns-per-WL (CPW), rows-per-BL (RPB), arrays-per-row (APR), and redundancy (RDN) in Table 2.3 are the number of columns connected to a word line in a bitcell sub-array, the number of rows connected to a bit line in a bitcell sub-array, the number of bitcell sub-arrays placed in a row in SRAM floorplan, and the number of redundancy to correct failed bitcells, respectively. These parameters are carefully

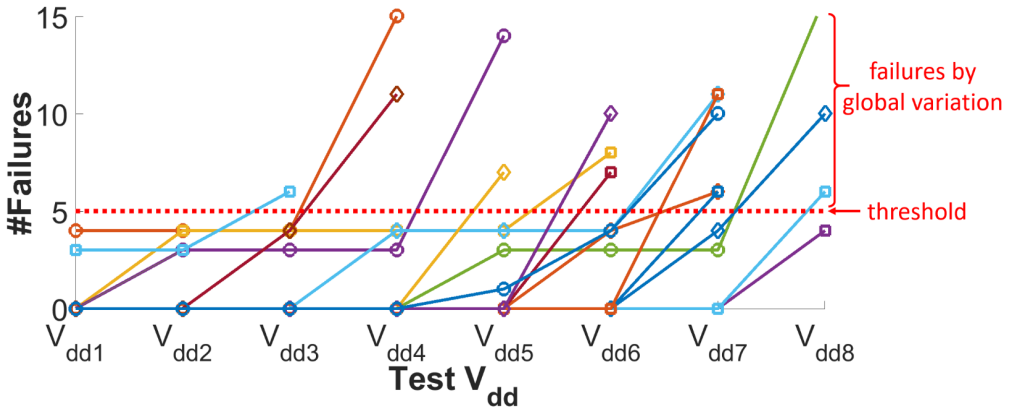


Figure 2.5: The changes of the number of bitcells with failure in the monitored test SRAM as the applied voltage V_{dd} ($V_{dd1} > V_{dd2} > \cdots > V_{dd8}$) goes down.

selected with the consideration of the memory structure of OpenSPARC T1 processor and the industry partner's memory design. In our work, we refer *target SRAM* to all SRAM blocks in Table 2.3, which are assumed to be placed in a chip[2].

2.2.3 V_{fail} to $\hat{V}_{ddmin}$ Inference

To derive $\hat{V}_{ddmin}$ from V_{fail} in silicon production phase, V_{fail}-V_{ddmin} correlation table is required. The correlation table is built-up in the design infra development phase. The building-up steps are shown in Fig. 2.3(a).

Finding V_{fail} of SRAM Monitor

We find V_{fail} of SRAM monitor by Monte Carlo Hspice simulation while varying the global corners. Besides the V_{fail} values, we take the number of bitcells with failures on each of the V_{fail} values to determine $\hat{V}_{ddmin}$ more accurately.

Note that V_{fail} refers to the maximum voltage on which the number of bitcells with failures exceeds a pre-determined threshold. The threshold value is determined by analyzing the failure trend on the monitored test SRAM i.e., the global corners

[2]The consideration of parameters will be discussed in Sec. 2.2.3

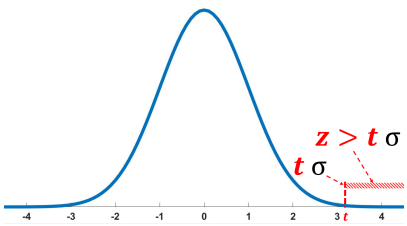
by the physical parameter variation. For example, Fig. 2.5 shows the changes of the number of bitcells with failures in the test SRAM for the applied voltage changes for each of 20 global corners on the SRAM. For some global corners, there is no increase on the number of bitcells with failure in a sub-range of the applied voltage. This is because such failures are caused by the extreme local random variation – random variation that is biased to the tail of distribution. For example in Fig. 1.2, extreme local random variation may cause some failures in die B at V_{dd1}, but the failures are not dominant to global variation. Marking V_{dd1} as V_{fail} enables global corner of die B to be inferred, which is the same as that of die C, causing pessimistic V_{ddmin} calculation. Thus, the threshold of the failure count that includes at least one failure contributed by global variation will be a little more than that by the local random variation. Since it is observed the maximum number of bitcells with failure by local random variation is 4 in our experiments, we can set the threshold to 5.

V_{fail} has a tight correlation with global variation under the assumption that the local random variations with different global variation are all identical, as explained in Fig. 1.2. Furthermore, we retain the number of bitcells with failure on V_{fail} for every instance of global variation tested in design time to utilize it for an accurate calculation of $\hat{V}_{ddmin}$ later whereas in the silicon production phase, we measure V_{fail} only.

Calculating *failure sigma* of SRAM monitor

We compute *failure sigma* of SRAM monitor through a probability analysis. Tentatively, we relax the assumption that the local random variation for every die is identical when deriving *failure sigma* for a test SRAM instance. *Failure sigma* is the largest local random variation expected to exist in the monitored SRAM with the highest probability. *Failure sigma* of each SRAM instance can be calculated as follows, using the number of bitcells failed on its V_{fail}:

$$P_t = 1 - \sum_{i=0}^{k-1} \binom{N}{i} \cdot cdf(t)^{N-i} \cdot (1 - cdf(t))^i \tag{2.1}$$



(a)

SRAM size [KB]	*k*	90%	99%	99.9%
16	5	3.83	3.74	3.67
	6	3.80	3.71	3.65
	7	3.77	3.69	3.63
32	5	4.00	3.91	3.85
	6	3.96	3.88	3.83
	7	3.94	3.86	3.81

(b)

Figure 2.6: (a) Probability distribution function near $t\sigma$, (b) *failure sigma* for N-bit SRAM monitored, k, and probability P_t.

where N is the number of bitcells in the monitored SRAM, k is the number of bitcells with failure observed on V_{fail}, and $cdf(\cdot)$ is the cumulative distribution function of local random variation. Eq.(2.1) computes the probability that the k^{th} worst local random variation exists in the region $z\sigma(z > t)$ in the N-bit SRAM when k bitcells are failed in read or write, as indicated in Fig. 2.6(a). For N and k, we determine t with 99.9% probability and use it as the value of *failure sigma*. An illustrating data is shown in Fig. 2.6(b) where for example, if 6 failures are observed on V_{fail} in a 16KB SRAM, there exists local random variation bigger than 3.65σ with 99.9% probability.

Calculating *required sigma* of target SRAM

Required sigma refers to the amount of local random variation that the target SRAM should be tolerant in read and write operation to satisfy target yield (e.g., 99.9%). *Required sigma* of target SRAM can be obtained by estimating the size of local random variation by iteratively computing Eqs.(2.2)∼(2.5) until the yield becomes 99.9%:

$$P_{CELL} = 2 \cdot (1 - cdf(M)) \tag{2.2}$$

$$P_{COL} = 1 - (1 - P_{CELL})^{N_{ROW}} \tag{2.3}$$

$$P_{MEM} = \sum_{i=N_{RC}+1}^{N_{COL}+N_{RC}} \binom{N_{COL}+N_{RC}}{i} \cdot P_{COL}^{i} \cdot (1-P_{COL})^{N_{COL}+N_{RC}-i} \tag{2.4}$$

$$\begin{aligned} Yield &= 1 - P_{MEM} \\ &= \sum_{i=0}^{N_{RC}} \binom{N_{COL}+N_{RC}}{i} \cdot P_{COL}^{i} \cdot (1-P_{COL})^{N_{COL}+N_{RC}-i} \end{aligned} \tag{2.5}$$

where M represents the maximum local random variation that the target SRAM can operate normally, P_{CELL}, P_{COL} and P_{MEM} are failure probabilities of a bitcell, column and SRAM block, N_{ROW}, N_{COL} are the numbers of rows, columns in SRAM block which are calculated from the parameters in Table 2.3, and N_{RC} is redundancy of SRAM block which is the same as RDN in Table 2.3. Since the yield computed by Eq.(2.5) corresponds to a single SRAM block, and target SRAM includes all SRAM blocks in Table 2.3, the final yield should be computed by multiplying the yields of all SRAM blocks. To meet 99.9% yield constraint for the SRAM blocks in Table 2.3, SRAM bitcell should be tolerant to 5.04σ local random variation.

Calculating V_{ddmin} of target SRAM

This step builds up V_{fail}-V_{ddmin} correlation table that will be used for extracting $\hat{V}_{ddmin}$ at the production phase. We accelerate the building-up process by applying a modified ADM/WRM flow shown in Fig. 2.7.

Note that ADM (Access disturb margin) and WRM (Write margin) flow [31, 32] are widely used in industry due to its low computational complexity and the capability of direct estimation to yield [40]. ADM and WRM are the largest local random variation of V_{th} that a bitcell can operate normally. The main purpose of using the conventional ADM/WRM flow is to evaluate the stability of bitcell against local random variation in the course of designing a bitcell while assuming a global worst corner.

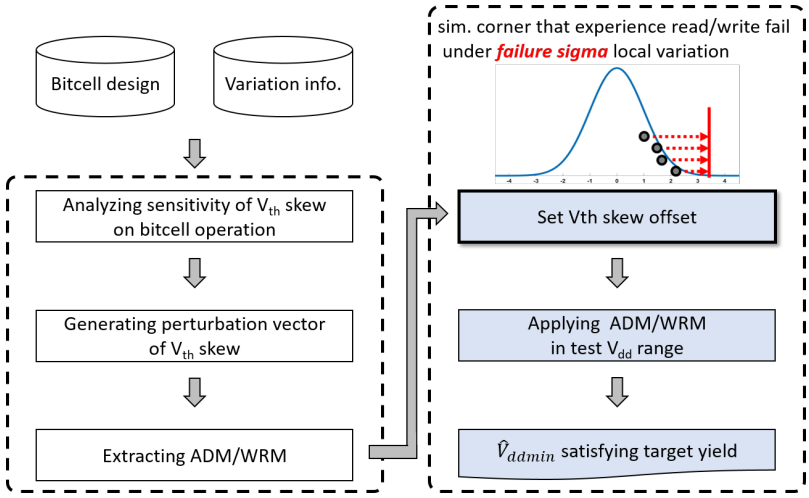


Figure 2.7: Our modified ADM/WRM flow for generating V_{ddmin} values, in which V_{th} skew offset is reflected on the ADM/WRM flow.

However, our interest in this work is to find a global corner of target SRAM by examining the data measured by SRAM monitor. Consequently, we attach additional processes to shift the simulation corner in ADM/WRM flow, so that it runs under the process variation, which is expected to be the same as that in the test SRAM.

The conventional ADM/WRM flow consists of 3 parts, which are the three boxes on the left side in Fig. 2.7 [32]: (1) analyzing the sensitivity of V_{th} skew on bitcell operation, (2) generating V_{th} unit perturbation vector for bitcell transistors ($U_{V_{th}}$) based on the analysis, and (3) monitoring failure in actual read and write operation on a bitcell with V_{th} skew variation:

$$\Delta V_{th} = U_{Vth} \times \sigma(V_{th}) \times (ADM \mid WRM) \tag{2.6}$$

where $\sigma(V_{th})$ is standard deviation of V_{th} of the bitcell transistors, and the last term is ADM or WRM value under test. Note that the largest value of the last term with no read or write failure will be the final value of ADM or WRM .

Our modified ADM/WRM flow is shown on the right side in Fig. 2.7. First, we

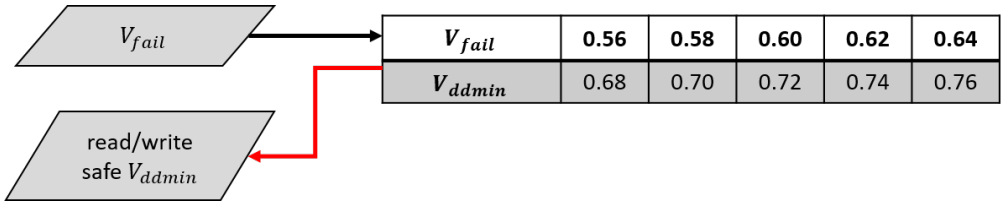


Figure 2.8: An illustration of V_{fail}-V_{ddmin} correlation table.

calculate V_{th} skew offset, which will become an initial V_{th} skew of bitcell transistors:

$$V_{th\_offset} = (ADM \mid WRM - failure\ sigma) \times U_{V_{th}} \tag{2.7}$$

Note that bitcell voltage is fixed to V_{fail} on which the *failure sigma* of SRAM monitor was extracted. While considering the V_{th} skew offset vector, we find the lowest voltage, V_{ddmin}, with no read and write failure. The V_{th} skew of bitcell transistors is computed by:

$$\Delta V_{th} = V_{th\_offset} + U_{V_{th}} \times \sigma(V_{th}) \times (required\ sigma) \tag{2.8}$$

where $U_{V_{th}}$ is extracted every time the supply voltage changes. The V_{th} skew offset is fixed to the value obtained during the process of finding V_{fail} by SRAM monitor. This is because the impact of the process variation on the operation of transistors varies depending on the supply voltage.

From the collected data of V_{ddmin}, we build a V_{fail}-V_{ddmin} correlation table as shown in Fig. 2.8. In silicon production phase, we select the voltage, i.e., $\hat{V}_{ddmin}$ from the V_{fail}-V_{ddmin} correlation table that corresponds the V_{fail} value measured by the SRAM monitor.

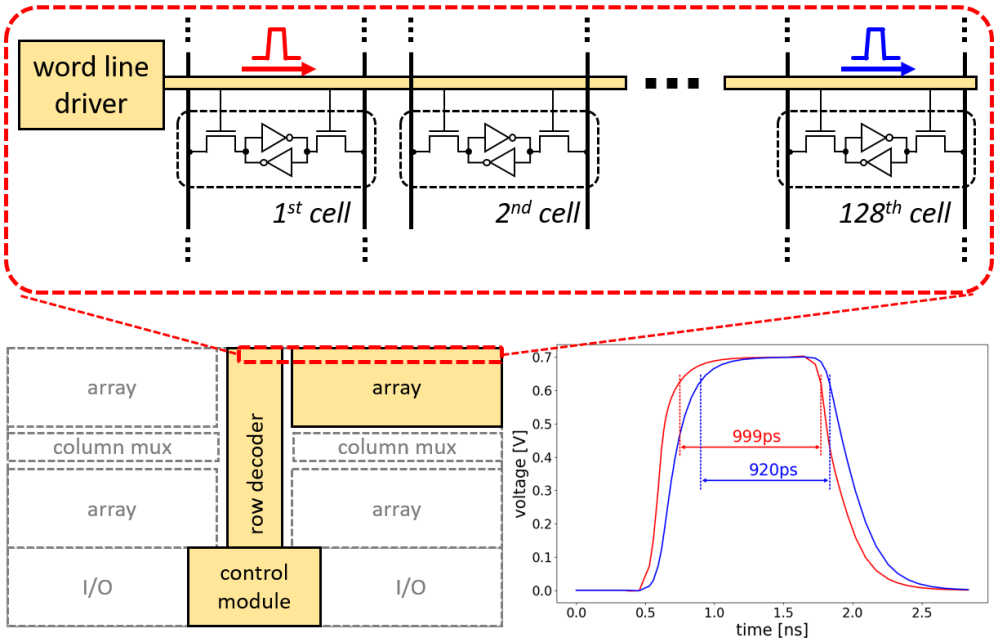


Figure 2.9: Example of an SRAM block structure and waveform of word line pulse affected by IR drop. Word line pulse is generated from control module, and propagated to selected word lines according to address bits. The pulse delivers to the cells one by one, from the first cell (red) to the last (blue).

2.3 SRAM On-chip Monitoring Methodology: Peripheral Circuit IR Drop and Variation

2.3.1 Consideration of IR Drop

Fig. 2.9 shows an example of SRAM block structure. Word line pulse is generated from control module, and propagated to selected word lines through row decoder according to address bits. The word line pulse is buffered by word line driver before passing word line, and turns on access transistors of bitcells connected to word line one by one, from the first cell to the last cell (maximum 128^{th} in our experiments). As process advances, per-unit-length resistance of metal is increasing because of thinner metal width. For example, per-unit-length resistance of 7nm process increases about 9 times to that of 28nm process[41]. This leads to a significant IR drop in word line pulse, which causes functionality issue in bitcells which are far apart from the word line driver [42].

Waveform of IR drop affected word line pulse is shown on the right side in Fig. 2.9. Red waveform is the word line pulse arrived at a bitcell closest to word line driver, and blue waveform is the pulse arrived at a bitcell farthest from word line driver. The word line pulse length of the first cell is 999ps. However, the length is changed to 920ps at the last cell (128^{th} cell) because of IR drop. Because bitcell margin becomes smaller as bitcell locates farther away from the word line driver, *required sigma* should be adjusted higher than the original value. We performed spice simulation for a word line with the consideration of IR drop and calculated margin of each bitcell. Then, we calculated local variation that a bitcell should withstand to meet yield constraint under IR drop by Eqs.(2.9)$\sim$(2.11).

$$P^{i}_{CELL} = 2 \cdot (1 - cdf(M^{i})) \tag{2.9}$$

$$P^{j}_{COL} = 1 - (1 - P^{i}_{CELL})^{N_{ROW}} \tag{2.10}$$

$$Yield = \sum_{k=0}^{N_{RC}} \sum_{T \in S_k} \prod_{j \in S} u^{(j,T)}$$
$$\text{where } u^{(j,T)} = \begin{cases} 1 - P_{COL}^{j}, & \text{if } j \in T \\ P_{COL}^{j}, & \text{otherwise} \end{cases} \tag{2.11}$$

M^i and P_{CELL}^i are margin and failure probability of i^{th} bitcell from word line driver, P_{COL}^j is failure probability of j^{th} column, and S_k denotes all subsets of k elements from $S = \{1, 2, 3, \ldots, N_{COL}\}$.

If IR drop is considered, the *required sigma* corresponds to M^1, which is the amount of local random variation that the *first* bitcell should be tolerant to satisfy the yield constraint. The *required sigma* considering IR drop is 5.06σ in our experiments, which is a little bit higher than the original value, which is 5.04σ. The new *required sigma* will replace the existing value in Eq.(2.8). Finally, V_{ddmin} will be changed since ΔV_{th} in Eq.(2.8) increases.

2.3.2 Consideration of Peripheral Circuit Variation

Process variation affects not only SRAM bitcell operation but also operation of peripheral circuit. Word line pulse, sense amplifier enable signal, precharge signal, and other control signals of SRAM are generated in peripheral circuit. Among those control signals, word line pulse is the signal directly related to the operation of SRAM bitcell since read and write operations proceed while the word line pulse stays in 'high' state. In other words, word line pulse length affects SRAM bitcell's read and write stability. If the word line pulse length changes, the bitcell margin changes. For example, write margin of bitcell for word line pulse length of 0.92ns increases by 0.04σ as the word line pulse length increases by 10% whereas it decreases by 0.03σ as the word line pulse length decreases by 10%.

Process variations on peripheral circuit and IR drop are independent each other, but their impacts on operation of SRAM bitcell are correlated. Consequently, they

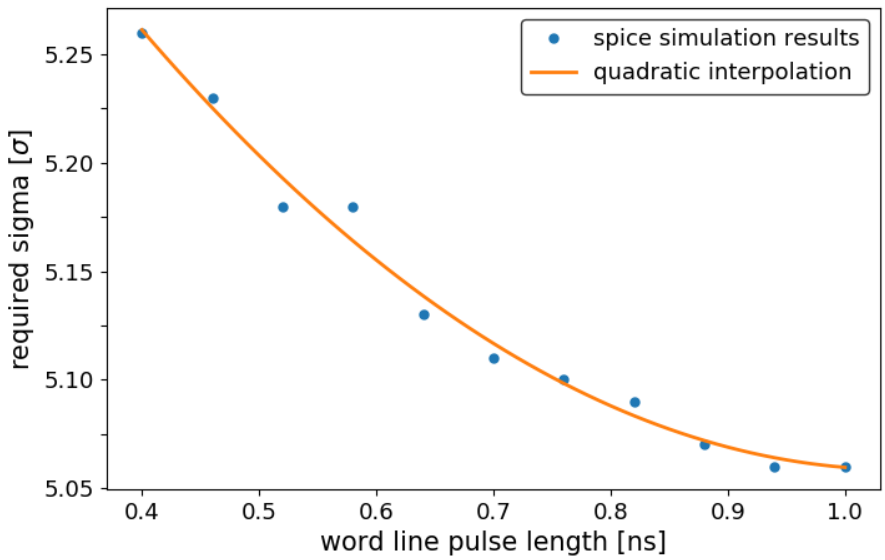


Figure 2.10: Required sigma increases as word line pulse length decreases.

should be considered together since both cause word line pulse length to be shorter, resulting in degradation of bitcell margin. We calculated the *required sigma* from Eqs.(2.9)∼(2.11) while varying the word line pulse length in spice simulation. The new *required sigma* values according to word line pulse length are shown in Fig. 2.10. *Required sigma* increases as word line pulse length decreases, because the decrease in bitcell margin caused by IR drop becomes bigger as the word line pulse length decreases.

Fig. 2.11 shows die count histogram according to the 3σ local worst word line pulse length for Vfail groups. Blue bars represent all dies in the groups, and orange bars represent dies with write failure. As shown in the figure, write failure does not show high correlation with word line pulse length because transistors in peripheral circuit and bitcells are affected by different global variations.

We modify the $\hat{V}_{ddmin}$ mapping in V_{fail}-V_{ddmin} correlation table to consider IR drop and peripheral circuit variation. The issue of non-consistent trend can be resolved

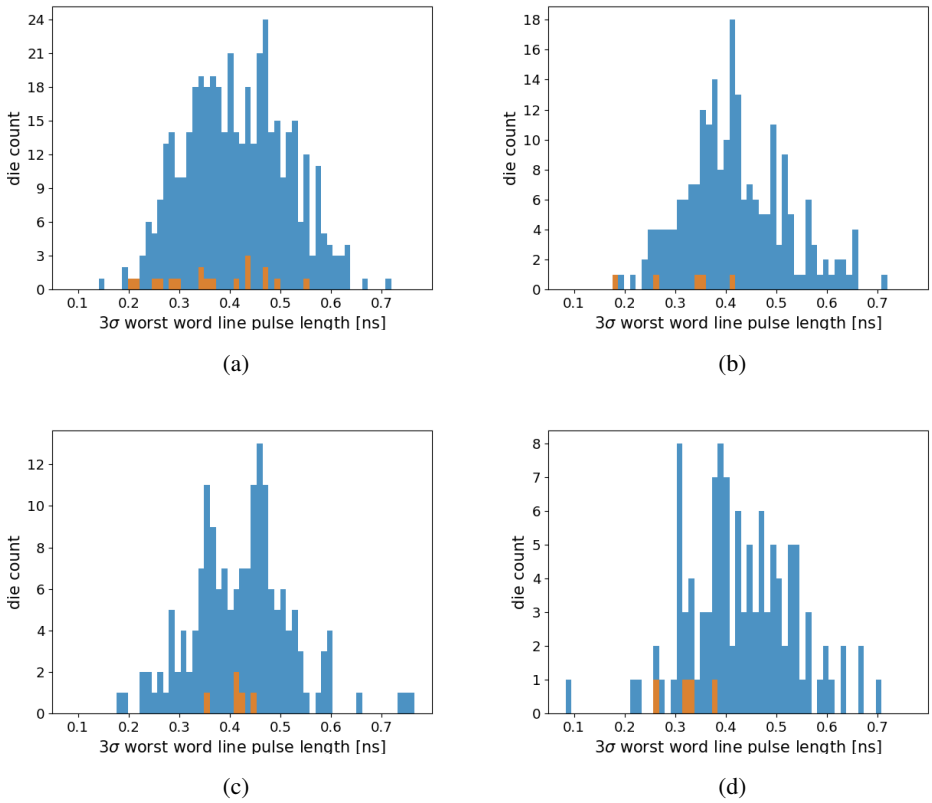


Figure 2.11: Histograms of all dies (blue) and dies with write failure (orange) according to word line pulse length. Each histogram is associated with V_{fail} group: (a) 0.56V, (b) 0.58V, (c) 0.60V, (d) 0.62V.

by modifying $\hat{V}_{ddmin}$ mapping because our proposed methodology decides $\hat{V}_{ddmin}$ statistically. To change the $\hat{V}_{ddmin}$, we simulated word line pulse in each die and replaced *required sigma* in Eq.(2.8) with the value from the interpolated curve in Fig. 2.10. Then, $\hat{V}_{ddmin}$ is recalculated statistically considering the newly derived V_{ddmin} of the dies.

2.3.3 V_{ddmin} Prediction including Access Failure Prohibition

Methodology presented in Sec. 2.2$\sim$ 2.3.2 estimates read and write V_{ddmin}. However, there is an additional issue of potential access failure if SRAM is designed for high performance on NTV regime. SRAM targeted to high performance will have a much small timing margin to achieve high speed read and write. Therefore, applying $\hat{V}_{ddmin}$ in V_{fail}-V_{ddmin} correlation table may cause access failure in which access time exceeds maximum tolerable time due to process variation. To resolve the issue of access failure, we need to increase $\hat{V}_{ddmin}$ of dies that are in danger of access failure.

Fig. 2.12 shows die count histogram according to 3σ local worst word line pulse length for V_{fail} groups. Blue bars represent all dies in the groups, and orange bars represent dies with access failure. As shown in the figure, dies with short word line pulse length are more vulnerable to access failure, and access failure shows high correlation with word line pulse length (L_{WL}). Based on the observation in Fig. 2.12, we reinforce our methodology to correct access failure by adjusting $\hat{V}_{ddmin}$ of dies whose estimated word line pulse length is shorter than pre-defined threshold value.

To retain the information of L_{WL} threshold value and adjusted $\hat{V}_{ddmin}$, we construct L_{WL}-V_{ddmin} correlation table as well as V_{fail}-V_{ddmin} correlation table in design infra development phase. Then, $\hat{V}_{ddmin}$ that prohibits read, write, and access failures can be selected directly from the tables in silicon production phase, as shown in Fig. 2.13.

Note that access failure does not occur in industry partner's 28nm SRAM design since it is optimized for 1.0V (super-threshold) operation and designed with sufficient

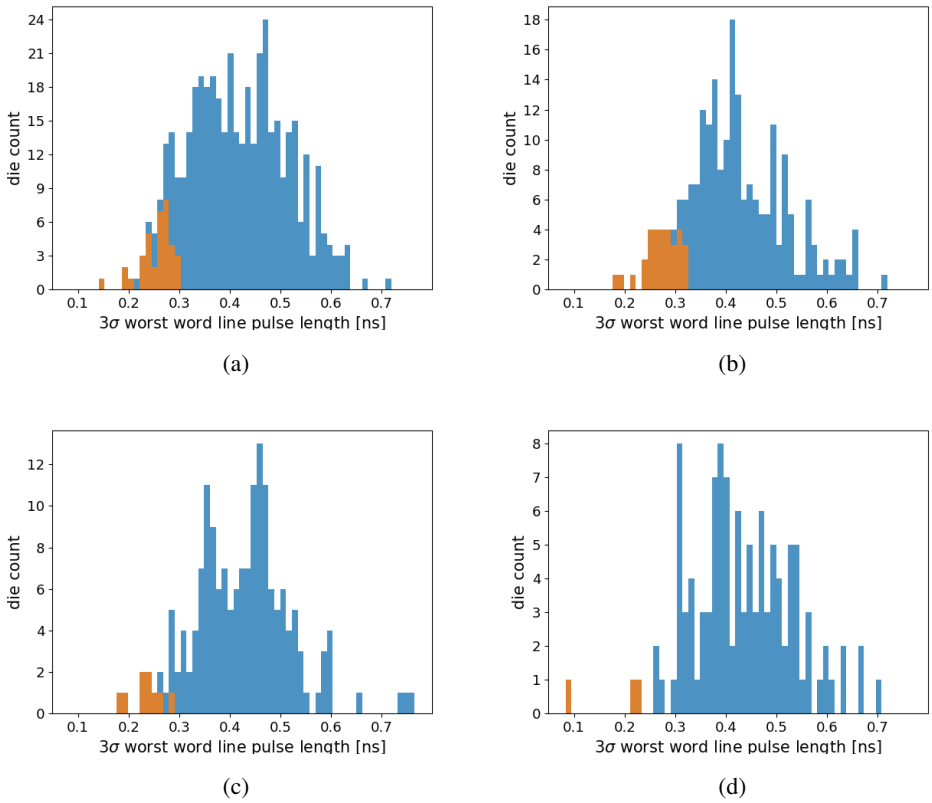
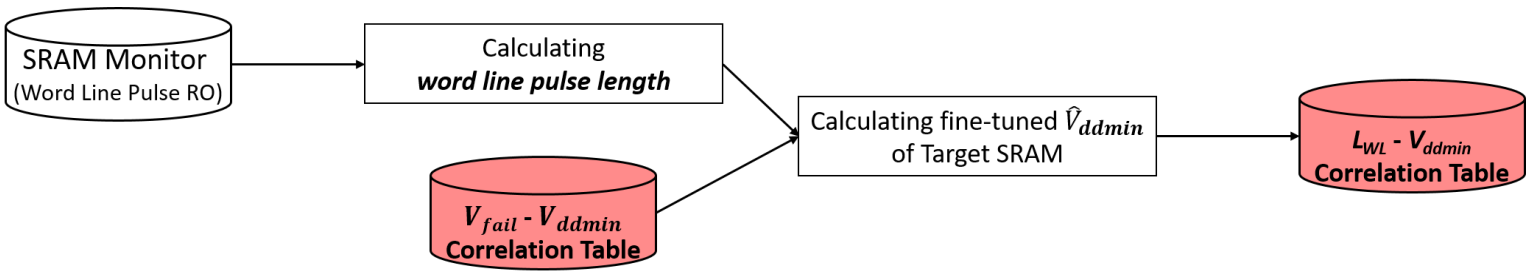
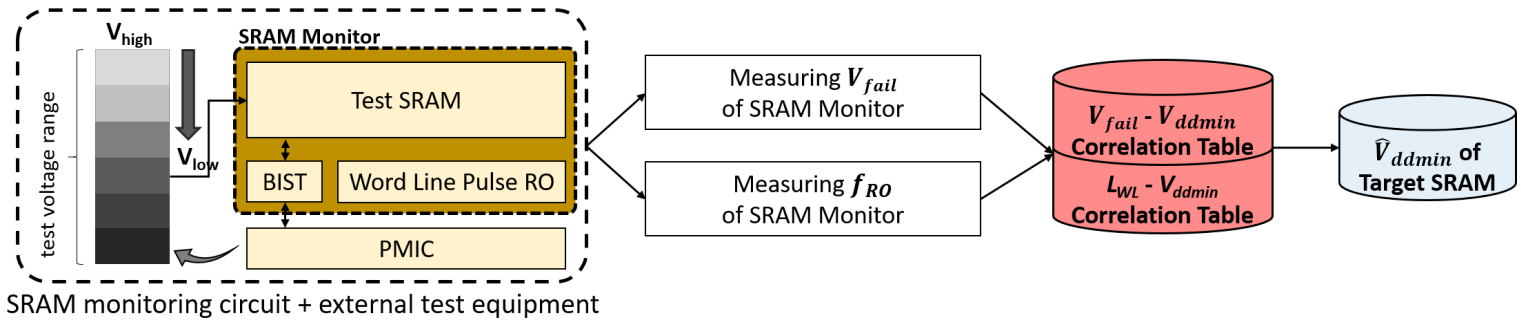


Figure 2.12: Histograms of all dies (blue) and dies with access failure (orange) according to word line pulse length. Each histogram is associated with V_{fail} group: (a) 0.56V, (b) 0.58V, (c) 0.60V, (d) 0.62V.



(a)



(b)

Figure 2.13: Extended flow of our proposed SRAM on-chip monitoring methodology to cope with access failure: (a) building-up L_{WL}-V_{ddmin} correlation table at design infra development phase, (b) deriving an SRAM $\hat{V}_{ddmin}$ on each die from V_{fail}-V_{ddmin} and L_{WL}-V_{ddmin} correlation tables at silicon production phase.

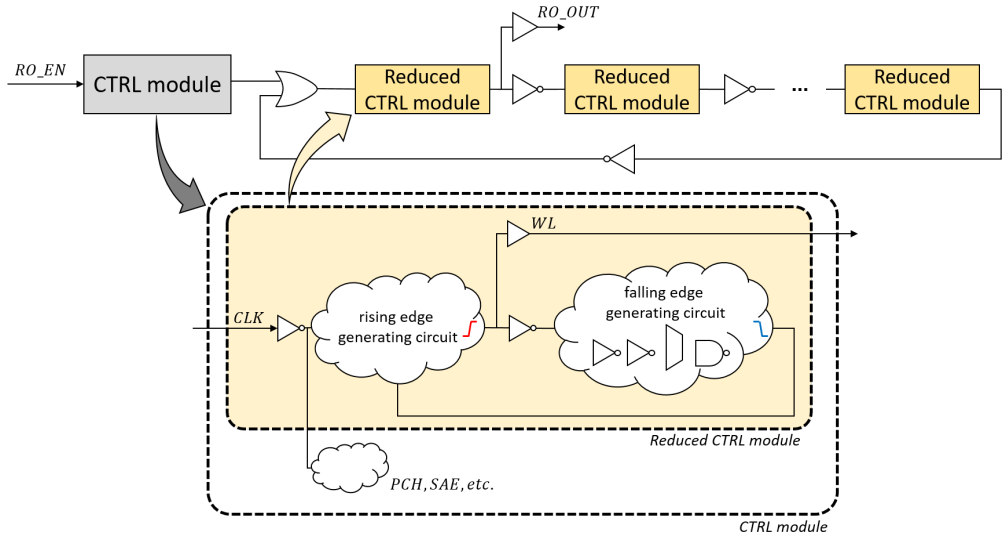


Figure 2.14: Ring oscillator for word line pulse length monitoring. Transistors on the path generating word line pulse from control module are extracted to build reduced control module.

timing margin. Assuming SRAMs aggressively optimized for high performance on NTV regime, we reduced the word line pulse length by 40% to simulate access failure in 0.6V. We confirmed, through industry partner, that this is valid assumption.

Calculating word line pulse length

We added a ring oscillator to SRAM monitor to estimate the word line pulse length of SRAM blocks on different dies. We extracted transistors on the path that generates word line pulse from control module to form a *reduced control module* as shown in Fig. 2.14. Then, we cascaded the *reduced control modules* to create a ring oscillator. Because the control module is triggered by clock signal and word line pulse includes both rising and falling edges, inserting a inverter between *reduced control modules* and connecting output of inverter to clock pin of *reduced control module* in next stage enable the circuit to oscillate.

To build word line pulse length estimation model, we firstly performed spice simulation on 100 dies while varying the global variation. From the simulation, we measured the frequency of word line pulse ring oscillator, f_{RO}, and the word line pulse length generated from control module, L_{WL}. Then, we used quadratic interpolation to draw relation between f_{RO} and L_{WL}. The spice simulation and interpolation results are shown in Fig. 2.15(a).

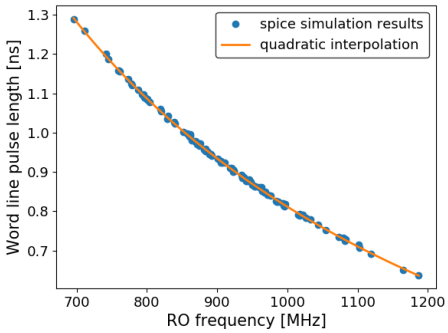
Then, we measured f_{RO} and L_{WL} from additional 1000 dies and estimated 3σ local worst L_{WL} from f_{RO} using the interpolation figured out in Fig. 2.15(a). Fig. 2.15(b) shows the estimation results when global variation alone is considered. The x and y axes are target L_{WL} and estimated L_{WL}, respectively. Estimation results show 0.97% of maximum error rate. Fig. 2.15(c) shows the estimation results when local variation in ring oscillator is considered. Since noise caused by local variation is injected to measurement, estimation results are degraded to 9.39%. Therefore, we introduced additional margin to guarantee pessimistic estimation for 99.9% yield. We calculated the change in bitcell margin according to the change in word line pulse length, and decided the margin to -30ps. The final L_{WL} estimation follows Eq.(2.12)

$$L_{WL\_3\sigma} = f(f_{RO}) + g(f_{RO}) + L_{margin} \tag{2.12}$$

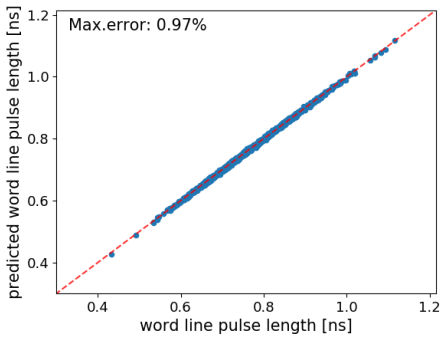
where $f(\cdot)$ is quadratic interpolation function in Fig. 2.15(a), $g(\cdot)$ is mapping function between nominal L_{WL} and 3σ local worst L_{WL}, and L_{margin} is the margin to guarantee pessimism. $g(\cdot)$ can be derived in a similar process of deriving $f(\cdot)$. We observed nominal and 3σ local worst L_{WL} in SS, TT, FF corners and built mapping function $g(\cdot)$. Note that $g(\cdot)$ depends on the estimation target. For example, if estimation target is 2σ local worst L_{WL} rather than 3σ, $g(\cdot)$ should be derived again according to the estimation target.

Calculating fine tuned V_{ddmin} of target SRAM

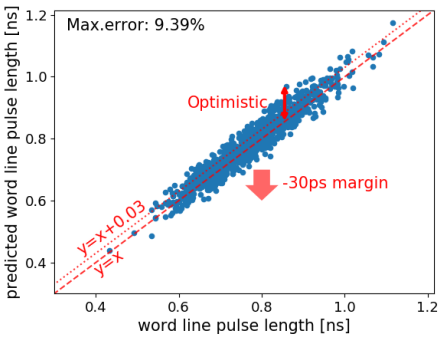
L_{WL}-V_{ddmin} correlation table contains information of L_{WL} threshold and V_{ddmin} adjustment value. $\hat{V}_{ddmin}$ of dies with estimated L_{WL} shorter than L_{WL} threshold is



(a)

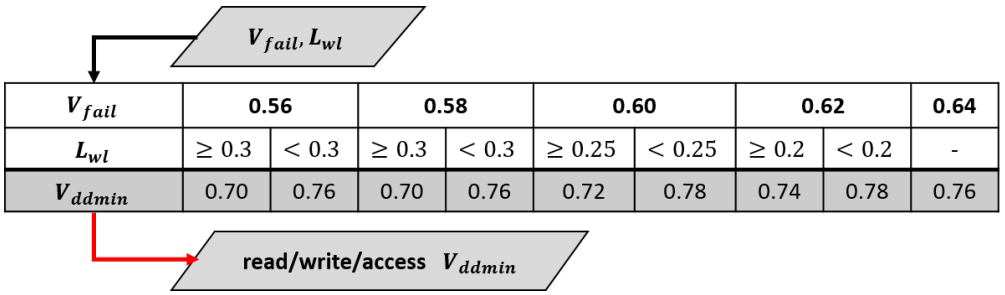


(b)



(c)

Figure 2.15: (a) Quadratic interpolation between 100 spice simulation results of ring oscillator frequency and word line pulse length. (b, c) 3σ local worst word line pulse length prediction results: (b) considering global variation only, and (c) considering local random variation induced noise in ring oscillator measurement.



V_{fail}	0.56		0.58		0.60		0.62		0.64
L_{wl}	≥ 0.3	< 0.3	≥ 0.3	< 0.3	≥ 0.25	< 0.25	≥ 0.2	< 0.2	-
V_{ddmin}	0.70	0.76	0.70	0.76	0.72	0.78	0.74	0.78	0.76

Figure 2.16: An illustration of L_{WL}-V_{ddmin} correlation table that is added to V_{fail}-V_{ddmin} correlation table.

adjusted to prohibit access failure.

In logic delay, 3σ local worst delay is commonly considered for timing closure. However, for SRAM, it is too pessimistic to consider 3σ local variation both for peripheral circuit and for bitcell since local variation in peripheral circuit and bitcell are independent to each other. Thus, we calculated bitcell margin while considering 0 (nominal), 1, 2, and 3σ local worst variation in peripheral circuit. Then, total yield is computed considering the probability of each occurrence.

Algorithm 1 describes how to calculate read, write, and access $\hat{V}_{ddmin}$. The algorithm first builds a set of all possible $\boldsymbol{l}$, $\boldsymbol{v}$ pairs in which each is a combination of elements of L_{WL_TH} and V_{step}, and the size of $\boldsymbol{l}$ and $\boldsymbol{v}$ is the number of V_{fail} groups in T_{old}(line 1). Then, 0$\sim$3σ local variation induced word line pulse length of each die is estimated from the f_{RO} of SRAM monitor (line 2). Notation k in the algorithm means it retains information of 0$\sim$3σ local variation induced values. For example, $L_{WL\_k\sigma}$ denotes 0$\sim$3σ local worst word line pulse length of all dies. Next, all the $\boldsymbol{l}$, $\boldsymbol{v}$ pairs in C are explored to find feasible pairs that meet 99.9% yield (lines 4$\sim$14). During iteration, $\hat{V}_{ddmin}$ of dies are adjusted based on the estimated L_{WL}, L_{WL} threshold($\boldsymbol{l_c}$), and $\hat{V}_{ddmin}$ adjustment step ($\boldsymbol{v_c}$) (line 5). We compared the estimated values with real values of 3σ local worst word line pulse length to identify dies whose $\hat{V}_{ddmin}$ will be adjusted. Then, access margin is calculated with the adjusted $\hat{V}_{ddmin}$(line 6). The access margin is calculated by modifying ADM flow, measuring access time rather than the current of access transistors. Yield of dies in each of $k\sigma$ peripheral variation groups are calculated by replacing M with $M_{k\sigma}$ in Eqs.(2.2)$\sim$(2.5) (line 7). Then, total yield considering the probability of $k\sigma$ local variation in peripheral circuit is computed as follows (line 8):

$$Yield = \prod_{k \in K} Y_{k\sigma} \cdot \frac{P_{k\sigma}}{\sum_{i \in K} P_{i\sigma}} \tag{2.13}$$

$$P_{k\sigma} = \begin{cases} cdf(k), & \text{if } k = 0 \\ cdf(k) - cdf(k-1), & \text{otherwise} \end{cases} \tag{2.14}$$

Algorithm 1: read/write/access $\hat{V}_{ddmin}$ calculation

input : V_{fail}-V_{ddmin} correlation table: T_{old}

V_{fail} of each dies: V_{fail}_list

f_{RO} of each dies: f_{RO}_list

L_{WL} thresholds: $L_{WL\_TH} = \{l_1, l_2, ..., l_M\}$

$\hat{V}_{ddmin}$ adjustment steps: $V_{step} = \{v_1, v_2, ..., v_N\}$

output: New V_{fail}-V_{ddmin} correlation table: T_{new}

$C \leftarrow$ every ($\boldsymbol{l}$, $\boldsymbol{v}$) of size $N_{V_{fail}\_groups}$

$L_{WL\_k\sigma} \leftarrow calculate\_L_{WL\_k\sigma}(f_{RO}\_list)$

$S \leftarrow \{\}$

while $!explored\_all(C)$ **do**

`// ` $\boldsymbol{l_c}$`, ` $\boldsymbol{v_c}$`: selected ` $\boldsymbol{l}$`, ` $\boldsymbol{v}$` in current iteration`

$V_{dd} \leftarrow assign\_V_{dd}(T_{old}, V_{fail}\_list, L_{WL\_3\sigma}, \boldsymbol{l_c}, \boldsymbol{v_c})$

$M_{k\sigma} \leftarrow calculate\_access\_margin(V_{dd})$

$Y_{k\sigma} \leftarrow calculate\_yield(M_{k\sigma})$

$Y \leftarrow calculate\_total\_yield(Y_{k\sigma})$

if $Y \geq 0.999$ **then**

$S \leftarrow S \cup (\boldsymbol{l_c}, \boldsymbol{v_c})$

else

$C \leftarrow C - child\_set(\boldsymbol{l_c}, \boldsymbol{v_c})$

end

end

$\boldsymbol{l_{min}}, \boldsymbol{v_{min}} \leftarrow select\_min\_power(S)$

$T_{new} \leftarrow build\_up\_table(T_{old}, \boldsymbol{l_{min}}, \boldsymbol{v_{min}})$

where $K = \{0, 1, 2, 3\}$. Exploring every pair of $\boldsymbol{l}$ and $\boldsymbol{v}$ in C is time consuming because there are 4^{10} pairs for M=4, N=4 for 5 V_{fail} groups. This exhaustive exploring space is reduced by branch and cut method (line 12). $\boldsymbol{l}$, $\boldsymbol{v}$ pairs that are not expected to satisfy 99.9% yield are excluded from the search space beforehand. For example, suppose the yield constraint is not satisfied for a certain $\boldsymbol{l_c}$ and $\boldsymbol{v_c}$. Then, it is clear that $\boldsymbol{l}$, $\boldsymbol{v}$ pairs whose elements are smaller than or equal to $\boldsymbol{l_c}$, $\boldsymbol{v_c}$ pair will not satisfy yield constraint. This is because smaller elements mean the L_{WL} threshold or $\hat{V}_{ddmin}$ adjustment step becomes smaller, which results in reducing the number of dies whose $\hat{V}_{ddmin}$ will be adjusted or reducing the $\hat{V}_{ddmin}$ adjustment step size. Both of them decrease the yield. As a result, $\boldsymbol{l}$, $\boldsymbol{v}$ pairs worse than $\boldsymbol{l_c}$, $\boldsymbol{v_c}$ pair in terms of yield can be excluded from C, reducing the size of search space. Among the feasible pairs, the L_{WL} threshold and corresponding $\hat{V}_{ddmin}$ adjustment step with minimum power consumption is selected (line 15). Finally, the new V_{fail}-V_{ddmin} correlation table merged with L_{WL}-V_{ddmin} correlation table is built up (line 16).

We explored the L_{WL} threshold from 0.20ns to 0.35ns with 0.05ns step interval, and $\hat{V}_{ddmin}$ adjustment step from 20mV to 80mV with 20mV step interval. The final V_{fail}-V_{ddmin} correlation table merged with L_{WL}-V_{ddmin} correlation table that built up from the algorithm is shown in Fig. 2.16.

2.4 Experimental Results

To validate our proposed approach, we used industry partner's 28nm PDK and one of their bitcell designs. We used Synopsys Hspice to do spice simulation in our flow, and FineSim to calculate power consumption in industry partner's memory block. For SRAM monitor and target SRAM, we used 16KB SRAM and modified SRAM blocks in OpenSPARC T1, respectively and analyzed 1000 dies to gather V_{ddmin} data, in which the SRAM monitor was tested by varying the supply voltage from 0.56V to 0.64V with a step size of 20mV.

2.4.1 $\hat{V}_{ddmin}$ Considering Read and Write Failures

Since the read operation on bitcells was stable at NTV regime, the read V_{fail} was not detected. This is due to the design of bitcells that is inherently less vulnerable for the read operation in low voltage than for the write operation. Thus, we collected a set of experimental results regarding the write operation. (Note that our proposed V_{ddmin} prediction flows for testing read stability as well as for testing write stability are identical except the extraction of ADM and WRM, which means a read stable V_{ddmin} will be collected if there is a bitcell unstable at NTV regime.)

Fig. 2.17 shows the results of $\hat{V}_{ddmin}$ calculation for 1000 dies, arranged according to the V_{fail} values(blue solid lines) computed at design phase by varying the global corners in Hspice simulation. The orange dotted lines indicate the V_{ddmin} values, corresponding to the V_{fail} values and global corners. The red lines represent the $\hat{V}_{ddmin}$ values taken from the V_{fail}-V_{ddmin} correlation table. The $\hat{V}_{ddmin}$ ensures 99.9% of SRAM non-failure probability for dies with V_{fail} values in the production phase. For example, for dies with 0.56V of V_{fail}, 0.68V can be applied to the dies for 99.9% SRAM non-failure probability. On the other hand, the gray dotted line and black horizontal line represent the V_{ddmin} values computed by the conventional flow based on [31, 32], and its $\hat{V}_{ddmin}$ (= 0.74V) satisfying 99.9% of SRAM non-failure probability. Conventional flow from [31, 32] is widely used in industry while designing a bitcell to estimate the stability of bitcell and decide its operating voltage. Since worst case should be considered without our methodology, 0.74V of $\hat{V}_{ddmin}$ should be applied uniformly to SRAM blocks in all dies. Note that 0.74V of $\hat{V}_{ddmin}$ with 0.60V peripheral voltage already reduced leakage power, read energy, and write energy by 70.0%, 50.17%, and 50.47% in average with performance degradation(x5.62 slower) compared to applying nominal voltage(1.0V).

Purple lines represent the $\hat{V}_{ddmin}$ when considering IR drop and peripheral circuit variation. $\hat{V}_{ddmin}$ of dies with 0.56V of V_{fail} is adjusted 20mV higher to meet 99.9% SRAM non-failure probability.

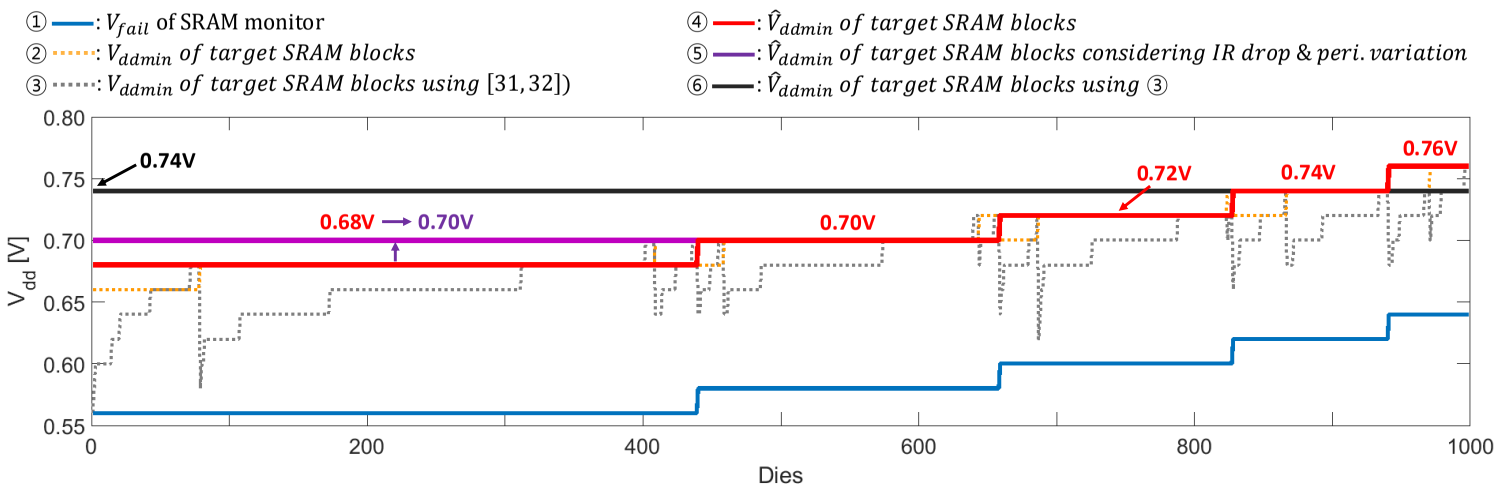


Figure 2.17: Comparison of the values of V_{ddmin} (② orange dotted lines) and $\hat{V}_{ddmin}$ (④ red lines and ⑤ purple line) computed by our prediction flow for 1000 dies for 99.9% yield constraint with the values of V_{ddmin} (③ gray dotted lines) and $\hat{V}_{ddmin}$ (⑥ black line) computed by the conventional flow using [31, 32].

Table 2.4: Dies and $\hat{V}_{ddmin}$ distributions by V_{fail}

V_{fail}[V]	#Dies	$\hat{V}_{ddmin}$[1]	$\hat{V}_{ddmin}$[2]
0.56	439	0.68	0.70
0.58	219	0.70	0.70
0.60	169	0.72	0.72
0.62	114	0.74	0.74
0.64	59	0.76	0.76

[1] IR drop and peripheral variation are not considered.

[2] IR drop and peripheral variation are considered.

Table 2.5: Savings on leakage power, read energy, and write energy of SRAM bitcell array over those by the conventional flow [31, 32] for read/write operation.

$\hat{V}_{ddmin}$	Δpower/energy	
read/write	leakage power	-10.45%
	read energy	-4.99%
	write energy	-5.45%

The dies and $\hat{V}_{ddmin}$ distribution based on V_{fail} values are summarized in Table 2.4. Power consumption compared to 0.74V of $\hat{V}_{ddmin}$ is summarized in Table 2.5. Leakage power, dynamic read energy, and dynamic write energy of bitcell array are reduced by 10.45%, 4.99%, and 5.45%, respectively.

2.4.2 $\hat{V}_{ddmin}$ Considering Read/Write and Access Failures

V_{fail}-V_{ddmin} correlation table merged with L_{WL}-V_{ddmin} correlation table is summarized in Table 2.6. $\hat{V}_{ddmin}$ of target SRAMs whose estimated word line pulse length shorter than the threshold value are adjusted. We explored the yield and power consumption while varying L_{WL} threshold from 0.20ns to 0.35ns with step interval of 0.05ns, and $\hat{V}_{ddmin}$ adjustment step from 20mV to 80mV with step interval of 20mV, respectively as explained in Sec. 2.3.3. Then, $\hat{V}_{ddmin}$ adjustment result showing the minimum power consumption while satisfying yield constraint is selected. As a result, $\hat{V}_{ddmin}$ is adjusted 60mV to 80mV higher than read/write $\hat{V}_{ddmin}$. For example, $\hat{V}_{ddmin}$ of target SRAM whose V_{fail} is 0.60V and word line pulse length is shorter than 0.20ns is adjusted to 0.80V. For dies whose V_{fail} is 0.64V, there is no $\hat{V}_{ddmin}$ adjustment because no access failure observed on that group. Note that some of $\hat{V}_{ddmin}$ values which have bigger L_{WL} values than L_{WL} threshold in Table 2.6 are different from those in Table 2.4. This is because word line pulse length is reduced by 40% to simulate access failure in 0.6V, as mentioned in Sec. 2.3.3. Unified $\hat{V}_{ddmin}$ for all dies is increased to 0.76V to prohibit access failure. Power and energy consumption of bitcell array compared to 0.76V of $\hat{V}_{ddmin}$ are summarized in Table 2.7. Leakage power, dynamic read, and write energy are reduced by 13.90%, 6.63%, and 6.60%, respectively.

2.4.3 Observation for Practical Use

Here, we discuss two potential issues of proposed methodology on practical use and their resolution ideas. First, our methodology takes rather long computation time (a few weeks) to build up the final V_{fail}-V_{ddmin} correlation table due to run lots of Monte

Table 2.6: Dies and $\hat{V}_{ddmin}$ distributions by V_{fail} and L_{WL}

V_{fail} [V]	L_{WL} threshold [ns]	#Dies	$\hat{V}_{ddmin}$
0.56	≥0.25	406	0.70
	<0.25	33	0.78
0.58	≥0.25	207	0.72
	<0.25	12	0.78
0.60	≥0.20	166	0.74
	<0.20	3	0.80
0.62	≥0.20	112	0.74
	<0.20	2	0.80
0.64	-	59	0.76

Table 2.7: Savings on leakage power, read energy, and write energy of SRAM bitcell array over those by the conventional flow [31, 32] for read/write/access operation.

$\hat{V}_{ddmin}$	Δpower/energy	
read/write/access	leakage power	-13.90%
	read energy	-6.63%
	write energy	-6.60%

Carlo simulation of SRAM monitor and an algorithm that collects data from all the analyzed dies. However, it will not be an issue because the whole process runs once at design infra development phase. Second, there would exist measurement overhead at silicon production phase because measuring V_{fail} of an SRAM monitor requires sweeping the supply voltage of bitcell array. However, the overhead of V_{fail} measurement can be reduced by using dual-rail voltage scheme [38] or measuring multiple SRAM monitors on different dies simultaneously.

Chapter 3

Allocation of Always-On State Retention Storage for Power Gated Circuits - Steady State Driven Approach

3.1 Motivations and Analysis

3.1.1 Impact of Self-loop on Power Gating

Figs. 3.1(a) and (b) show a section of Verilog code which commonly appears in RTL description of design behavior and the corresponding synthesized structure, respectively. Flip-flops in Fig. 3.1(b) contain combinational mux-feedback loops. In our presentation, we call such flip-flops *self-loop* FFs and the rest *ordinary* FFs.

Observation 1: *How much do the self-loop FFs* **negatively** *influence reducing state retention storage, thereby leakage power, in power gating?* Note that we should *replace every self-loop FF with a distinct retention flip-flop with at least one bit storage* for state retention since we have no idea whether the flip-flop state, when waking up, comes from the self-loop or the driving flip-flops other than itself (e.g., the red signal flow in Fig. 3.1(b)). In addition, even if we know where the state comes from, it is impossible to restore the state without retention storage when the state comes from the self-loop. For example, Fig. 3.2(b) and Fig. 3.2(c) show the retention storage allocation in the presence and absence of self-loop on flip-flop f_2 in a small flip-flop

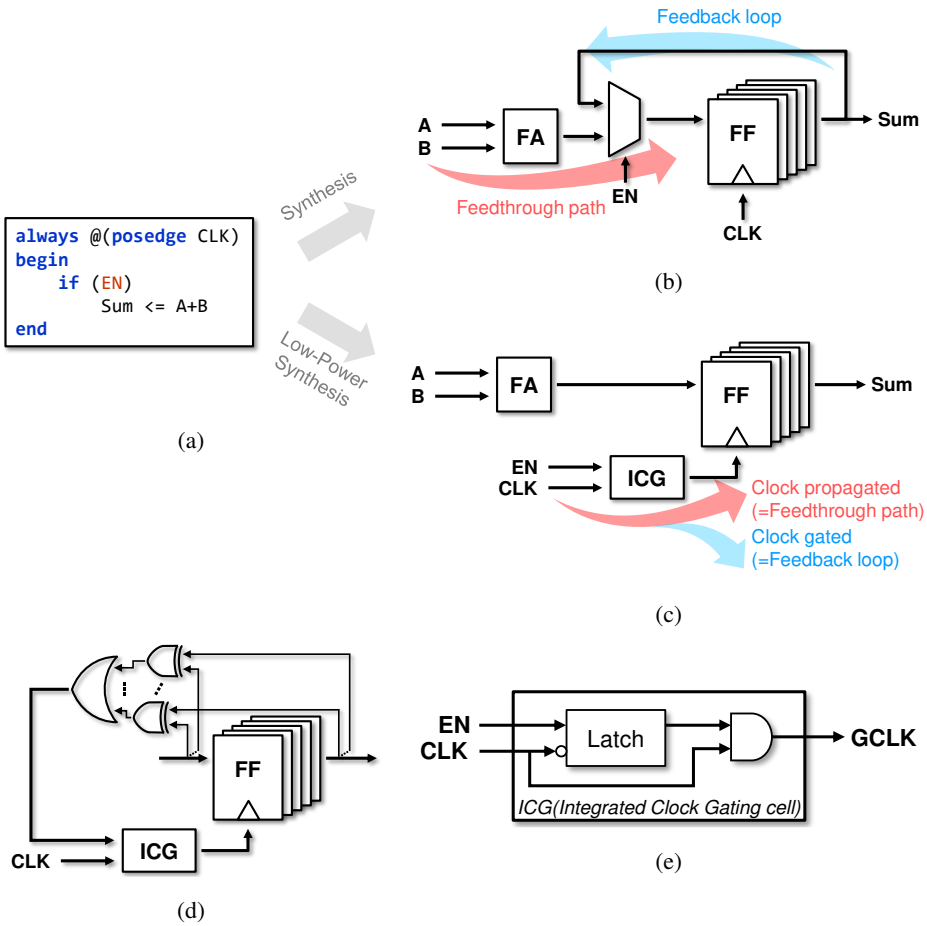
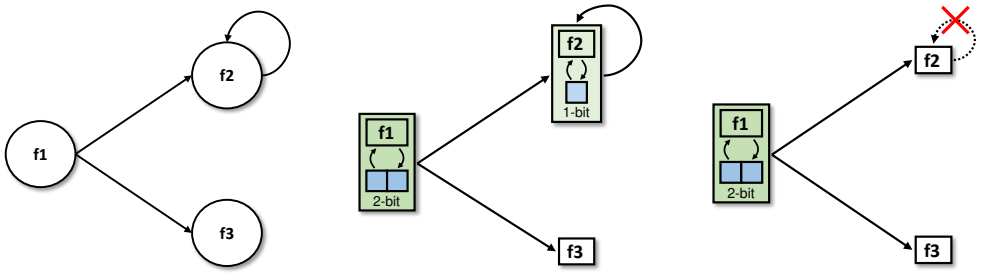


Figure 3.1: (a) An HDL verilog description. (b) The flip-flops with mux-feedback loop synthesized for the code in (a). (c) The logic structure for (b) supporting idle logic driven clock gating. (d) The logic structure supporting data toggling driven clock gating. (e) The structure of ICG(Integrated Clock Gating cell).



(a) Self-loop on f_2 (b) Allocating 3 bits (c) Allocating 2 bits

Figure 3.2: (a) Flip-flop dependency graph of circuit containing three FFs *with* one self-loop FF. (b) Minimal allocation of retention storage for (a). (c) Minimal allocation of retention storage for (a), assuming the self-loop FF as a FF with no self-loop.

dependency graph in Fig. 3.2(a), respectively. It is reported that even though multi-bit retention storage can be aggressively utilized to maximally reduce the state retention storage, the saving amount is not expected to be more than 3.15% due to the presence of self-loop FFs in circuits [25].

Observation 2: *How much do the self-loop FFs* **positively** *help clock gating save dynamic power?* While the self-loop FFs adversely affect the minimization of state retention storage, it is very useful in clock gating since it requires nearly no clock gating overhead. For example, Fig. 3.1(c) shows the clock gated circuit directly transformed from that in Fig. 3.1(b), from which we can see that the gated logic completely removes the multiplexers while allocating just one ICG (integrated clock gating) block. This style of clock gating is called *idle logic driven clock gating*. Designers in industry make use of this style of clock gating to save dynamic power as much as possible by intentionally writing code like that shown in Fig. 3.1(a). To add up more power saving, the *data toggling based clock gating* is also used as shown in Fig. 3.1(d) by allocating the XOR gates to check if the flip-flop states are unchanged or not.[1]

Observation 3: *How many self-loop flip-flops do the circuits contain?* Table 3.1 sum-

[1] In Sec. 3.2.3, we show a way of sharing those XORs in clock gating with our state monitoring logic in power gating.

marizes the number of self-loop FFs in the circuits synthesized from IWLS2005 benchmark [43] and OpenCores [44] code. It is shown that the self-loop FFs occupy 56%∼99% (82.71% on average) among all flip-flops in circuits. Based on observations 1 and 2, prior works have been in a dilemma in minimizing retention storage in power gating due to the abundance of self-loop FFs. This work breaks this inherent bottleneck in power gating and never takes away the benefit reaped from clock gating at the same time.

Table 3.1: The number of self-loop FFs in circuits from IWLS2005 benchmarks and OpenCores.

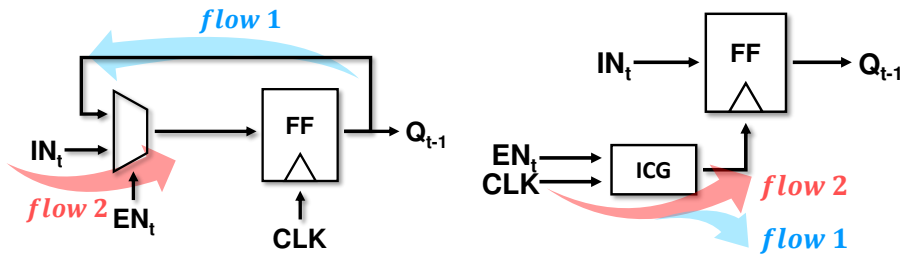
Designs	# of FFs	# of self-loop FFs	% of self-loop FFs
SPI	229	195	85.15%
AES_CORE	530	296	55.85%
WB_CONMAX	770	610	79.22%
MEM_CTRL	1563	1319	84.39%
AC97_CTRL	2199	1705	77.54%
WB_DMA	3109	2878	92.57%
PCI	3220	2829	87.86%
VGA_LCD	17050	16892	99.07%
Avg.	-	-	82.71%

3.1.2 Circuit Behavior Before Sleeping

The signal flow path to Q_t at clock time t on a self-loop FF in Fig. 3.3(a) is one of the two signal flows, depending on EN value at t:

- *flow 1*: $Q_{t-1} \rightarrow MUX \rightarrow FF$
- *flow 2*: $IN_t \rightarrow MUX \rightarrow FF$

Consequently, if it is certain that the value of IN_t at cycle time t and the value of Q_{t-1} at time $t-1$ are identical, we can disregard the role of mux-feedback loop in Fig. 3.3.



(a) (b)

Figure 3.3: Two signal flow paths to Q_t at cycle time t in the self-loop FFs, which are implemented with (a) mux-feedback loop and (b) idle logic driven clock gating.

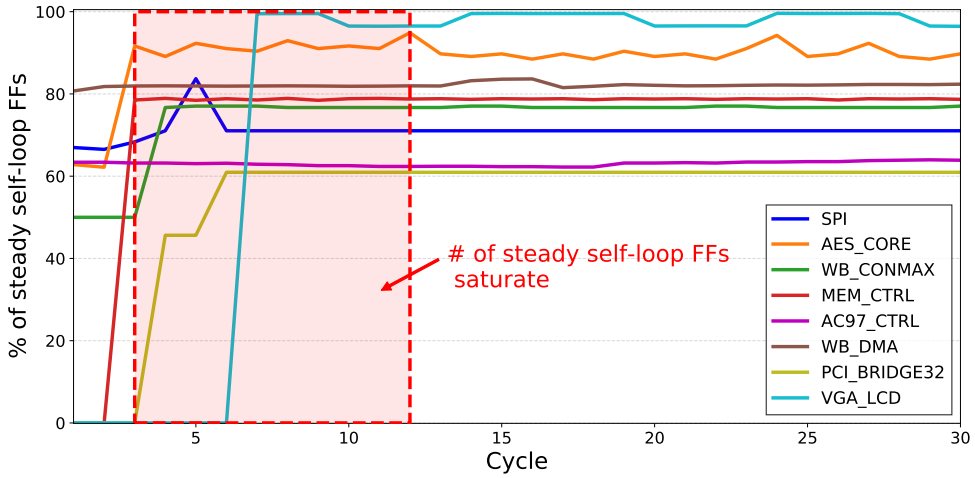


Figure 3.4: The changes of the portion of steady self-loop FFs in simulation as the circuits gracefully move to sleep mode.

We formally state this condition:

Self-loop removal condition: *The self-loop signal flow (i.e., flow 1) at cycle time* t *in a self-loop FF (e.g., Fig. 3.3) can be safely disregarded if it satisfies*

$$IN_t = Q_{t-1}. \tag{3.1}$$

Thus, the more the number of self-loop FFs is satisfying the condition of *Eq.3.1* at a certain cycle time t in a circuit, the higher the reduction of state retention storage is in power gating the circuit. The circuit simulation results in Fig. 3.4 support the feasibility of significantly reducing the amount of state retention storage in association with the self-loop FFs. From the gate level simulation, we observed the states of self-loop FFs at the moment the circuits are expected to be power gated. Precisely, Fig. 3.4 shows the changes of the portion of self-loop FFs in steady state (i.e., meeting *Eq.3.1*) as circuits gracefully go down to sleep mode while maintaining the steady primary inputs to the circuits for up to 30 clock cycles, for one of the repeated power gating simulations. It shows that over 60% among self-loop FFs in all circuits are in stable state during the grace period when the circuits are about to make a transition to sleep mode.

3.1.3 Wakeup Latency vs. Retention Storage

It is clear that a long wakeup delay enables to provide an increased opportunity of reducing total size of retention storage at the expense of circuit performance. However, the saving of total retention storage size and total number of retention FFs start to saturate when the wakeup latency l exceeds 2 or 3[2], as shown in Fig. 3.5. (We ran the allocation method in [24] to all benchmark circuits, assuming every self-loop FF as an ordinary one with no self-loop, and averaged the saving numbers.)

[2]Our experiments set the wakeup latency $l = 2$ as well as 3

Table 3.2: Changes of the number of steady self-loop flip-flops as γ changes.

Designs	# of simulations	γ					
		0	0.01	0.02	0.03	0.04	0.05
SPI	1791	157 (80.51%)	159 (81.54%)	159 (81.54%)	159 (81.54%)	159 (81.54%)	159 (81.54%)
AES_CORE	512	130 (43.92%)	130 (43.92%)	130 (43.92%)	130 (43.92%)	130 (43.92%)	130 (43.92%)
WB_CONMAX	1086	354 (58.03%)	354 (58.03%)	354 (58.03%)	354 (58.03%)	354 (58.03%)	354 (58.03%)
MEM_CTRL	12856	753 (57.09%)	765 (58.00%)	768 (58.23%)	777 (58.91%)	809 (61.33%)	857 (64.97%)
AC97_CTRL	168	152 (8.91%)	157 (9.21%)	163 (9.56%)	170 (9.97%)	171 (10.03%)	171 (10.03%)
WB_DMA	17776	1512 (52.54%)	1577 (54.79%)	1580 (54.90%)	1585 (55.07%)	1586 (55.11%)	1589 (55.21%)
PCI_BRIDGE32	376	599 (21.17%)	616 (21.77%)	619 (21.88%)	627 (22.16%)	645 (22.80%)	652 (23.05%)
VGA_LCD	228	4499 (26.63%)	4530 (26.82%)	4663 (27.60%)	4682 (27.72%)	4700 (27.82%)	4705 (27.85%)
Avg.	-	- (43.6%)	- (44.26%)	- (44.46%)	- (44.67%)	- (45.07%)	- (45.58%)

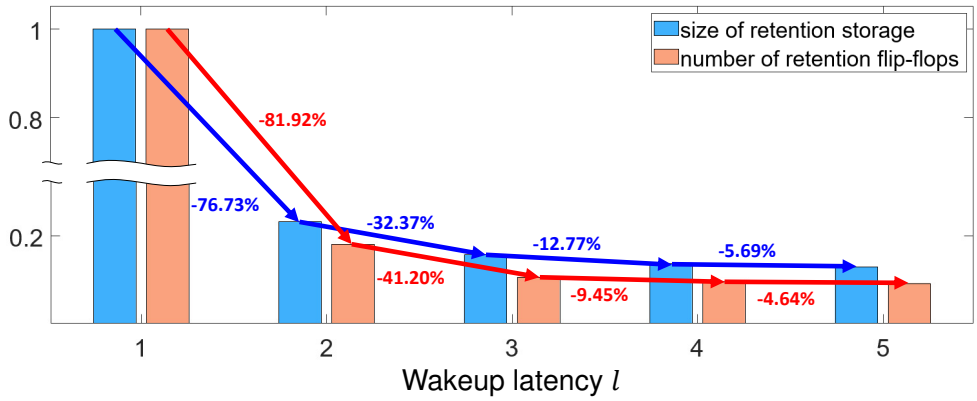


Figure 3.5: The normalized saving of total retention storage size and total number of retention FFs for wakeup latency l set to 1, 2, 3, 4, and 5, which shows that $l = 2$ or 3 suffices.

3.2 Steady State Driven Retention Storage Allocation

Our proposed steady state driven retention storage allocation, which is also summarized in Fig. 3.6, is composed of three steps:

(**Step 1**) *Extracting self-loop FFs* that are highly likely to be in steady state during the grace time period for circuit moving to sleep mode.

(**Step 2**) *Applying the conventional non-uniform MBRFF allocation* with $l = 2$ or 3 to the circuit produced by removing the self-loop from the FFs obtained in Step 1 to minimize the leakage power dissipation caused by the always-on state retention storage.

(**Step 3**) *Designing and optimizing the state monitoring logic for the self-loop flip-flops that do not need retention storage* according to the result of Step 2, fully utilizing the existing logic that supports clock gating to lighten the monitoring logic.

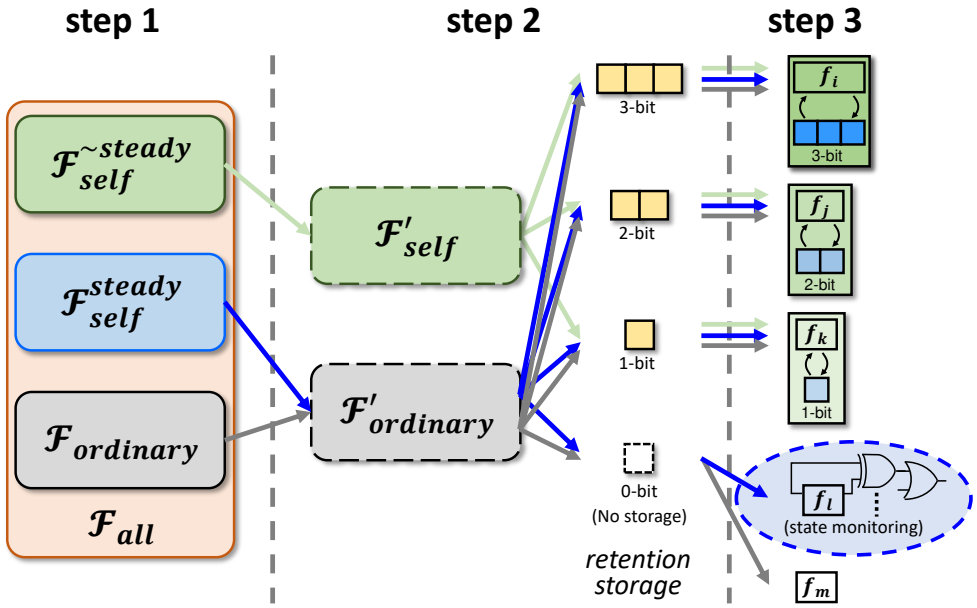


Figure 3.6: Classification and deployment of retention bits on flip-flops in the three steps of our strategy of retention storage allocation with $l = 3$.

3.2.1 Extracting Steady State Self-loop FFs

For an input circuit $\mathcal{C}$, let $\mathcal{F}_{all}$ and $\mathcal{F}_{self}$ be the sets of all flip-flops in $\mathcal{C}$ and all self-loop FFs in $\mathcal{C}$, respectively. Then, we perform a gate-level simulation on $\mathcal{C}$ while maintaining stable primary inputs and compute the data toggling probability, $prob(f_i)$, of every flip-flip f_i in $\mathcal{F}_{self}$, from which we extract a set, $\mathcal{F}^{steady}_{self}$, of self-loop FFs satisfying $prob(\cdot) \leq \gamma$ where γ is a user defined parameter.Thus, this step partitions $\mathcal{F}_{all}$ into $\mathcal{F}_{ordinary}$ (= $\mathcal{F}_{all}$ - $\mathcal{F}_{self}$), $\mathcal{F}^{steady}_{self}$, and $\mathcal{F}^{\sim steady}_{self}$ (= $\mathcal{F}_{self}$ - $\mathcal{F}^{steady}_{self}$) as shown in Step 1 of Fig. 3.6.

Determination of γ value: In our gate level simulation, we assume that the circuit needs a few clock cycles (e.g. 15 cycles in Fig. 3.4) before entering sleep mode after when a pre-defined sequence of input vectors is applied. Thus, we keep the last input vector steady for the clock cycles, and check, for each self-loop flip-flop, if it satisfies the self-loop removal condition in *Eq.1*. We perform this simulation 168∼17,776

Table 3.3: Changes of $prob_f$ as γ changes.

Designs	γ					
	0	0.01	0.02	0.03	0.04	0.05
SPI	0.0000	0.0034	0.0034	0.0034	0.0034	0.0034
AES_CORE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
WB_CONMAX	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MEM_CTRL	0.0000	0.0136	0.0369	0.1796	0.1921	0.3012
AC97_CTRL	0.0000	0.0060	0.0298	0.0655	0.0655	0.0655
WB_DMA	0.0000	0.0223	0.0491	0.0513	0.0765	0.1251
PCI_BRIDGE32	0.0000	0.0133	0.0213	0.0372	0.3590	0.5266
VGA_LCD	0.0000	0.0395	0.0570	0.0789	0.1447	0.2500
Avg.	0.0000	0.0122	0.0247	0.0520	0.1051	0.1590

times, depending on the size of the given test vectors for each benchmark circuit and compute, for each self-loop flip-flop, the proportion of how many times it satisfies *Eq.1*, indicating its *steady probability* over the entire sleep mode simulation. Then, we compute its data toggling probability $prob(\cdot)$ in Sec. 3.2.1, which is called $1-$*steady probability*, from which we produce $\mathcal{F}_{self}^{steady}$ by collecting every self-loop flip-flop that meets $prob(\cdot) \leq \gamma$.

Table 3.2 shows the changes of the number of steady self-loop flip-flops and the portion among all self-loop flip-flops as the γ value changes. In addition, Table 3.3 shows the failure probability $prob_f$ of entering sleep state for each benchmark circuit.[3] By observing the changing trend of the values of $|\mathcal{F}_{self}^{steady}|$ and $prob_f$ in Table 3.2 and Table 3.3, we set γ to 0.02.

[3]Impact of failure probability $prob_f$ on energy saving will be discussed in Sec. 3.2.4.

3.2.2 Allocating State Retention Storage

Allocating state retention storage should consider that every **self-loop** flip-flop should be replaced with a distinct retention FF with **at least one bit storage**, which is in fact the major source of preventing the exploitation of MBRFFs from saving the total storage size.

Our allocation strategy is simple namely *treating all self-loop FFs in $\mathcal{F}_{steady}$ obtained in Step 1 as if they were the same as the flip-flops with* **no self-loop** (i.e., partitioning $\mathcal{F}_{all}$ into $\mathcal{F'}_{ordinary}$ (= $\mathcal{F}_{ordinary} \cup \mathcal{F}_{self}^{steady}$) and $\mathcal{F'}_{self}$ (= $\mathcal{F}_{self}^{\sim steady}$) as shown in Step 2 of Fig. 3.6, and performing the following two steps:

2.1 Generating a set S of flip-flop dependency subgraphs by decomposing the original circuit graph, so that every self-loop FF $f_i \in \mathcal{F}_{self}^{\sim steady}$ in the decomposed maximal subgraphs should have no driving flip-flops (i.e., no predecessors) since we cannot say that its state will be surely recovered by the help of its driven flip-flop(s).

2.2 Applying any conventional retention storage allocation algorithm to all subgraphs in S independently while ensuring at least one-bit allocation for every self-loop FF $f_i \in \mathcal{F}_{self}^{\sim steady}$.[4]

3.2.3 Designing and Optimizing Steady State Monitoring Logic

From the allocation result in Step 2, the flip-flops in $\mathcal{F}_{self}^{steady}$ can be classified into two groups: (1) FFs with retention storage, (2) FFs with no retention storage, as shown by the blue arrows from Step 1 to Step 3 in Fig. 3.6, in which supporting of group 2 is possible only when all flip-flops in group 2 should satisfy the self-loop removal condition (i.e., *Eq.3.1*), as described in Sec. 3.1.2. We design a logic circuitry monitoring the

[4]We applied the algorithm in [24] as our retention storage allocation in experiments though any of the conventional algorithms is applicable.

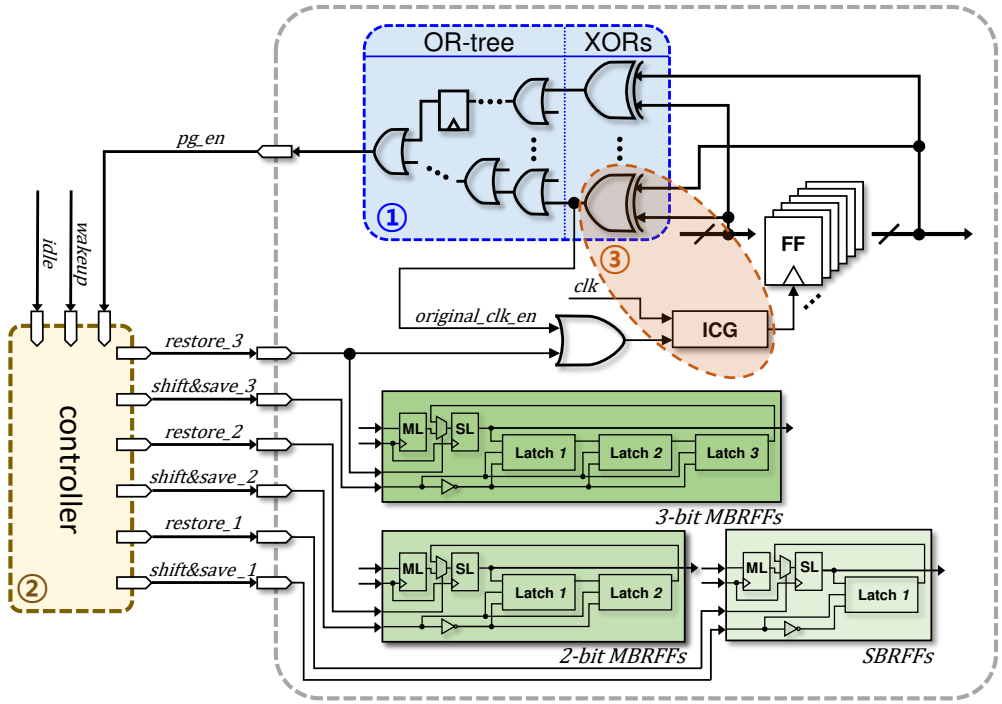


Figure 3.7: State monitoring circuitry for the flip-flops in $\mathcal{F}_{loop}^{steady}$ with no retention storage (①), power gating controller (②), and resource sharing with clock gating logic (③).

condition of the flip-flops in group 2 (labeled f_l in Step 3 of Fig. 3.6). The flip-flops in $\mathcal{F}_{self}^{\sim steady}$ and $\mathcal{F}_{ordinary}$ do not require monitoring logic, as they have retention storage and have no self-loop, respectively.

① *State monitoring logic for flip-flops in* $\mathcal{F}_{self}^{steady}$: Our state monitoring logic in gating power is shown in the blue box in Fig. 3.7, containing XOR gates, one for each in $\mathcal{F}_{self}^{steady}$ with no retention storage and ORing them to produce the active-low steady signal *pg_en*.[5]

While constructing the OR-tree, additional flip-flops can be inserted to delay state monitoring signal for correct operation with 3-bit retention FFs. For example, state monitoring signal generated from fanout flip-flops of 3-bit retention FF should be delayed by 1 cycle to trigger *pg_en* at the same clock cycle with the signal generated from fanout flip-flops of 2-bit retention FF.

When the circuit is idle, power gating controller (②) initiates state saving by enabling *shift&save_3*, *shift&save_2* followed by *shift&save_1* in the subsequent clock cycle, where the *shift&save_N* and *restore_N* are control signals for *N-bit* retention FF. The state monitoring result *pg_en* is detected at the l^{th} clock cycle in powerdown mode. Conversely, signals *restore_3*, *restore_2* and *restore_1* are enabled one by one sequentially when signal *wakeup* is issued to the controller.

② *State transition diagram for power gating controller*: An example of timing diagram for state monitoring and state saving is shown in Fig. 3.8. The time interval marked in yellow indicates that the monitoring circuitry detects some of states in $\mathcal{F}_{self}^{steady}$ with no retention storage are not steady at cycle time t_m, letting the circuit still stay in active mode. On the other hand, the time interval marked in blue indicates that the states are all steady at t_{m+3}, letting the states be saved by *shift&save_3*, *shift&save_2* and *shift&save_1*, so that the circuit safely goes to sleep mode. Fig. 3.9 shows the state transition diagram for controlling the save/restore operation shown in Fig. 3.8 accord-

[5]Impact on circuit performance caused by constructing state monitoring logic will be discussed in Sec. 3.4.3

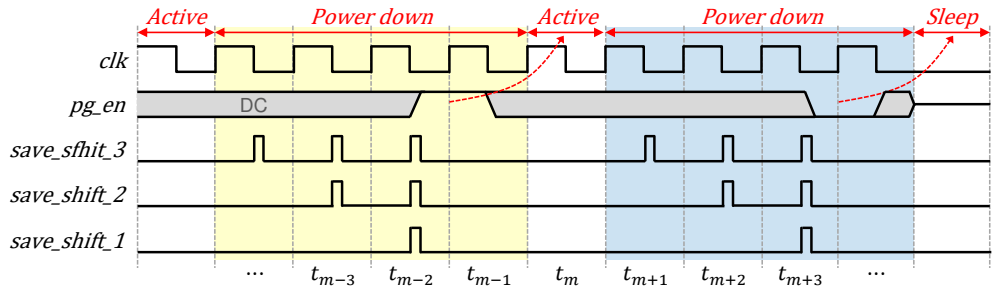


Figure 3.8: Timing diagram showing the transition to sleep mode by monitoring (pg_en) in ① for l (= 3) clock cycles.

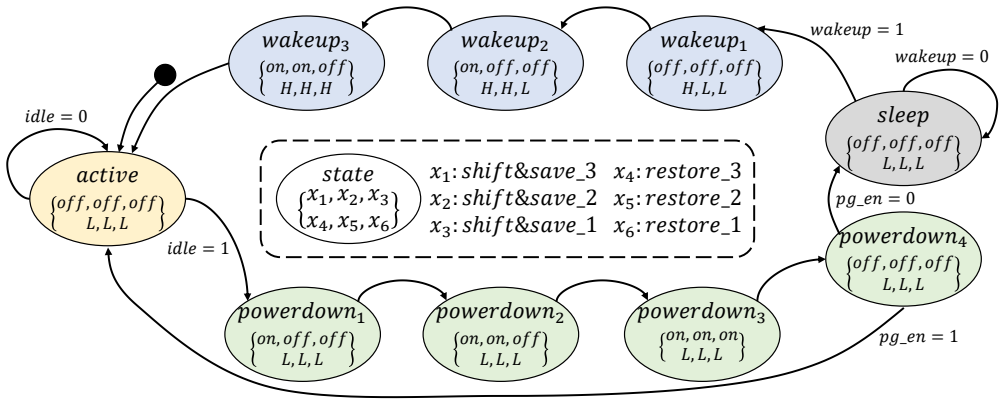


Figure 3.9: State transition diagram for the power gating controller in ②.

ing to the input signals *idle*, *wakeup*, and *pg_en*, from which we can see that only when the circuit is in *idle* and *pg_en* is enabled at l^{th} clock cycle during powerdown mode, the circuit switches to sleep mode. In the Fig. 3.9, *shift&save* signals are indicated as on/off depending on whether it is toggled or not, and the restore signals are indicated as H/L depending on whether it is retained high or low during the clock cycle.

③ *Sharing clock gating resource for state monitoring*: Idle logic driven clock gating (e.g., Fig. 3.1(c)) and data toggling driven clock gating (e.g., Fig. 3.1(d)) are two popular clock gating methods used in industry. As target designs increasingly demand fast clock speeds, it is essential to deploy clock gating to reduce the dynamic power. To boost up the power saving, the data toggling driven clock gating is additionally applied to the flip-flop by allocating XOR gate, as shown in ③ of Fig. 3.7. Consequently, we can share the expensive XOR gate by the toggling based clock gating with our steady state aware power gating.

3.2.4 Analysis of the Impact of Steady State Monitoring Time on the Standby Power

Since our power gating approach is based on the steady state monitoring, a circuit enters sleep mode only when all the monitored self-loop FFs are ensured to be in steady state at the moment they contribute to *pg_en* signal. Thus, the circuit will postpone the transition to sleep mode for a short time until it receives the monitoring signal of all steady states, which shortens the time period in sleep mode accordingly. We formally analyze how much the standby power consumption is affected by the reduced sleep time.

P_a, P_s : (Given) the active and standby power dissipation.

t_a : (Given) the time period the circuit is in executing task.

t_s : (Given) the time period the circuit can be in sleep.

ρ : (Given) the ratio of t_s to t_a.

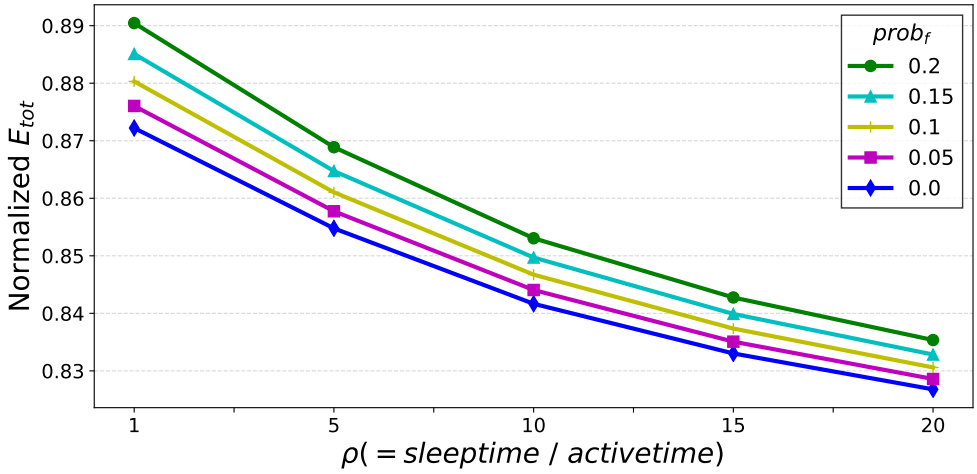


Figure 3.10: The changes of total energy consumption as the values of $prob_f$ and ρ vary. Energy consumption is normalized to that of [24]. Our simulation in Step 1 corresponds to energy curve between blue and purple curves, since we selected a set of self-loop FFs for every benchmark circuit so that the $prob_f$ value became nearly 0.

$prob_f$: (Given) the failure probability of all steady states of the self-loop flop-flops with no retention storage.

t_d : (Given) the time interval between two successive monitoring attempts due to the failure of all steady states (i.e., = $\lambda \times t_a$) where we set $\lambda = 0.1$.

t_{loss} : the delayed time period before entering sleep mode due to the failure(s) of all steady states.

Then, the delay penalty t_{loss} can be computed by

$$t_{loss} = \sum_{m=1}^{M-1} (prob_f)^m \times m \cdot t_d \times (1 - prob_f) + prob_f^M \cdot M \cdot t_d \qquad (3.2)$$

where $M = \lfloor t_s/t_d \rfloor$. The first term denotes successful sleep mode entering after m consecutive failures, and the second term denotes failure to enter sleep mode at all.

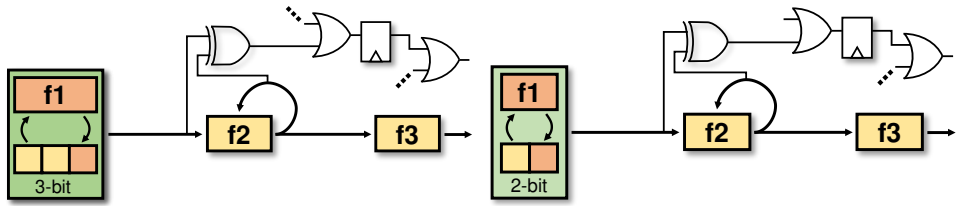
Then we compute the total energy consumption E_{tot}:

$$E_{tot} = (t_a + t_{loss}) \times P_a + (t_s - t_{loss}) \times P_s. \tag{3.3}$$

Fig. 3.10 shows the energy curves as the values of $prob_f$ and ρ vary while the wakeup latency l is set to 3. P_a, P_s values are from Table 3.7 in Sec. 3.4, and results of retention storage refinement which will be discussed in Sec. 3.3 is also included. In our experiments, we match the extraction of self-loop FFs in Step 1 for every circuit with the energy curve between blue and purple curves by constraining the $prob_f$ value to be almost 0, as shown in the Table 3.3. Then, by varying the ρ value (i.e., the ratio of sleep time to active time), we analyze the changes of active and standby power from *Eq.3.3*.

3.3 Retention Storage Refinement Utilizing Steadiness

Proposed method in Sec. 3.2 reduce the total size of retention storage by disregarding self-loop of flip-flops that satisfy *self-loop removal condition*. Retention storage can be reduced further if consecutive identical data are stored when the circuit goes down to sleep mode.



(a) Before refinement on f_1 (b) After refinement on f_1

Figure 3.11: Retention storage in f_1 can be reduced from (a) 3-bit to (b) 2-bit if *retention storage refinement condition* is satisfied.

For example, the wakeup latency l in Fig. 3.11(a) is 3, and f_1 is replaced with a 3-bit retention FF. As f_2 is a steady self-loop FF with state monitoring logic, it is

guaranteed that f_2 was steady at the moment when data of f_2 is stored in the retention storage of f_1 because state monitoring logic has already monitored whether f_2 is steady or not. In other words, state of f_2 is retained for at least 2 cycles, which implies that the states between f_2 and f_4 are identical. As a result, the first 2 bits of retention storage in f_1 stores the same data as it can be seen in Fig. 3.11(a). Then it is possible to reduce the retention storage in f_1 to 2 bits, as shown in Fig 3.11(b), while guaranteeing the correct operation by saving states for 2 cycles and restoring states for 3 cycles.

Consequently, retention storage of MBRFF can be reduced by 1 bit if consecutively saved states are guaranteed to be identical. We formally state this condition:

Retention storage refinement condition: *Retention storage of MBRFF (e.g., f_1 in Fig. 3.11) can be reduced by 1 bit if all the last flip-flops in fanout cone are guaranteed to be steady every time the circuit enters sleep mode.*

By extracting flip-flops that satisfy *retention storage refinement condition*, total size of retention storage, as well as the standby power consumption, is reduced further while not changing the total number of retention FFs.

3.3.1 Extracting Flip-flops for Retention Storage Refinement

Retention storage refinement is performed after the Step 2 of retention storage allocation (Sec. 3.2.1). The detailed process of refining retention storage is described in Algorithm 2.

The algorithm first extracts the list of flip-flops that have multi-bit retention storage (line 1). For each of the retention FFs in RFF_list, line 2 to 16 try to reduce the retention storage. In line 3, the algorithm extracts all the fanout paths and then checks if *retention storage refinement condition* is satisfied for the retention FF (line 4). If the condition is satisfied, flip-flops at the last of each of fanout paths are then referred to candidate for state monitoring (line 7∼line 14). Line 9 checks if the last FF (lFF) has already allocated retention storage. Then, fanin path of lFF is collected (line 12) and monitor is inserted to flip-flops in the fanin path if needed (line 13), which will be

Algorithm 2: Retention storage refinement algorithm

Input: Circuit C with retention storage allocation

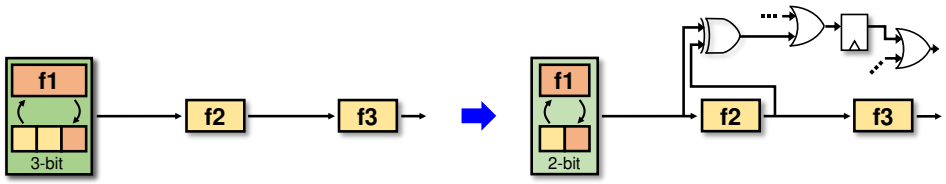
Result: Circuit C' after retention storage refinement

```
1  RFF_list ← get_MBRFFs(C)
2  for RFF ∈ RFF_list do
3      fo_path_list ← search_fo_path(from : RFF, max_depth :
         ret_storage(RFF) − 1)
4      if ! is_steady(fo_path_list) then
5          continue
6      end
7      for p ∈ fo_path_list do
8          lFF ← last FF of p
9          if is_allocated_retention(lFF) then
10             continue
11         end
12         fi_path_list ← search_fi_path(to : lFF, max_depth :
             latency − 1)
13         insert_monitor(fi_path_list)
14     end
15     reduce_storage(RFF)
16 end
```

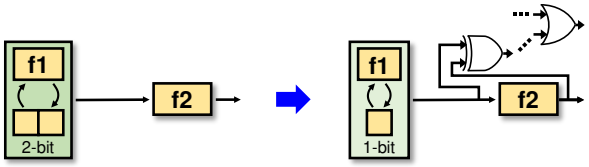
discussed in Sec. 3.3.2. Finally, retention storage of RFF is reduced by 1 bit (line 15).

3.3.2 Designing State Monitoring Logic and Control Signals

Additional state monitoring logic may be required for the retention storage refinement. However, The amount is negligible since it mostly reuses the state monitoring logic implemented for initial retention storage allocation.



(a) Reduce from 3-bit to 2-bit



(b) Reduce from 2-bit to 1-bit

Figure 3.12: State monitoring logic insertion scheme for (a) 3-bit to 2-bit reduction and (b) 2-bit to 1-bit reduction. State monitoring logic is newly inserted only when there is no pre-existing state monitoring logic in the fanin path of last flip-flop (f_3 in (a), f_2 in (b)).

Depending on the existence of self-loop in the fanout flip-flops of MBRFF, there are 4 possible cases for retention storage refinement when $l = 3$, and 2 possible cases when $l = 2$. Among the 6 possible cases, we only show 2 cases that requires additional state monitoring logic insertion in Fig. 3.12. Assume that every self-loop FFs are steady, and the data is fed sequentially from the retention FF f_1 to the following f_2 and f_3, which are either self-loop or ordinary FF. In the below description, *reduce* or *reduction* means reduction in retention storage.

① *Reduce from 3-bit to 2-bit*: When a 3-bit retention FF is reduced to a 2-bit retention

FF, there are four cases depending on whether f_2 or f_3 is self-loop FF. However, as shown in Fig. 3.12(a), additional state monitoring logic is required only when both of f_2 and f_3 are ordinary flip-flops, because the pre-existing state monitoring logic can be used if either f_2 or f_3 is a self-loop FF. Among the f_2 and f_3, inserting state monitoring logic to f_2 rather than f_3 reduces the necessity of additional state monitoring logic for the case that retention storage of f_2 is reduced again to 1-bit.

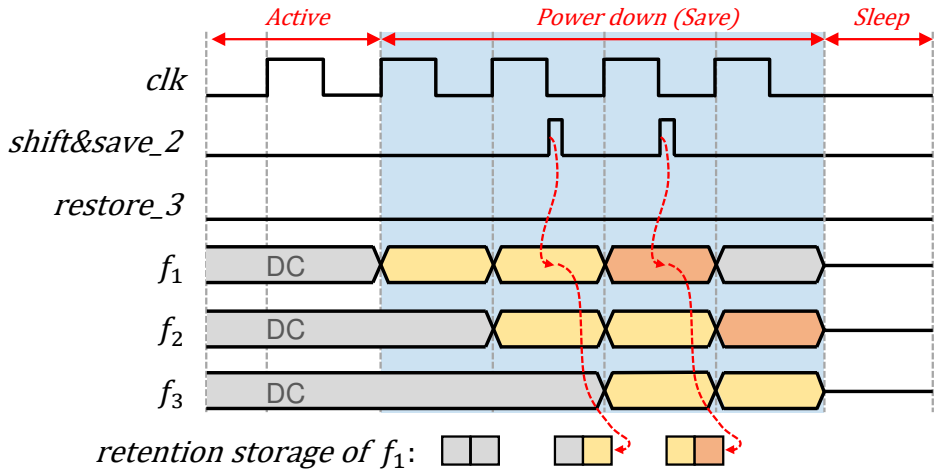
② *Reduce from 2-bit to 1-bit*: State monitoring logic should be inserted to f_2 for retention storage refinement on f_1, as shown in Fig. 3.12(b). Since state monitoring logic already exists if f_2 is self-loop FF, additional state monitoring logic is required only when f_2 is ordinary flip-flop.

Reduced retention flip-flops experience mismatches between the number of bits and the restore cycles. For example, in the 3-bit to 2-bit case, flip-flop should restore data for 3 cycles within 2-bit retention storage. This problem is resolved by changing connection of control signals so that save and restore operations are done in different number of cycles, as follows:

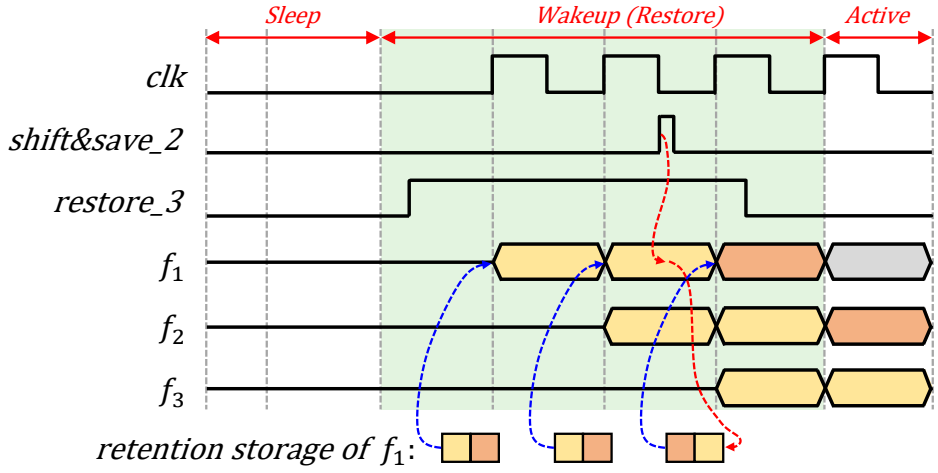
Control signal correction: *Reduced retention flip-flop whose retention storage is reduced from N-bit to N'-bit should be controlled by save&restore signal of N'-bit retention flip-flop (shift&save_N') and restore signal of N-bit retention flip-flop (restore_N).*

For example, if 3-bit retention FF is reduced to 2-bit retention FF (Fig. 3.11), the flip-flop should be controlled by *save&restore_2* signal, saving states for only 2 cycles. Note that *save&restore_2* signal saves only last 2 bits among 3 bits of data stored by *save&restore_3*. States are restored for 3 cycles by *restore_3* signal during wakeup mode, while same state is restored twice at the first 2 cycles because states in retention storage will be shifted by *save&restore_2* signal. Fig. 3.13 shows the timing diagram of control signals and corresponding states of flip-flops for Fig. 3.11. States of f_1, f_2, and f_3 at third clock cycle in powerdown mode are saved in 2-bit retention storage of f_1, and restored at the last clock cycle in wakeup mode.

Fig. 3.14 shows the flow of our design methodology to allocate retention storage



(a) save operation



(b) restore operation

Figure 3.13: Timing diagram of control signals and states of each flip-flops after retention storage refinement in Fig. 3.11.

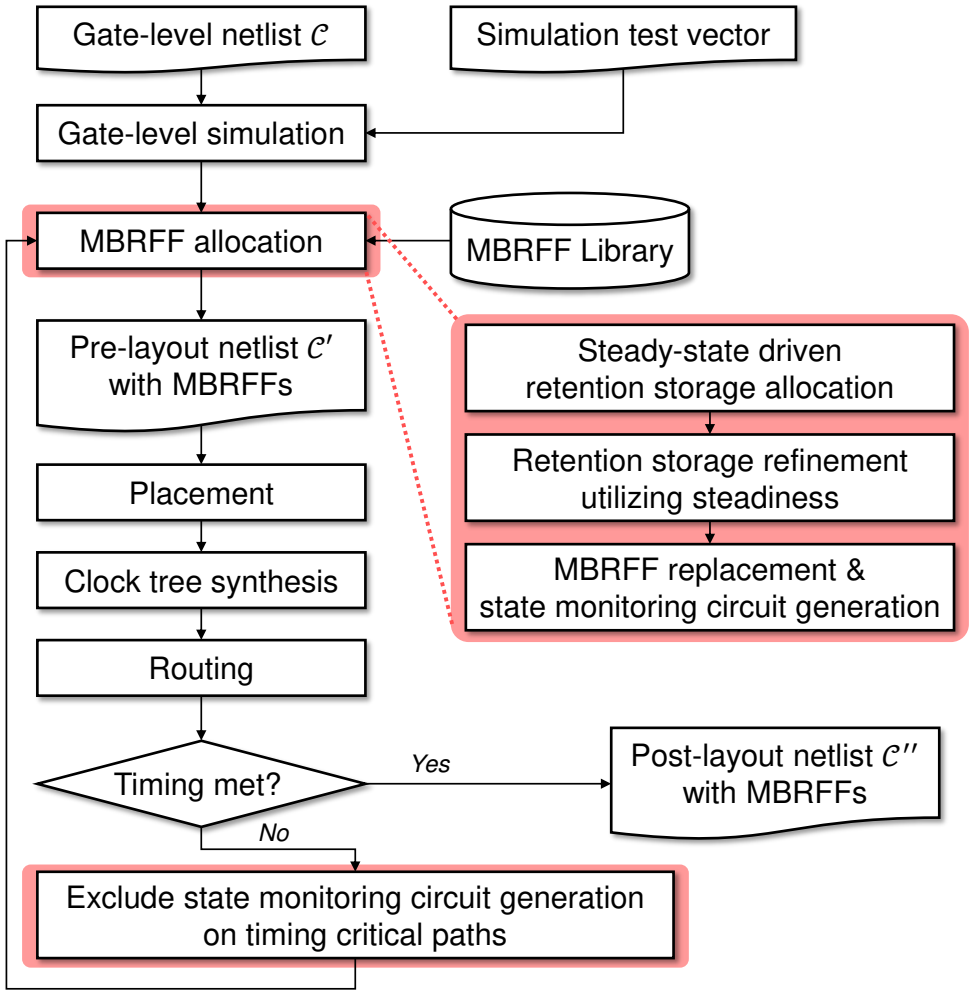


Figure 3.14: Flow of our retention storage allocation and state monitoring circuit generation methodology.

Table 3.4: Comparison of total number of flip-flops deploying state retention storage (#RFFs) and total bits of retention storage (#Rbits) used by [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs).

$l = 2$

Designs	SBRFF alloc.	No-Opt [24]		Partial-Opt [25]		Full-Opt1 (ours)		Full-Opt2 (ours)
	#Rbits	#Rbits	#RFFs	#Rbits	#RFFs	#Rbits	#RFFs	#Rbits
SPI	229	229 (0.00%)	229 (0.00%)	195 (14.85%)	195 (14.85%)	120 (47.60%)	99 (56.77%)	99 (56.77%)
AES_CORE	530	525 (0.94%)	521 (1.70%)	417 (21.32%)	393 (25.85%)	393 (25.85%)	393 (25.85%)	393 (25.85%)
WB_CONMAX	770	770 (0.00%)	642 (16.62%)	738 (4.16%)	738 (4.16%)	642 (16.62%)	642 (16.62%)	642 (16.62%)
MEM_CTRL	1563	1491 (4.61%)	1436 (8.13%)	1414 (9.53%)	1403 (10.24%)	996 (36.28%)	902 (42.29%)	914 (41.52%)
AC97_CTRL	2199	2162 (1.68%)	2133 (3.00%)	2044 (7.05%)	1969 (10.46%)	2152 (2.14%)	2092 (4.87%)	2102 (4.41%)
WB_DMA	3109	3026 (2.67%)	3022 (2.80%)	2947 (5.21%)	2942 (5.37%)	2512 (19.20%)	2129 (31.52%)	2129 (31.52%)
PCI_BRIDGE32	3220	3181 (1.21%)	3109 (3.45%)	3000 (6.83%)	2970 (7.76%)	3069 (4.69%)	2769 (14.01%)	2769 (14.01%)
VGA_LCD	17050	17047 (0.02%)	17015 (0.21%)	16942 (0.63%)	16940 (0.65%)	12606 (26.06%)	12531 (26.5%)	12574 (26.25%)
Avg.	-	- (1.39%)	- (4.49%)	- (8.70%)	- (9.92%)	- (22.31%)	- (27.30%)	- (27.12%)

$l = 3$

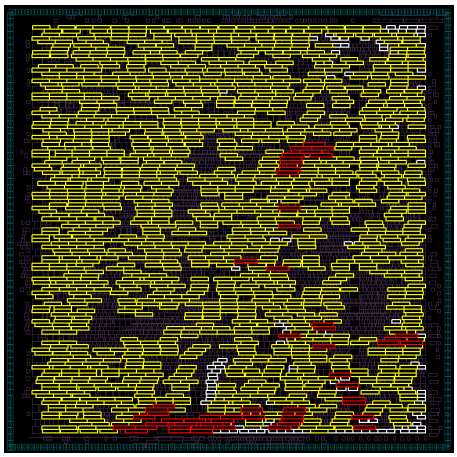
Designs	SBRFF alloc.	No-Opt [24]		Partial-Opt [25]		Full-Opt1 (ours)		Full-Opt2 (ours)
	#Rbits	#Rbits	#RFFs	#Rbits	#RFFs	#Rbits	#RFFs	#Rbits
SPI	229	229 (0.00%)	229 (0.00%)	-	-	118 (48.47%)	98 (57.21%)	98 (57.21%)
AES_CORE	530	525 (0.94%)	521 (1.70%)	-	-	393 (25.85%)	393 (25.85%)	393 (25.85%)
WB_CONMAX	770	770 (0.00%)	642 (16.62%)	-	-	514 (33.25%)	514 (33.25%)	514 (33.25%)
MEM_CTRL	1563	1487 (4.86%)	1433 (8.32%)	-	-	984 (37.04%)	894 (42.80%)	907 (41.97%)
AC97_CTRL	2199	2142 (2.59%)	2121 (3.55%)	-	-	2108 (4.14%)	2064 (6.14%)	2066 (6.05%)
WB_DMA	3109	3026 (2.67%)	3022 (2.80%)	-	-	2209 (28.95%)	1619 (47.93%)	1619 (47.93%)
PCI_BRIDGE32	3220	3147 (2.27%)	3060 (4.97%)	-	-	3049 (5.31%)	2728 (15.28%)	2728 (15.28%)
VGA_LCD	17050	17043 (0.04%)	16998 (0.30%)	-	-	12606 (26.06%)	12519 (26.57%)	12564 (26.31%)
Avg.	-	- (1.67%)	- (4.78%)	-	-	- (26.13%)	- (31.88%)	- (31.73%)

and generate state monitoring circuit. Given gate-level netlist $\mathcal{C}$ and test vector for power gating simulation, steady FFs are identified from gate-level simulation. Then, proposed retention storage allocation method is applied to $\mathcal{C}$, which consists of *steady-state driven retention storage allocation* (Sec. 3.2) and *retention storage refinement* (Sec. 3.3). Since proposed method affects the circuit performance by inserting additional state monitoring logic, the final layout is assigned to post-layout netlist $\mathcal{C}''$ only if timing is met. If not, the flow prohibits the state monitoring circuit generation on timing critical paths (i.e. allocate retention storage) followed by the another iteration.

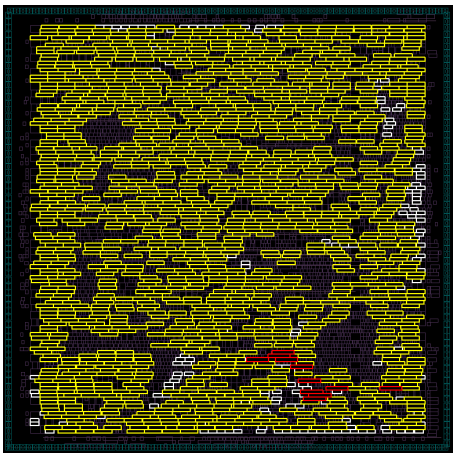
3.4 Experimental Results

We implemented our method in Python using python-igraph package [45] for graph analysis and Gurobi Optimizer [46] for ILP based heuristic algorithm. We also implemented two recent state-of-the-art MBRFF allocation algorithms in [24, 25] and tested them on circuits from IWLS2005 benchmarks [43] and OpenCores [44] to compare their performance in terms of the number of flip-flops with retention storage, total retention bits, and active/standby power[6] with ours. Benchmark circuits are synthesized and implemented using Synopsys Design Compiler and IC compiler with Synopsys 32/28nm generic library. Gate level simulation is performed by using Cadence Xcelium and power consumption is measured by using Synopsys PrimePower while all the circuits are operating at 100MHz in active mode without causing any timing violation. We set the wakeup latency constraint l to 2 and 3 in our experiments as validated by our observation, from which we extracted the steady self-loop FFs by setting parameter γ to 0.02.

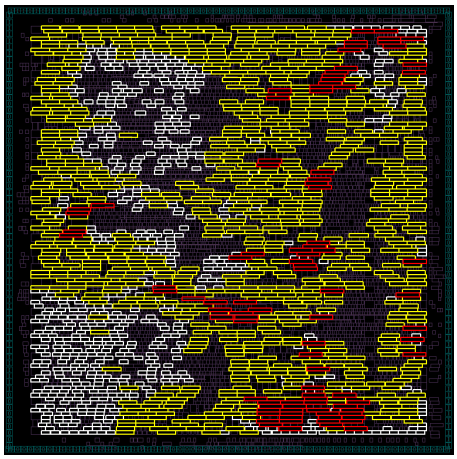
[6] *Active* power refers to the sum of dynamic and leakage power in *active* mode consumed by the circuits including the save/restore control logic while *standby* power refers to the leakage power in *sleep* mode.



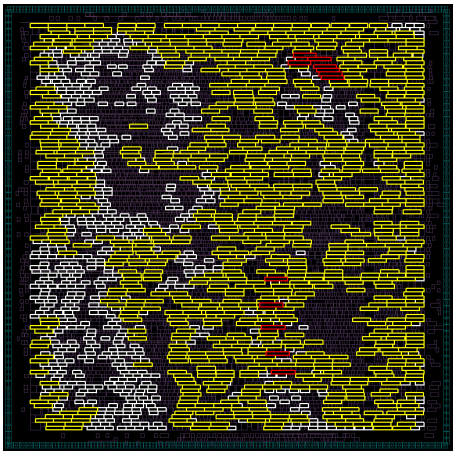
(a) [24] (No optimization on self-loop FFs)



(b) [25] (Partial optimization on self-loop FFs)



(c) Ours (Full optimized on self-loop FFs)



(d) Ours (Full optimized on self-loop FFs with retention storage refinement)

Figure 3.15: Layouts for MEM_CTRL. The colored rectangles represent flip-flops: flip-flops with no retention storage (white), flip-flops with 1-bit retention storage (yellow), and flip-flops with 2-bit retention storage (red).

3.4.1 Comparison of State Retention Storage

Table 3.4 shows a comparison of total bits of retention storage (#Rbits) and total number of retention flip-flops (#RFFs) used by [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs). In the table, Full-Opt1 and Full-Opt2 indicate the proposed method without and with the retention storage refinement, respectively. Column for the number of retention FFs for Full-Opt2 is omitted because it is identical to that of Full-Opt1. To compare the size of retention storage with respect to the total number of bits, we set the baseline in the comparison to that of SBRFF allocation constraining wakeup latency $l = 1$. Note that Partial-Opt is not applicable when $l = 3$ since the method is constrained to $l = 2$.

The low reduction by the conventional allocation methods ([24, 25]) in comparison with ours clearly indicates that for the conventional methods, the self-loop FFs are indeed a big obstacle in saving the state retention bits. For example, for circuits SPI, MEM_CTRL, WB_DMA, and VGA_LCD in which over 80% of FFs have mux-feedback self-loops, the retention bit saving gap between ours and the conventional methods is prominent (i.e., 3x$\sim$40x more saving).

Note that for AC97_CTRL, #Rbits and #RFFs of our method are larger than those of Partial-Opt, causing more power consumption. This is because the ratio of steady self-loop FFs to all self-loop FFs in AC97_CTRL is relatively lower than other circuits, as shown in Table 3.2, which is not a favorable condition for our method to be effective.

Fig. 3.15 shows the layouts of MEM_CTRL produced by [24], [25], and ours with $l = 2$. It is identified that the number of retention FFs is reduced in Fig. 3.15(c) compared to Figs. 3.15(a) and (b), and the number of 2-bit retention FFs is reduced in Fig. 3.15(d) due to the retention storage refinement.

Table 3.5, 3.6 and Fig. 3.16 show the detailed cell area comparison of each logic component for $l = 2$ and $l = 3$. FF, Ctrl, and Comb represent the normal FF or retention FF, always-on control logic, and combinational logic including state monitoring

Table 3.5: Comparison of cell area occupied by flip-flops(FF), always-on control logic(Ctrl) and combinational logic including state monitoring logic and excluding always-on control logic(Comb) in [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs). Wakeup latency l is 2.

Designs	No-OPT [24]		Partial-Opt [25]		Full-Opt1 (Ours)		Full-Opt2 (Ours)	
	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)
SPI	6190	FF: 3156 Ctrl: 251 Comb: 2783	5945 (3.95%)	FF: 2932 (7.10%) Ctrl: 213 (15.01%) Comb: 2800 (0.60%)	5810 (6.14%)	FF: 2416 (23.44%) Ctrl: 110 (55.98%) Comb: 3284 (-17.97%)	5681 (8.21%)	FF: 2296 (27.24%) Ctrl: 111 (55.58%) Comb: 3274 (-17.63%)
AES_CORE	29259	FF: 7232 Ctrl: 744 Comb: 21283	28776 (1.65%)	FF: 6449 (10.82%) Ctrl: 571 (23.29%) Comb: 21714 (-2.22%)	28305 (3.26%)	FF: 6303 (12.84%) Ctrl: 529 (28.89%) Comb: 21473 (-0.89%)	28305 (3.26%)	FF: 6303 (12.84%) Ctrl: 529 (28.89%) Comb: 21473 (-0.89%)
WB_CONMAX	67010	FF: 10489 Ctrl: 935 Comb: 55586	67302 (-0.44%)	FF: 10499 (-0.09%) Ctrl: 1148 (-22.77%) Comb: 54998 (-0.14%)	64978 (3.03%)	FF: 9844 (6.16%) Ctrl: 934 (0.16%) Comb: 54201 (2.49%)	64978 (3.03%)	FF: 9844 (6.16%) Ctrl: 934 (0.16%) Comb: 54201 (2.49%)
MEM_CTRL	33805	FF: 20940 Ctrl: 1925 Comb: 10941	33389 (1.23%)	FF: 20463 (2.28%) Ctrl: 1804 (6.28%) Comb: 10655 (-1.89%)	30133 (10.86%)	FF: 17393 (16.94%) Ctrl: 1241 (35.54%) Comb: 11499 (-5.10%)	30182 (10.72%)	FF: 16939 (19.10%) Ctrl: 1242 (35.49%) Comb: 12000 (-9.69%)
AC97_CTRL	42558	FF: 29916 Ctrl: 2681 Comb: 9962	41674 (2.08%)	FF: 29015 (3.01%) Ctrl: 2596 (3.17%) Comb: 9253 (-0.74%)	42572 (-0.03%)	FF: 29809 (0.36%) Ctrl: 2639 (1.57%) Comb: 10125 (-1.64%)	42350 (0.49%)	FF: 29522 (1.32%) Ctrl: 2632 (1.82%) Comb: 10196 (-2.35%)
WB_DMA	79528	FF: 42454 Ctrl: 4072 Comb: 33002	79917 (-0.49%)	FF: 41932 (1.23%) Ctrl: 4426 (-8.67%) Comb: 32364 (-1.92%)	77285 (2.82%)	FF: 38237 (9.93%) Ctrl: 3184 (21.82%) Comb: 35864 (-8.67%)	75019 (5.67%)	FF: 36096 (14.98%) Ctrl: 3034 (25.49%) Comb: 35889 (-8.75%)
PCI_BRIDGE32	63511	FF: 43865 Ctrl: 3963 Comb: 15682	62601 (1.43%)	FF: 42698 (2.66%) Ctrl: 3883 (2.01%) Comb: 14832 (-2.41%)	63567 (-0.09%)	FF: 42878 (2.25%) Ctrl: 3657 (7.73%) Comb: 17032 (-8.61%)	62160 (2.13%)	FF: 41262 (5.94%) Ctrl: 3558 (10.23%) Comb: 17340 (-10.57%)
VGA_LCD	323058	FF: 233906 Ctrl: 22030 Comb: 67122	324482 (-0.44%)	FF: 233239 (0.29%) Ctrl: 22675 (-2.93%) Comb: 61056 (-1.66%)	285175 (11.73%)	FF: 202361 (13.49%) Ctrl: 17010 (22.78%) Comb: 65803 (1.96%)	283671 (12.19%)	FF: 202235 (13.54%) Ctrl: 16809 (23.70%) Comb: 64628 (3.72%)
Avg.	-	-	- (1.12%)		- (4.72%)	-	- (5.71%)	-

Table 3.6: Same as Table 3.5, with wakeup latency $l = 3$.

Designs	No-OPT [24]		Partial-Opt [25]		Full-Opt1 (Ours)		Full-Opt2 (Ours)	
	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)	Cell Area (μm^2)	Detailed Area (μm^2)
SPI	6190	FF: 3156 Ctrl: 251 Comb: 2783	-	FF: - Ctrl: - Comb: -	5731 (7.40%)	FF: 2402 (23.88%) Ctrl: 102 (59.43%) Comb: 3228 (-15.96%)	5561 (10.16%)	FF: 2287 (27.53%) Ctrl: 111 (55.58%) Comb: 3163 (-13.62%)
AES_CORE	29314	FF: 7232 Ctrl: 753 Comb: 21330	-	FF: - Ctrl: - Comb: -	28305 (3.44%)	FF: 6303 (12.84%) Ctrl: 529 (29.71%) Comb: 21473 (-0.67%)	28757 (1.90%)	FF: 6303 (12.84%) Ctrl: 538 (28.49%) Comb: 21916 (-2.75%)
WB_CONMAX	66876	FF: 10489 Ctrl: 944 Comb: 55443	-	FF: - Ctrl: - Comb: -	64110 (4.14%)	FF: 8929 (14.88%) Ctrl: 770 (18.42%) Comb: 54411 (1.86%)	64110 (4.14%)	FF: 8929 (14.88%) Ctrl: 770 (18.42%) Comb: 54411 (1.86%)
MEM_CTRL	33907	FF: 20911 Ctrl: 1886 Comb: 11110	-	FF: - Ctrl: - Comb: -	30222 (10.87%)	FF: 17315 (17.20%) Ctrl: 1205 (36.09%) Comb: 11702 (-5.33%)	30163 (11.04%)	FF: 16891 (19.23%) Ctrl: 1204 (36.17%) Comb: 12069 (-8.63%)
AC97_CTRL	42576	FF: 29800 Ctrl: 2629 Comb: 10147	-	FF: - Ctrl: - Comb: -	42222 (0.83%)	FF: 29536 (0.89%) Ctrl: 2586 (1.62%) Comb: 10100 (0.47%)	42081 (1.16%)	FF: 29309 (1.65%) Ctrl: 2607 (0.83%) Comb: 10165 (-0.18%)
WB_DMA	79574	FF: 42454 Ctrl: 4064 Comb: 33056	-	FF: - Ctrl: - Comb: -	75915 (4.60%)	FF: 36032 (15.13%) Ctrl: 2476 (39.07%) Comb: 37407 (-13.16%)	71973 (9.55%)	FF: 32693 (22.99%) Ctrl: 2354 (42.06%) Comb: 36925 (-11.70%)
PCI_BRIDGE32	63359	FF: 43637 Ctrl: 3923 Comb: 15798	-	FF: - Ctrl: - Comb: -	63546 (-0.30%)	FF: 42710 (2.13%) Ctrl: 3563 (9.20%) Comb: 17273 (-9.34%)	61852 (2.38%)	FF: 40934 (6.19%) Ctrl: 3542 (9.72%) Comb: 17376 (-9.99%)
VGA_LCD	327750	FF: 233861 Ctrl: 22792 Comb: 71097	-	FF: - Ctrl: - Comb: -	283998 (13.35%)	FF: 202362 (13.47%) Ctrl: 16744 (26.54%) Comb: 64891 (8.73%)	284556 (13.18%)	FF: 202162 (13.55%) Ctrl: 16841 (26.11%) Comb: 65553 (7.80%)
Avg.	-	-	-	-	- (5.54%)	-	- (6.69%)	-

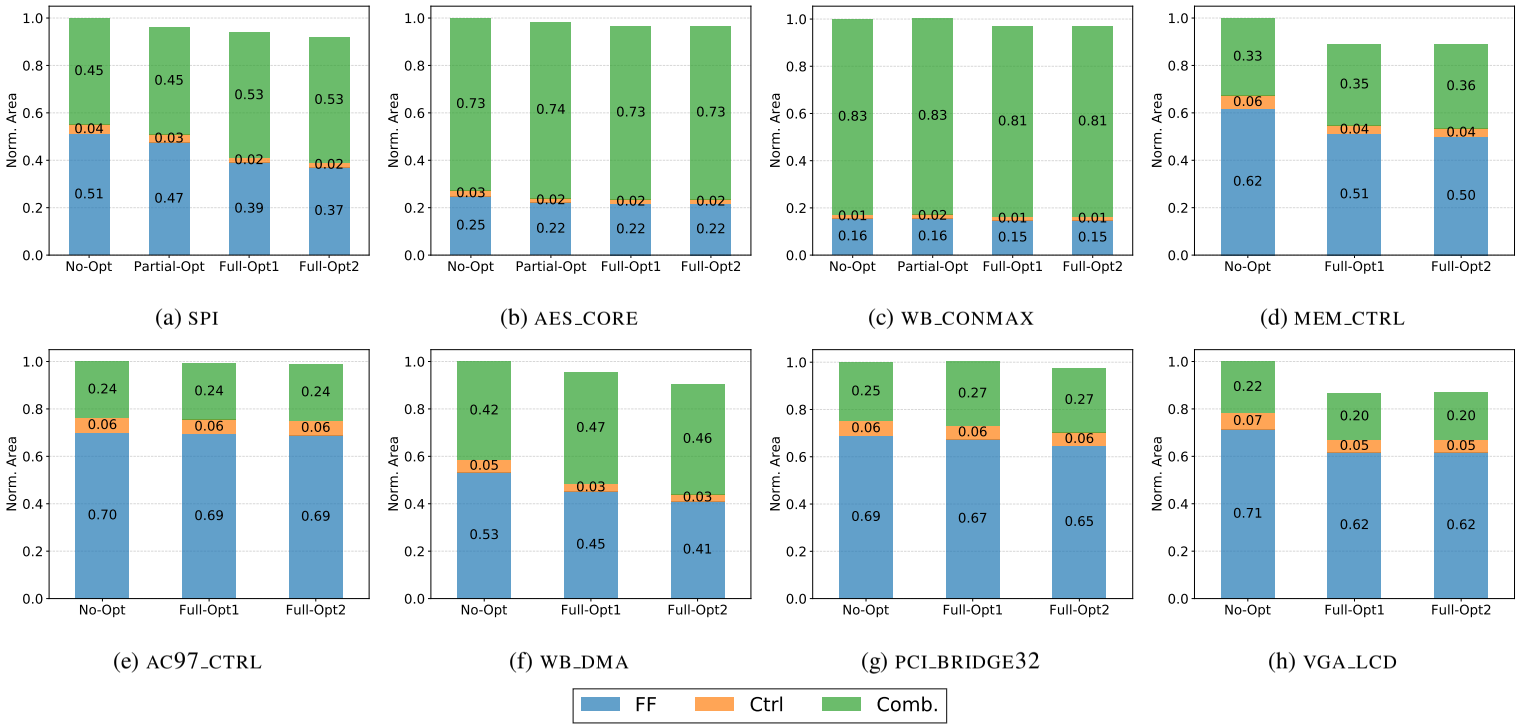


Figure 3.16: Detailed comparison of cell area in each method for each design with (a)∼(d) $l = 2$ and (e)∼(h) $l = 3$.

logic and excluding the always-on control logic, respectively. After retention storage refinement, cell area of all the designs are decreased due to smaller number of large retention FFs followed by less always-on control logic overhead. As a result, total cell area is decreased by 5.71% for $l = 2$ and 6.69% for $l = 3$.

3.4.2 Comparison of Power Consumption

Table 3.7 shows the comparison of the active power which is the sum of dynamic and leakage power in active mode and the standby power which is the leakage power consumed by the high-Vth always-on retention storage in sleep mode for the power gated circuits produced by [24] (No-Opt), [25] (Partial-Opt), and ours (Full-Opt1, Full-Opt2). Unlike the comparison of the retention storage in Table 3.4, active and standby power are compared with that of No-Opt for fair comparison with respect to wakeup latency constraint l. In summary, our steady state monitoring approach is able to reduce the active and standby power by 10.84% and 19.41% when $l = 2$, and 12.16% and 22.34% when $l = 3$, respectively. In addition, we measured the standby power consumed by each of logic element groups and showed in Fig. 3.17. In the figures, RFF (blue), Ctrl (orange), and Power Management (green) are standby power consumed by retention FFs, always-on control logic, and power management cells such as isolation cells and power switch cells. As a result of the proposed method, the size of retention storage is reduced, thereby reducing the standby power consumed by the retention FFs and always-on control logic.

Since the power gated design whose retention storage is allocated by proposed method has the possibility of failing to enter sleep mode, power reduction in Table 3.7 cannot be applied directly. Instead, we analyzed the impact of failure probability $prob_f$ on total energy consumption in Sec. 3.2.4. With the consideration of $prob_f$ for each benchmark circuit with $\gamma = 0.02$ shown in Table 3.3, our method reduced E_{tot} by more than 10% as shown in Fig. 3.10.

Table 3.7: Comparison of the active power (= dynamic + leakage in active mode) and standby power (= leakage in sleep mode) consumed by [24] (No optimization on self-loop FFs), [25] (Partial optimization on self-loop FFs), and ours (Full optimization on self-loop FFs).

$l = 2$

Designs	Active power (= dynamic+leakage in active mode) (μW)				Standby power (=leakage in sleep mode) (μW)			
	No-OPT [24]	Partial-Opt [25]	Full-Opt1 (Ours)	Full-Opt2 (Ours)	NO-OPT [24]	Partial-Opt [25]	Full-Opt1 (Ours)	Full-Opt2 (Ours)
SPI	1041	960 (7.79%)	697 (33.03%)	676 (35.02%)	62.9	55.49 (11.81%)	36.84 (41.45%)	35.11 (44.20%)
AES_CORE	7928	7741 (2.36%)	7832 (1.21%)	7832 (1.21%)	194.7	168.7 (13.35%)	161.6 (17.00%)	161.6 (17.00%)
WB_CONMAX	47700	47400 (0.63%)	47100 (1.26%)	47100 (1.26%)	572.2	608.9 (-6.41%)	524.2 (8.39%)	524.2 (8.39%)
MEM_CTRL	3424	3448 (-0.70%)	2970 (13.26%)	2970 (13.26%)	426.7	411.3 (3.61%)	303.3 (28.92%)	299.4 (29.83%)
AC97_CTRL	3026	2982 (1.45%)	2981 (1.49%)	2938 (2.91%)	554.0	538.3 (2.83%)	549.5 (0.81%)	545.7 (1.50%)
WB_DMA	10100	10100 (0.00%)	9617 (4.78%)	9557 (5.38%)	911.9	953.2 (-4.53%)	751.7 (17.57%)	709.2 (22.23%)
PCI_BRIDGE32	5429	5263 (3.06%)	4939 (9.03%)	4765 (12.23%)	831.2	813.2 (2.17%)	795.8 (4.26%)	754.6 (9.22%)
VGA_LCD	25100	24900 (0.80%)	21000 (16.33%)	20700 (17.53%)	4340.0	4419 (-1.82%)	3367 (22.42%)	3346 (22.90%)
Avg.	-	- (1.92%)	- (10.05%)	- (10.84%)	-	- (2.63%)	- (17.6%)	- (19.41%)

$l = 3$

Designs	Active power (= dynamic+leakage in active mode) (μW)				Standby power (=leakage in sleep mode) (μW)			
	No-OPT [24]	Partial-Opt [25]	Full-Opt1 (Ours)	Full-Opt2 (Ours)	NO-OPT [24]	Partial-Opt [25]	Full-Opt1 (Ours)	Full-Opt2 (Ours)
SPI	1041	-	670 (35.64%)	652 (37.34%)	62.9	-	35.61 (43.40%)	35.05 (44.29%)
AES_CORE	7942	-	7832 (1.39%)	7856 (1.08%)	195.6	-	161.6 (17.38%)	161.6 (17.38%)
WB_CONMAX	47700	-	47200 (1.05%)	47200 (1.05%)	581.7	-	492.1 (15.40%)	492.1 (15.40%)
MEM_CTRL	3452	-	3004 (12.98%)	3040 (11.94%)	421.7	-	296.6 (29.67%)	291.9 (30.78%)
AC97_CTRL	3075	-	2949 (4.10%)	2948 (4.13%)	548.5	-	541.3 (1.31%)	540.5 (1.46%)
WB_DMA	10100	-	9435 (6.58%)	9197 (8.94%)	915.2	-	639.1 (30.17%)	583.9 (36.20%)
PCI_BRIDGE32	5340	-	4895 (8.33%)	4763 (10.81%)	825.7	-	774.1 (6.25%)	753.4 (8.76%)
VGA_LCD	26400	-	20800 (21.21%)	20600 (21.97%)	4442.0	-	3341 (24.79%)	3346 (24.67%)
Avg.	-	-	- (11.41%)	- (12.16%)	-	-	- (21.05%)	- (22.34%)

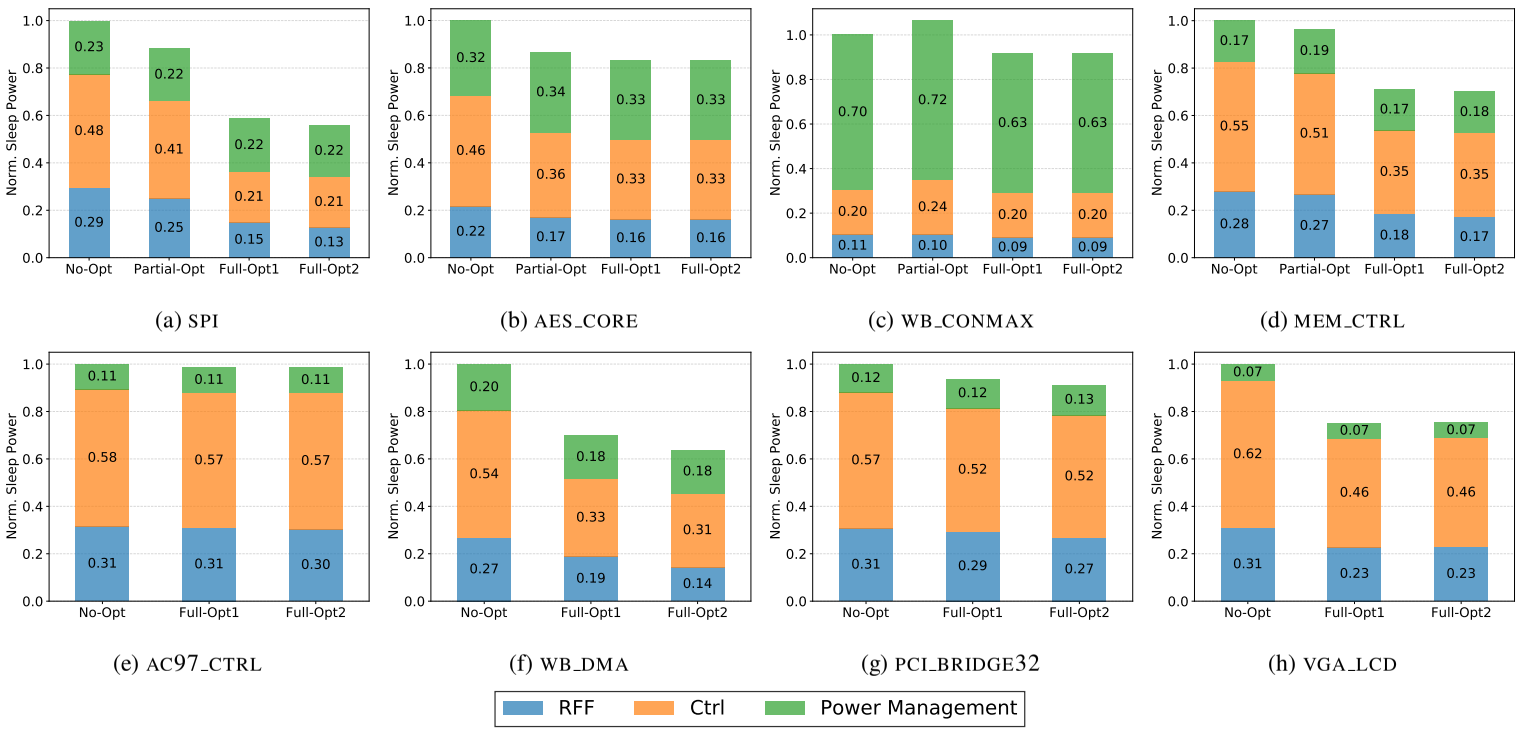


Figure 3.17: Detailed comparison of normalized standby power in each method for each design with (a)∼(d) $l = 2$ and (e)∼(h) $l = 3$.

3.4.3 Impact on Circuit Performance

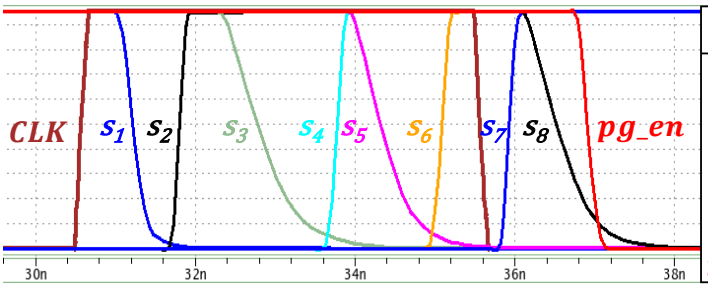
Our retention storage allocation method requires insertion of state monitoring logic, which induce non-negligible path delay for *pg_en* signal generation. However, it should be noted that path delay does not matter since the *pg_en* signal is not used in active mode, and for most of power gating controllers, the supply voltage gradually goes down, causing clock speed to be slow enough to afford the delay increase [47]. The delay caused by monitoring logic is proportional to $log\ n$ where n is the number of required XOR gates as shown in Fig. 3.18, in which total of 596 XORed signals are ORed through only 8 levels of logic. The corresponding pg_en signals do not cause any timing violation in the circuit operating in 100MHz.

Table 3.8: f_{max} comparison of No-Opt [24] and Full-Opt2

Designs	No-Opt	Full-Opt2 (Ours)	
	f_{max} (MHz)	f_{max} (MHz)	# iteration
SPI	297.67	348.30	1
AES_CORE	265.29	254.73	1
WB_CONMAX	232.73	238.14	2
MEM_CTRL	231.15	266.67	1
AC97_CTRL	476.28	497.09	1
WB_DMA	164.63	176.55	1
PCI_BRIDGE32	244.39	272.34	2
VGA_LCD	212.01	276.01	4

Table 3.8 shows maximum frequency of each design along with the number of iteration in Fig. 3.14 while ignoring the delay of *pg_en* signal in active mode. Through the iteration, we approved the final layout when the performance loss due to state monitoring is less than 5%. As shown in the table, for most designs our method reveals better performance over the conventional method within a few iterations. However, it is hard to clearly find out the reason why the performance of a particular circuit is improved or degraded because they are optimized during logic synthesis and P&R by tool with

different retention storage allocation and state monitoring logic. One obvious fact is that the delay of a flip-flop with retention storage is a little longer than that of a flip-flop with no retention storage whereas the state monitoring logic causes increase in the path delay. In this light, our method reduces the number of retention flip-flops by 27.30% (for $l = 2$ in Table 3.4), which is good for timing, but it uses state monitoring logic, which is bad for timing. For AES_CORE, we can roughly say that timing degradation by state monitoring logic may outweigh timing improvement by reducing the flip-flop count with retention storage.



	# XORed signals	delay [ns]
s_1	0(before XOR)	0.66
s_2	2	1.24
s_3	7	2.24
s_4	28	3.21
s_5	51	3.78
s_6	107	4.52
s_7	167	5.36
s_8	593	5.98
pg_en	596	6.35

Figure 3.18: Spice simulation generating pg_en signal through state monitoring logic for circuit MEM_CTRL.

3.4.4 Support for Immediate Power Gating

Power gated design whose retention storage are allocated by proposed method can enter sleep mode only when all the self-loop FFs being monitored are guaranteed to be steady. Therefore, it cannot cope with situations where immediate power gating is required, such as when the chip temperature has reached its thermal limit. In order to avoid rejection of entering sleep mode due to power gating failure probability and enter sleep mode immediately, it should be possible to enter sleep mode regardless of the monitoring result.

To support immediate power gating, we additionally allocated 1-bit retention storage to all the self-loop FFs that no retention storage is allocated previously, and con-

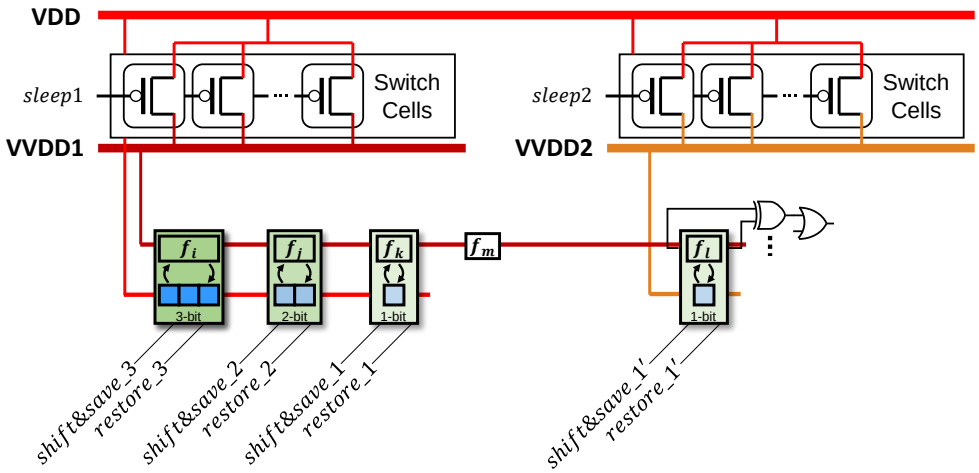


Figure 3.19: Power connection to flip-flops whose retention storage are allocated by proposed method supporting immediate power gating.

Table 3.9: Power state table of powers in Fig. 3.19

Power mode	VVDD1	VVDD2	VDD
ACTIVE	ON	ON	ON
SLEEP1	OFF	ON	ON
SLEEP2	OFF	OFF	ON

nected control signals. The resultant power connection and its power state table are shown in Fig. 3.19 and Table 3.9, where all the combinational cells and ordinary FFs are powered by VVDD1 and newly allocated 1-bit retention storage is powered by VVDD2. Labels of each flip-flop in Fig. 3.19 correspond to that of each flip-flop in Fig. 3.6. ACTIVE and SLEEP2 mode in Table 3.9 are same as active and sleep mode discussed in Sec. 3.4.2. When immediate power gating is required (SLEEP1), only VVDD2 and VVDD are turned on to retain all the states of retention storage, regardless of *self-loop removal condition*. Note that control signals of newly allocated 1-bit retention storage cannot be shared with that of previously allocated 1-bit retention storage because the newly allocated 1-bit retention storage does not save and restore states when the circuit enters into SLEEP2 and wakeup.

Table 3.10: Total number of flip-flops deploying state retention storage (#RFFs) and total bits of retention storage (#Rbits) used by ours supporting immediate power gating

Designs	Full-Opt2 + iPG(ours)			
	$l = 2$		$l = 3$	
	#Rbits	#RFFs	#Rbits	#RFFs
SPI	229 (0.00%)	229 (0.00%)	229 (0.00%)	229 (0.00%)
AES_CORE	521 (1.70%)	521 (1.70%)	521 (1.70%)	521 (1.70%)
WB_CONMAX	770 (0.00%)	770 (0.00%)	642 (16.62%)	642 (16.62%)
MEM_CTRL	1455 (6.91%)	1443 (7.68%)	1448 (7.36%)	1435 (8.19%)
AC97_CTRL	2152 (2.14%)	2142 (2.59%)	2123 (3.46%)	2121 (3.55%)
WB_DMA	3054 (1.77%)	3054 (1.77%)	3003 (3.41%)	3003 (3.41%)
PCI_BRIDGE32	3105 (3.57%)	3105 (3.57%)	3071 (4.63%)	3071 (4.63%)
VGA_LCD	17049 (0.01%)	17006 (0.26%)	17039 (0.06%)	16994 (0.33%)
Avg.	- (2.01%)	- (2.20%)	- (4.65%)	- (4.80%)

Table 3.10 shows the total bits of retention storage (#Rbits)) and total number of retention flip-flops (#RFFs) used by proposed method with additional 1-bit reten-

Table 3.11: Active power and standby power in each of sleep modes consumed by ours supporting immediate power gating.

Designs	Full-Opt2 + iPG(ours)					
	$l = 2$			$l = 3$		
	Active power (μW)	Standby power (SLEEP1) (μW)	Standby power (SLEEP2) (μW)	Active power (μW)	Standby power (SLEEP1) (μW)	Standby power (SLEEP2) (μW)
SPI	1041 (12.32%)	67.67 (-7.55%)	44.75 (28.88%)	930 (10.66%)	71.78 (-14.08%)	47.06 (25.21%)
AES_CORE	7928 (-1.15%)	206.9 (-6.27%)	197.4 (-1.39%)	8019 (-0.97%)	206.9 (-5.78%)	197.4 (-0.92%)
WB_CONMAX	47700 (1.05%)	587.1 (-2.60%)	594.6 (-3.91%)	47000 (1.47%)	554.2 (4.73%)	560.3 (3.68%)
MEM_CTRL	3424 (-8.88%)	449.8 (-5.41%)	344.3 (19.31%)	3620 (-4.87%)	443.4 (-5.15%)	335.4 (20.46%)
AC97_CTRL	3026 (-2.28%)	570.4 (-3.01%)	590.3 (-6.55%)	3140 (-2.11%)	572.7 (-4.41%)	588 (-7.20%)
WB_DMA	10100 (-4.95%)	972.3 (-6.62%)	783.3 (14.10%)	10800 (-6.93%)	974.1 (-6.44%)	665.7 (27.26%)
PCI_BRIDGE32	5429 (-3.68%)	871.6 (-4.86%)	819.9 (1.36%)	5471 (-2.45%)	858.2 (-3.94%)	808 (2.14%)
VGA_LCD	25100 (-1.59%)	4620 (-6.45%)	3638 (16.18%)	25100 (4.92%)	4591 (-3.35%)	3605 (18.84%)
Avg.	- (-1.15%)	- (-5.35%)	- (8.50%)	- (-0.03%)	- (-4.80%)	- (11.18%)

tion storage allocation for immediate power gating. The baseline in the comparison is SBRFF allocation in Table 3.4. Due to the allocation of additional 1-bit retention storage for immediate power gating, the average saving of #Rbits and #RFFs are decreased to level a slightly higher than that of No-Opt.

Table 3.11 shows the active and standby power consumption in each of the power modes in Table 3.9, used by proposed method with additional 1-bit retention storage allocation for immediate power gating. The power saving is compared with that of No-Opt [24]. Due to the increased number of retention storage, additional always-on control logic for them, and additional power switch cells to control VVDD2, average power saving is decreased, even consuming more power in ACTIVE and SLEEP1 mode. Standby power consumed by each cell type in SLEEP1 and SEEP2 modes are shown in Fig. 3.20.

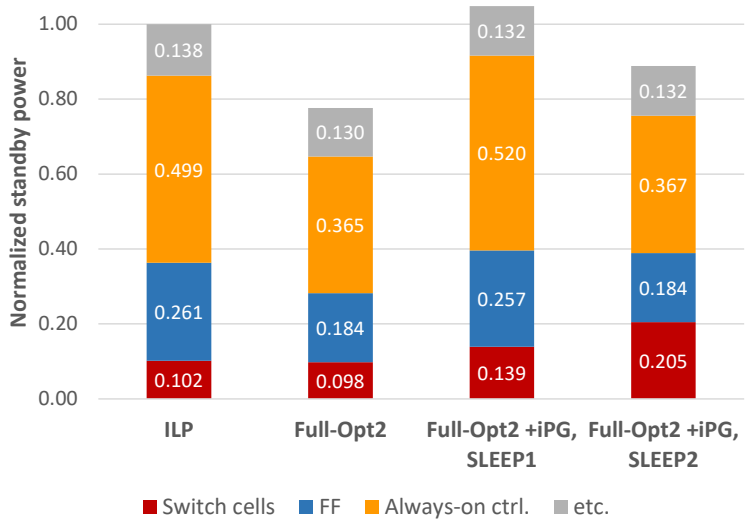


Figure 3.20: Detailed comparison of normalized standby power consumed by each cell type in each of power modes when wakeup latency l is 3.

Similar to Sec. 3.2.4, we formally analyze how much the additional 1-bit retention storage for immediate power gating affects to the total energy consumption. Fig. 3.21

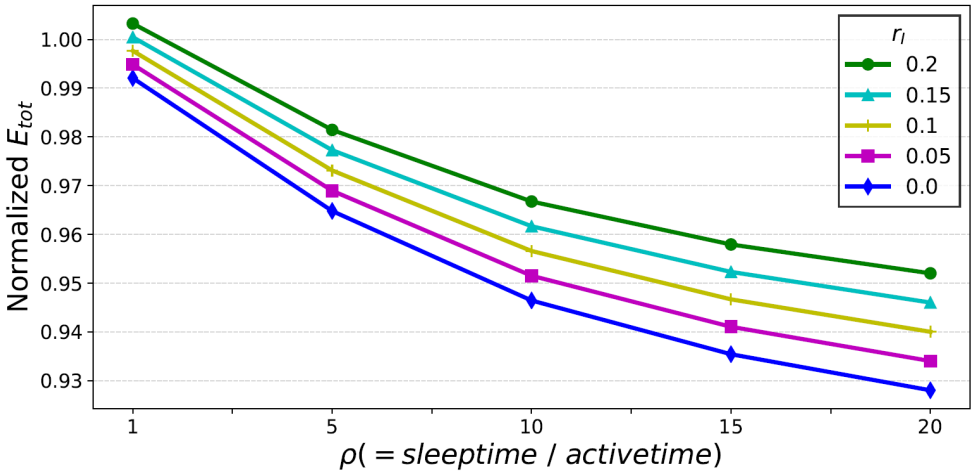


Figure 3.21: The changes of total energy consumption as the values of r_I and ρ vary, while γ is fixed to 0.02. Energy consumption is normalized to that of [24].

shows the change of total energy consumption while varying r_I and ρ with fixed $\gamma(= 0.02)$, where r_I is ratio of the number of immediate power gating to total number of power gating. Although there is still energy saving depending on the ρ and r_I values, because of overhead induced by additional logic supporting immediate power gating, ρ bigger than 10 and r_I smaller than 0.05 are required for more than 5% energy saving.

Chapter 4

Conclusions

4.1 Chapter 2

In Chapter 2, we proposed a comprehensive on-chip monitoring methodology for accurately estimating SRAM V_{ddmin} on each die that does not cause SRAM read, write failures. In addition, for the high-speed SRAM operating on NTV regime, prevention of potential SRAM access failure was considered. Precisely, we proposed an SRAM monitor, from which we measured a maximum voltage, V_{fail} that causes functional failure on that SRAM monitor. Then, we proposed a novel methodology of inferring SRAM V_{ddmin} on each die from the measured V_{fail} of SRAM monitor on the same die. IR drop and process variation of peripheral circuit as well as process variation on bitcell transistors were considered to mimic the real SRAM operation. Through experiments with industrial SRAM block design, we confirmed our proposed methodology could save leakage power by 10.45%, read energy by 4.99%, and write energy by 5.45% when an SRAM bitcell array of 16KB is used as an SRAM monitor to estimate V_{ddmin} of SRAM blocks of total size of 12.58MB in a chip.

4.2 Chapter 3

In chapter 3, we proposed a new power gating methodology to break the critical (inherently unavoidable) bottleneck in minimizing total size for state retention storage by safely treating a large portion of the self-loop FFs as if they were the same as the flip-flops with no self-loop. Specifically, we developed a novel mechanism of state monitoring on a partial set of self-loop FFs, by which their state retention storage was never needed, enabling a significant saving on the total size of the always-on state retention storage for power gating. In addition, we developed a novel retention storage refinement method that permanently reduce the size of retention storage of retention FFs utilizing state monitoring. Through experiments with benchmark circuits, it was shown that our proposed method was able to reduce total number of retention bits and standby power by 27.12% and 19.41% respectively when at most 2-bit retention FF is used, and 31.73% and 22.34% respectively when at most 3-bit retention FF is used, in comparison with state-of-the-art conventional method.

Bibliography

[1] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 12, pp. 1859–1880, 2005.

[2] T. Gemmeke, M. M. Sabry, J. Stuijt, P. Schuddinck, P. Raghavan, and F. Catthoor, "Memories for ntc," in *Near Threshold Computing*. Springer, 2016, pp. 75–100.

[3] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, "Practical strategies for power-efficient computing technologies," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 215–236, 2010.

[4] S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch, "On characterizing near-threshold sram failures in finfet technology," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 53.

[5] N. N. Mojumder, S. Mukhopadhyay, J.-J. Kim, C.-T. Chuang, and K. Roy, "Design and analysis of a self-repairing sram with on-chip monitor and compensation circuitry," in *26th IEEE VLSI Test Symposium (vts 2008)*. IEEE, 2008, pp. 101–106.

[6] F. Ahmed and L. Milor, "Online measurement of degradation due to bias temperature instability in srams," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 24, no. 6, pp. 2184–2194, 2015.

[7] X. Wang, W. Xu, and C. H. Kim, "Sram read performance degradation under asymmetric nbti and pbti stress: Characterization vehicle and statistical aging data," in *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*. IEEE, 2014, pp. 1–4.

[8] T.-H. Kim, R. Persaud, and C. H. Kim, "Silicon odometer: An on-chip reliability monitor for measuring frequency degradation of digital circuits," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 874–880, 2008.

[9] P. Jain, A. Paul, X. Wang, and C. H. Kim, "A 32nm sram reliability macro for recovery free evaluation of nbti and pbti," in *2012 International Electron Devices Meeting*. IEEE, 2012, pp. 9–7.

[10] X. Wang, C. Lu, and Z. Mao, "Charge recycling 8t sram design for low voltage robust operation," *AEU-International Journal of Electronics and Communications*, vol. 70, no. 1, pp. 25–32, 2016.

[11] X. Wang, Y. Zhang, C. Lu, and Z. Mao, "Power efficient sram design with integrated bit line charge pump," *AEU-International Journal of Electronics and Communications*, vol. 70, no. 10, pp. 1395–1402, 2016.

[12] D. Nayak, D. P. Acharya, P. K. Rout, and U. Nanda, "A novel charge recycle read write assist technique for energy efficient and fast 20 nm 8t-sram array," *Solid-State Electronics*, vol. 148, pp. 43–50, 2018.

[13] D. Nayak, P. K. Rout, S. Sahu, D. P. Acharya, U. Nanda, and D. Tripthy, "A novel indirect read technique based sram with ability to charge recycle and differential read for low power consumption, high stability and performance," *Microelectronics Journal*, p. 104723, 2020.

[14] Y. Shin, J. Seomun, K.-M. Choi, and T. Sakurai, "Power gating: Circuits, design methodologies, and best practice for standard-cell vlsi designs," *ACM Transac-*

tions on Design Automation of Electronic Systems (TODAES), vol. 15, no. 4, pp. 1–37, Oct. 2010.

[15] E. Choi, C. Shin, T. Kim, and Y. Shin, "Power-gating-aware high-level synthesis," in *Proceeding of the 13th international symposium on Low power electronics and design (ISLPED'08)*, 2008, pp. 39–44.

[16] Y.-G. Chen, Y. Shi, K.-Y. Lai, G. Hui, and S.-C. Chang, "Efficient multiple-bit retention register assignment for power gated design: Concept and algorithms," in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, p. 309–316.

[17] M. A. Sheets, "Standby power management architecture for deep-submicron systems," Ph.D. dissertation, UNIVERSITY OF CALIFORNIA, BERKELEY, 2006.

[18] S. Greenberg, J. Rabinowicz, R. Tsechanski, and E. Paperno, "Selective state retention power gating based on gate-level analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1095–1104, 2013.

[19] S. Greenberg, J. Rabinowicz, and E. Manor, "Selective state retention power gating based on formal verification," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 3, pp. 807–815, 2014.

[20] T.-W. Chiang, K.-H. Chang, Y.-T. Liu, and J.-H. R. Jiang, "Scalable sequence-constrained retention register minimization in power gating design," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015.

[21] K.-H. Chang, Y.-T. Liu, C. S. Browy, and C.-L. Huang, "Systems and methods for partial retention synthesis," Jan. 20 2015, uS Patent 8,938,705.

[22] Y.-G. Chen, H. Geng, K.-Y. Lai, Y. Shi, and S.-C. Chang, "Multibit retention registers for power gated designs: Concept, design, and deployment," *IEEE Trans-*

actions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 4, p. 507–518, Apr. 2014.

[23] S.-H. Lin and M. P.-H. Lin, "More effective power-gated circuit optimization with multi-bit retention registers," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, p. 213–217.

[24] G.-G. Fan and M. P.-H. Lin, "State retention for power gated design with non-uniform multi-bit retention latches," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, p. 607–614.

[25] G. Hyun and T. Kim, "Allocation of state retention registers boosting practical applicability to power gated circuits," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019.

[26] ——, "Allocation of multibit retention flip-flops for power gated circuits: Algorithm-design unified approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 5, pp. 892–903, May 2021.

[27] S. Kim and T. Kim, "Minimally allocating always-on state retention storage for supporting power gating circuits," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 482–487.

[28] T. Kim, K. Jeong, T. Kim, and K. Choi, "Sram on-chip monitoring methodology for energy efficient memory operation at near threshold voltage," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 146–151.

[29] T. Kim, K. Jeong, J. Choi, T. Kim, and K. Choi, "Sram on-chip monitoring methodology for high yield and energy efficient memory operation at near threshold voltage," *Integration*, vol. 74, pp. 81–92, 2020.

[30] T.-B. Chan, W.-T. J. Chan, and A. B. Kahng, "On aging-aware signoff for circuits with adaptive voltage scaling," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 10, pp. 2920–2930, 2014.

[31] C. Wann, R. Wong, D. J. Frank, R. Mann, S.-B. Ko, P. Croce, D. Lea, D. Hoyniak, Y.-M. Lee, J. Toomey *et al.*, "Sram cell design for stability methodology," in *IEEE VLSI-TSA International Symposium on VLSI Technology, 2005.(VLSI-TSA-Tech).* IEEE, 2005, pp. 21–22.

[32] R. C. Wong, "Direct sram operation margin computation with random skews of device characteristics," in *Extreme Statistics in Nanoscale Memory Design*. Springer, 2010, pp. 97–136.

[33] T. Kim, G. Hyun, and T. Kim, "Steady state driven power gating for lightening always-on state retention storage," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 79–84.

[34] T. Kim, H. Park, and T. Kim, "Allocation of always-on state retention storage for power gated circuits—steady-state-driven approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 499–511, 2021.

[35] Private communication with DE team in Foundry Business, Samsung Electronics.

[36] A. J. Van De Goor, "Using march tests to test srams," *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 8–14, 1993.

[37] K. Kim, Y. Lim, G. Oh, S. Chung, and B. Lee, "Failure analysis of sram dq fault using bist pattern," in *ISTFA 2018: Proceedings from the 44th International Symposium for Testing and Failure Analysis*. ASM International, 2018, p. 474.

[38] Y. Gu, D. Yan, V. Verma, M. R. Stan, and X. Zhang, "Sram based opportunistic energy efficiency improvement in dual-supply near-threshold processors," in

2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018, pp. 1–6.

[39] I. Parulkar, A. Wood, J. C. Hoe, B. Falsafi, S. V. Adve, J. Torrellas, and S. Mitra, "Opensparc: An open platform for hardware reliability experimentation," in *Fourth Workshop on Silicon Errors in Logic-System Effects (SELSE)*. Citeseer, 2008, pp. 1–6.

[40] W. Choi and J. Park, "Improved perturbation vector generation method for accurate sram yield estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1511–1521, 2016.

[41] L.-C. Lu, "Physical design challenges and innovations to meet power, speed, and area scaling trend," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*. ACM, 2017, pp. 63–63.

[42] B. Wu, J. E. Stine, and M. R. Guthaus, "Fast and area-efficient sram word-line optimization," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[43] C. Albrecht, "Iwls2005 benchmarks," in *IWLS*, 2005. [Online]. Available: https://iwls.org/iwls2005/benchmarks.html

[44] Oliscience, "Opencores," 1999. [Online]. Available: https://opencores.org

[45] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, 2006. [Online]. Available: http://igraph.org

[46] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2019. [Online]. Available: http://www.gurobi.com

[47] R. Chadha and J. Bhasker, *An ASIC Low Power Primer*. Springer New York, 2013.

초 록

칩의 저전력 동작은 중요한 문제이며, 공정이 발전하면서 그 중요성은 점점 커지고 있다. 본 논문은 칩을 구성하는 정적 램(SRAM) 및 로직(logic) 각각에 대해서 저전력으로 동작시키는 방법론을 논한다.

우선, 본 논문에서는 칩을 문턱 전압 근처의 전압(NTV)에서 동작시키고자 할 때 모니터링 회로의 측정을 통해 칩 내의 모든 SRAM 블록에서 동작 실패가 발생하지 않는 최소 동작 전압을 추론하는 방법론을 제안한다. 칩을 NTV 영역에서 동작시키는 것은 에너지 효율성을 증대시킬 수 있는 매우 효과적인 방법 중 하나이지만 SRAM의 경우 동작 실패 때문에 동작 전압을 낮추기 어렵다. 하지만 칩마다 영향을 받는 공정 변이가 다르므로 최소 동작 전압은 칩마다 다르며, 모니터링을 통해 이를 추론해낼 수 있다면 칩별로 SRAM에 서로 다른 전압을 인가해 에너지 효율성을 높일 수 있다. 본 논문에서는 다음과 같은 과정을 통해 이 문제를 해결한다: (1) 디자인 인프라 설계 단계에서는 SRAM의 최소 동작 전압을 추론하고 칩 생산 단계에서는 SRAM 모니터의 측정을 통해 전압을 인가하는 방법론을 제안한다; (2) 칩의 SRAM 비트셀(bitcell)과 주변 회로를 포함한 SRAM 블록들의 공정 변이를 모니터링할 수 있는 SRAM 모니터와 SRAM 모니터에서 모니터링할 대상을 정의한다; (3) SRAM 모니터의 측정값을 이용해 같은 칩에 존재하는 모든 SRAM 블록에서 목표 신뢰수준 내에서 읽기, 쓰기, 및 접근 동작 실패가 발생하지 않는 최소 동작 전압을 추론한다. 벤치마크 회로의 실험 결과는 본 논문에서 제안한 방법을 따라 칩별로 SRAM 블록들의 최소 동작 전압을 다르게 인가할 경우, 기존 방법대로 모든 칩에 동일한 전압을 인가하는 것 대비 수율은 같은 수준으로 유지하면서 SRAM 비트셀

배열의 전력 소모를 감소시킬 수 있음을 보인다.

두 번째로, 본 논문에서는 파워 게이트 회로에서 기존의 보존용 공간 할당 방법들이 지니고 있는 문제를 해결하고 누설 전력 소모를 더 줄일 수 있는 방법론을 제안한다. 기존의 보존용 공간 할당 방법은 멀티플렉서 피드백 루프가 있는 모든 플립플롭에는 무조건 보존용 공간을 할당해야 해야 하기 때문에 다중 비트 보존용 공간의 장점을 충분히 살리지 못하는 문제가 있다. 본 논문에서는 다음과 같은 방법을 통해 보존용 공간을 최소화하는 문제를 해결한다: (1) 보존용 공간 할당 과정에서 멀티플렉서 피드백 루프를 무시할 수 있는 조건을 제시하고, (2) 해당 조건을 이용해 멀티플렉서 피드백 루프가 있는 플립플롭이 많이 존재하는 회로에서 보존용 공간을 최소화한다; (3) 추가로, 플립플롭에 이미 할당된 보존용 공간 중 일부를 제거할 수 있는 조건을 찾고, 이를 이용해 보존용 공간을 더 감소시킨다. 벤치마크 회로의 실험 결과는 본 논문에서 제안한 방법론이 기존의 보존용 공간 할당 방법론보다 더 적은 보존용 공간을 할당하며, 따라서 칩의 면적 및 전력 소모를 감소시킬 수 있음을 보인다.

주요어: 정적 램, 온-칩 모니터링, 공정 변이, 파워 게이팅, 상태 보존, 누설 전력
학번: 2016-20884