



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Ph.D. DISSERTATION

Design Methodology for Cost Effective Clock and Power Gating

비용 효율적인 클럭 및 파워 게이팅 설계 방법론

BY

GYOUNGHWAN HYUN

FEBRUARY 2020

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

Ph.D. DISSERTATION

Design Methodology for Cost Effective Clock and Power Gating

비용 효율적인 클럭 및 파워 게이팅 설계 방법론

BY

GYOUNGHWAN HYUN

FEBRUARY 2020

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

Design Methodology for Cost Effective Clock and Power Gating

비용 효율적인 클럭 및 파워 게이팅 설계 방법론

지도교수 김 태 환
이 논문을 공학박사 학위논문으로 제출함

2019년 12월

서울대학교 대학원

전기정보공학부

현 경 환

현경환의 공학박사 학위 논문을 인준함

2020년 1월

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

Abstract

Low power design is of great importance in modern system-on-chips (SoCs). This dissertation studies on low power design methodologies for saving dynamic and static power consumption. Precisely, we unveil two novel techniques of cost effective low power design.

Firstly, we propose a novel clock gating method for reducing the dynamic power consumption. Flip-flop's input data toggling based clock gating is one of the most commonly used clock gating methods, in which one critical and inherent limitation is the sharp increase of gating logic as more flip-flops are involved in gating. In this dissertation, we propose a new clock gating method to overcome this limitation. Specifically, (1) we analyze the resources of gating logic in the input data toggling based clock gating, from which an ineffectiveness in resource utilization is observed and we propose a new clock gating technique called *flip-flop state driven clock gating* which completely eliminates the essential and expensive component of XOR gates for detecting input toggling of flip-flops; (2) we provide the supporting logic circuitry of our proposed XOR-free clock gating, confirming its safe applicability through a comprehensive timing analysis; (3) we propose, based on the flip-flops' state profile, a clock gating methodology that seamlessly combines our flip-flop state based clock gating with the toggling based clock gating. Through experiments with benchmark circuits, it is confirmed that our clock gating method is very effective in reducing power, which otherwise the toggling based clock gating shall miss the power saving opportunity, while meeting all timing constraints.

Secondly, for reducing the static power consumption, we solve two critical limitations of the conventional approaches to the allocation of state retention storage for power gated circuits. Those are (1) the *long wakeup delay* caused by the senseless use of multi-bit retention flip-flops (MBRFFs) and (2) the inability to optimize retention

flip-flops for the *flip-flops with mux-feedback loop*. It should be noted that the conventional approaches have regarded the long wakeup delay as an inevitable consequence of maximizing the reduction of total storage size for state retention while they have treated the flip-flops with mux-feedback loop (called self-loop flip-flop) as nonoptimizable component, but practically, the self-loop flip-flops synthesized from hardware description language (HDL) code are not far from a small amount and thus, can in no way be negligible. More precisely, for solving (1), we show that the use of MBRFFs with up to two bits, consequently, constraining the wakeup delay to no more than two clock cycles, is enough to maintain the high reduction of total retention storage and for solving (2), we devise a 2-phase retention control mechanism for a pair of flip-flops, one of which has self-loop, by which just a single retention bit can be used to restore state of the two flip-flops, and propose an independent set based algorithm for maximally extracting the non-conflict pairs from circuits. Through experiments with benchmark circuits, it is shown that our proposed method is very effective against reducing the state retention storage and the power consumption compared with the existing best MBRFF allocation while the wakeup delay is strictly limited to two clock cycles.

Keywords: Low power, Clock gating, Power gating, State retention, Retention flip-flop

Student number: 2016-30222

Contents

Abstract	i
Contents	iii
List of Tables	v
List of Figures	vii
1 INTRODUCTION	1
1.1 Clock Gating	1
1.2 Power Gating and State Retention	3
1.3 Multi-bit Retention Registers	4
1.4 Contributions of This Dissertation	6
2 FLIP-FLOP STATE DRIVEN CLOCK GATING: CONCEPT, DESIGN, AND METHODOLOGY	9
2.1 Motivations	9
2.1.1 Toggling based Clock Gating	9
2.1.2 Area and Power by Clock Gating	10
2.2 The Proposed Clock Gating	13
2.2.1 Concept of Flip-flop State Driven Clock Gating	13
2.2.2 Design of Gating Logic Circuitry	17
2.2.3 Integrated Clock Gating Methodology	22

2.2.4	Cost Formulation	23
2.3	Experiments	25
2.3.1	Experimental Setup	25
2.3.2	Experimental Results	26
3	ALGORITHM AND DESIGN OPTIMIZATION OF ALLOCATING MULTI-BIT RETENTION FLIP-FLOPS FOR POWER GATED CIRCUITS	32
3.1	Motivations	32
3.1.1	Flip-flops with Mux-feedback Loop	32
3.1.2	Impact of Wakeup Delay	37
3.2	The Proposed Allocation Algorithm	39
3.3	Design of Multi-Bit Retention Flip-Flop and Multi-Bit Extension . . .	48
3.3.1	Multi-Bit Retention Flip-Flop	48
3.3.2	Multi-Bit Flip-Flop Extension	52
3.4	Experiments	54
3.4.1	Experimental Setup	54
3.4.2	Experimental Results	57
4	CONCLUSIONS	65
4.1	Flip-flop State Driven Clock Gating: Concept, Design, and Methodology	65
4.2	Algorithm and Design Optimization of Allocating Multi-bit Retention Flip-flops for Power Gated Circuits	66
	Abstract (In Korean)	71

List of Tables

2.1	Benchmark circuit information.	27
2.2	Comparison of the clock gating ratio, the number of clock gating cells (i.e., ICGs), the clock tree power P_{clk} , the flip-flop power P_{ff} , the combination logic power P_{combi} including the power by clock gating logic, the total power P_{total} produced (1) by the conventional idle logic driven clock gating (Logic-driven CG), produced (2) by Logic-driven CG followed by the conventional toggling driven clock gating (Toggling-driven CG), and produced (3) by Logic-driven CG followed by our state driven clock gating (State-driven CG), finally applying Toggling-driven CG. [†] Excluding the power consumed by the internal clock inverters in the flip-flops. [‡] Including the power consumed by the internal clock inverters in the flip-flops.	28
2.3	Comparison of total area and cell area used (1) by the conventional idle logic driven clock gating (Logic-driven CG), used (2) by Logic-driven CG followed by the conventional toggling driven clock gating (Toggling-driven CG), and used (3) by Logic-driven CG followed by our state driven clock gating (State-driven CG), finally applying Toggling-driven CG.	30
3.1	Proportion of self-loop FFs in IWLS benchmark circuits.	33

3.2	State retention and restoration for the three types of retention flip-flop. (The notations in parentheses represent the values in the retention storage of the corresponding SBRFFs/MBRFF and cycle times.)	40
3.3	State restoration of flip-flops in a retention pair $p(f_i, f_j)$ where f_i is minimally allocated by a 1st-phase SBRFF. The state of f_i at the second cycle will be set by one of the blue solid and red dotted arrows if f_i is a self-loop FF (i.e., type 2 retention pair) and otherwise (i.e., type 1 retention pair) will be set only by the blue arrow.	42
3.4	Benchmark circuit information.	57
3.5	Comparison of the previous non-uniform MBRFF approaches ([18]) with flow options 1 and 2, and our proposed approach.	58
3.6	Comparison of active power and sleep power consumption used by SBRFF (conventional single bit retention flip-flop allocation for single bit flip-flops), SBR-MBFF (conventional single bit retention flip-flop allocation for multi-bit flip-flops), MBRFF (our multi-bit retention flip-flop allocation for single-bit flip-flops), and MBR-MBFF (our multi-bit retention flip-flop allocation for multi-bit flip-flops).	60
3.7	Comparison of total area and cell area used by SBRFF (conventional single bit retention flip-flop allocation for single bit flip-flops), SBR-MBFF (conventional single bit retention flip-flop allocation for multi-bit flip-flops), MBRFF (our multi-bit retention flip-flop allocation for single-bit flip-flops), and MBR-MBFF (our multi-bit retention flip-flop allocation for multi-bit flip-flops).	62
3.8	Multi-bit information of the MBR-MBFF based designs.	63

List of Figures

1.1	Clock gating.	1
1.2	Toggling-based clock gating.	2
1.3	Structure of a single-bit retention flip-flop. A slave latch for retention is designed with high-Vt transistors and powered by an always-on power supply (VDD).	4
1.4	Structure of a multi-bit retention flip-flop (MBRFF). The k -bit ($k \geq 1$) <i>shift storage element</i> is used to store prior k consecutive states of the <i>regular flip-flop</i> just before entering the sleep mode.	5
2.1	(a) Block-level structure of the input data toggling based clock gating [8], in which the logic blocks added for clock gating are marked with yellow color. (b) The changes of area including XORs (i.e., FFs + XORs + OR-tree + ICG) and excluding XORs (i.e., FFs + OR-tree + ICG) as the flip-flop grouping size k changes. (c) The changes of power consumption including (i.e., FFs + XORs + OR-tree + ICG) and excluding XORs (i.e., FFs + OR-tree + ICG) as the grouping size changes.	11
2.2	Flip-flop distribution with respect to the state-1 probability measured by simulation for a set of IWLS benchmark circuits.	14

2.3	(a) State-0/1 waveforms of flip-flops f_1, f_2, \dots, f_5 . in which f_1, f_2 and f_3 are stuck at state-0 most of time while f_4 and f_5 are stuck at state-1 most of time. (b) State-0 waveforms and ratios in Definition 2 of $\{f_1, f_2\}$ and $\{f_1, f_3\}$ derived from (a). (c) State-1 waveform and ratio in Definition 3 of $\{f_4, f_5\}$ derived from (a).	15
2.4	Circuitry of the proposed state driven clock gating. (a) Block-level structure, in which the logic blocks added for clock gating are marked with yellow color. (b) Clock Disable (CD) block including OR-tree to assert the clock enable for flip-flops that have a high probability of state-0. (c) Clock Disable (CD) block including AND-tree to assert the clock enable for flip-flops that have a high probability of state-1. . . .	18
2.5	Timing waveforms of the signals on (a) CD block of OR-tree type in Fig. 2.4(b) and (b) CD block of AND-tree type in Fig. 2.4.	20
2.6	Timing waveforms by HSPICE simulation for the clock-to-Q delay (T_{CLKQ}) on a flip-flop in the toggling driven clock gating (Q_{tg} : red curves) and a flip-flop in our state driven driven clock gating (Q_{sdcg} : blue curves).	21
2.7	Timing waveforms by HSPICE simulation for the delay of CD blocks in the toggling driven clock gating (red curves) and our state driven clock gating (blue curves). The flip-flop group size is $k = 2, 4, \dots, 16$.	21
2.8	Example of generating a flip-flop group S_i for clock gating. The input flip-flop set \mathcal{F} is $\{f_1, f_2, f_3, f_4, f_5\}$ and f_1 is picked as a seed in Step 2.2.	24
2.9	Tested clock gating flows. (a) Logic-driven CG only. (b) Logic-driven CG followed by Toggling-driven CG. (c) Logic-driven CG followed by our State-driven CG and finally Toggling-driven CG.	26

2.10	Layouts for WB_DMA. The colored rectangles represent flip-flops: un-gated (white), gated by Logic-driven CG (yellow), gated by Toggling-driven CG (orange), gated by our State-driven CG (blue). (a) Layout produced by Logic-driven CG. (b) Layout produced by Toggling-driven CG. (c) Layout produced by our State-driven CG.	31
3.1	Synthesis of flip-flops with mux-feedback loop. (a) a Verilog HDL code (b) a synthesized structure for the code in (a).	33
3.2	Two flow options used by the conventional approaches of allocating retention flip-flops to circuits with self-loop FFs.	35
3.3	(a) A better allocation of retention bits over that in Fig. 3.2. (b) The state restoration of the wakeup sequence for the retention bit allocation in (a). ($\{\cdot\cdot\cdot\}$ indicates the state(s), to be used when waking up, stored in the retention bit(s) and d_i^j indicates the state of flip-flop f_i at time $t_j, j = 1, 2$.)	36
3.4	Average changes of the reduction rate of the total size of retention storage by <i>Flow options</i> 1 and 2 using [18] as the wakeup latency changes for IWLS benchmark circuits in Table 3.1.	38
3.5	Retention pair $p(f_i, f_j)$	41
3.6	Retention pairs of type 1 (in red color) and type 2 (in blue color) in a flip-flop dependency graph G	41
3.7	Conditions of conflict between retention pairs. (a) <i>Condition 1</i> (b) <i>Condition 2</i> (c) <i>Condition 3</i>	43
3.8	Minimally required retention bits for the flip-flops in a retention pair $p(f_1, f_2)$ and its driving and driven flip-flops.	44
3.9	Two retention pairs that cause conflicts due to each of <i>conditions 1, 2, and 3</i> , and the minimally required retention bits for the green pair when the yellow pair is allocated by the minimally required retention bits.	45

3.10	Tie-breaking based on the cost formulation in $Eq.(3.1)$	47
3.11	Conceptual schematic of the pulsed latch array of the k -bit retention register in [14]. High-Vt transistors powered by an always-on power supply are used for the retention latches (in green color).	48
3.12	Schematic of our 2-bit MBRFF. The always-on supply region in green color consists of two retention latches and two input inverters for control pins $NRET$ and $SHIFT$	49
3.13	Waveforms of the 2-bit shift register using two control signals.	50
3.14	Spice simulation waveforms of a 2-bit MBRFF.	51
3.15	Comparison of the internal structures of the 1-bit master-slave based flip-flop and the 2-bit master-slave flip-flop.	52
3.16	Schematic of a 2-bit/2-bit MBR-MBFF. The cells in light blue color are shared between two master flip-flops $FF1$ and $FF2$	53
3.17	Comparison of power and area of a MBR-MBFF between the conventional design and the proposed design. The size of the retention storage is 2-bit.	55
3.18	Design flow of the proposed MBR-MBFF approach.	56
3.19	Sleep power breakdown of our MBRFF approach without MBFF extension (MBRFF) and MBRFF approach with MBFF extension (MBR-MBFF).	63
3.20	Comparison of always-on cell area of our MBRFF approach without MBFF extension (MBRFF) and MBRFF approach with MBFF extension (MBR-MBFF).	64

Chapter 1

INTRODUCTION

1.1 Clock Gating

In synchronous digital system, a large portion of dynamic power is consumed by the clock signal, taking over 40% of total power consumption in the entire systems [1, 2]. As a vehicle to reduce the dynamic power consumption, a technique by gating clock signals has been known to be one of the most powerful techniques. As shown in Fig. 1.1, *Clock gating* saves dynamic power by shutting off a subtree of clock network during idle state of the driven logic blocks or by disabling the clock signals to a group or groups of flip-flops during their untoggling states.

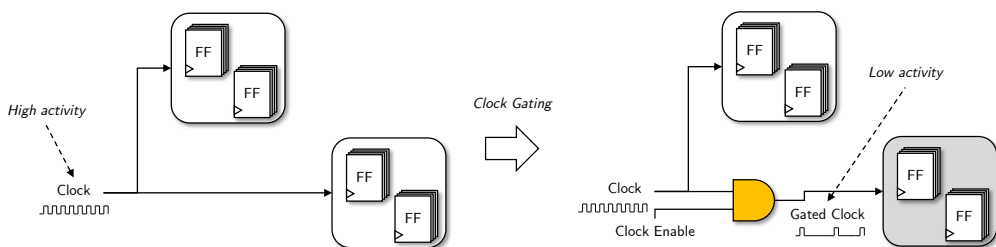


Figure 1.1: Clock gating.

For the situation where designers have a prior knowledge of or can explicitly or implicitly extract simple logic conditions for some blocks in a circuit that can be safely

in an idle state, the idle state based clock gating can be applied to the part of clock network that drives the blocks. Note that the clock enable/disable logic conditions are generally extracted in the system or RTL design stage since the inter-dependency of the execution among the logic sub-blocks is easily identifiable in those stages. However, for the case where it is not easy or apparent to identify simple logic for clock gating or there is no subcircuit block of reasonable size for clock gating, the toggling based clocking gating can be the best alternative as shown in Fig. 1.2.

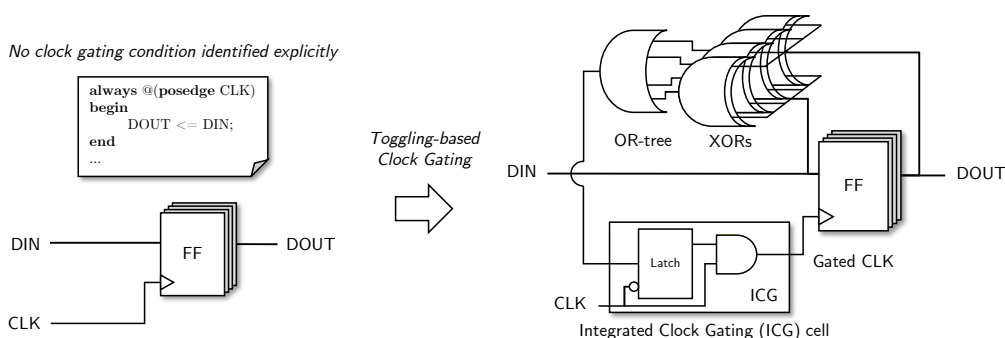


Figure 1.2: Toggling-based clock gating.

One important consideration in applying the toggling based gating is that since the toggling based clock gating is a flip-flop level fine-grained gating, the sharp increase of the supplemental logic required for detecting input toggling/untoggling of every flip-flop for clock gating as well as generating enabling/disabling clock signal is a big power burden on the circuits. Even so, there are many design cases available, to which the application of toggling based clock gating is more effective in reducing power over that of the idle state based clock gating [3], for instance, being reported for some controller design that its idle based gating to RTL blocks reduces the power dissipation by 23~27% while its toggling based gating to flip-flops reduces the power more than double [4].

The traditional toggling based clock gating can be implemented at various design levels, from architecture-level to cell-level (e.g., [5, 6, 7, 8]). The main focuses of most of the existing methods of the clock gating are aggressively selecting candidate flip-

flops for clock gating and grouping the flip-flops for gating to (partially) share gating logic at the expense of degradation of power saving (e.g., [9, 10]).

1.2 Power Gating and State Retention

As the semiconductor process technology shrinks, the impact of leakage power on power consumption has been significantly increased and it has been extremely important to reduce leakage power in modern SoCs. Power gating has been widely used and has become one of the most popular design techniques to reduce the leakage power consumption, thus to extend the battery lifetime in industrial products [11]. However when the circuit goes the sleep mode by power gating, the state of flip-flops may be lost. Therefore, a proper state restoration scheme is required to backup the state when the circuit wakes up [12].

One commonly adopted restoration methodology is replacing each regular flip-flop with a unique retention flip-flop that is able to perform the additional role of retaining a prior state during the sleep mode [13]. While there are several variants on the implementation of retention flip-flop, it is basically composed of two components: one is a master flip-flop and the other is a slave latch. The master is designed with low- V_t transistors for fast switching during the active mode while the slave retains the prior state (i.e., 0 or 1) of the master flip-flop during the sleep mode and is designed with high- V_t transistors to save leakage power during the sleep mode as shown in Fig. 1.3. According to [14], applying this single-bit retention flip-flops (SBRFFs) causes in general 20% area overhead over the regular flip-flops. Therefore, it is quite important to minimize the total storage size of retention flip-flops (equivalently, the total number of bits to be used to retain circuit state during the sleep mode) to be deployed.

To reduce total storage size of retention flip-flops, the selective state retention power gating (SSRPG) techniques have been proposed [15, 16]. This techniques are basically based on the assumption that relatively small essential flip-flops are required

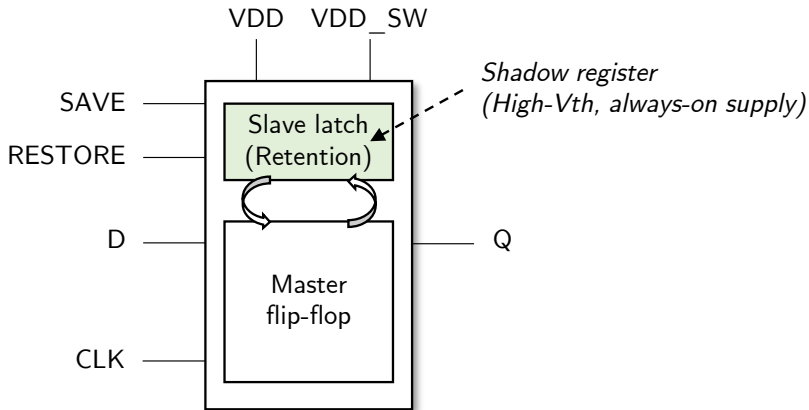


Figure 1.3: Structure of a single-bit retention flip-flop. A slave latch for retention is designed with high-Vt transistors and powered by an always-on power supply (VDD).

to restore the state of entire flip-flops of the designs. However, these approaches are based on an exhaustive simulation or a formal analysis and require a knowledge of operations of the design, which are impractical in general.

1.3 Multi-bit Retention Registers

On the other hand, Chen *et al.* [14, 17] proposed to use multi-bit retention flip-flops (MBRFFs) whose internal structure is shown in Fig. 1.4.

While an SBRFF has a single-bit storage element for retaining a single-bit state, an MBRFF has a k -bit ($k \geq 1$) shift storage element for retaining k -bit states of the regular (master) flip-flop. Before a circuit enters the sleep mode, consecutive prior states of the master flip-flop are stored in the k -bit storage element sequentially and then the states stored are retained during the sleep mode. When the circuit wakes up, the stored state data will be shifted out to the master flip-flop for k consecutive cycles, propagating them to the neighboring flip-flops through the connected combinational logic. Chen *et al.* [14, 17] attempted to solve the problem of allocating k -bit *uniform* MBRFFs with the objective of minimizing the number of flip-flops to be replaced with

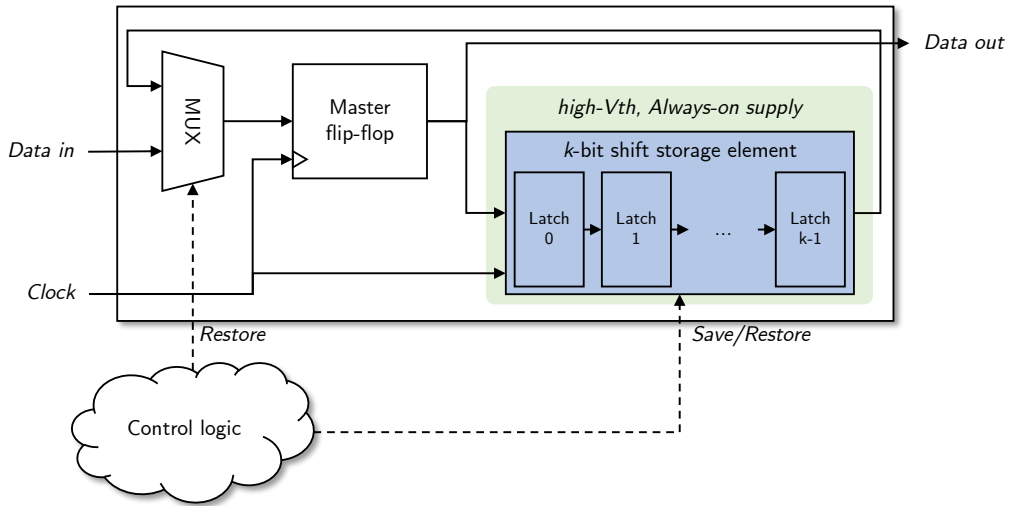


Figure 1.4: Structure of a multi-bit retention flip-flop (MBRFF). The k -bit ($k \geq 1$) *shift storage element* is used to store prior k consecutive states of the *regular flip-flop* just before entering the sleep mode.

k -bit retention flip-flops. They showed that even though the transition latency between sleep and active modes takes $k - 1$ more cycles over the case of applying SBRFFs, the use of MBRFFs considerably decreases total storage size of the retention registers used in power gating, thereby further reducing the power consumption. Based on the MBRFF technique, recently Fan *et al.* [18] proposed an integer linear programming (ILP) based incremental greedy method that is able to allocate *non-uniform* MBRFFs of various sizes of retention storage under the constraint of maximum wake-up latency bound. The objective function to be minimized in each iteration of the method is the amount of storage size required for retaining the state of flip-flops in the ready list. However, both uniform and non-uniform MBRFF replacement techniques have two critical limitations. Those are (1) the *long wakeup delay* caused by the senseless use of MBRFFs and (2) the inability to optimize retention flip-flops for the *flip-flops with mux-feedback loop*. Note that the prior approaches have regarded the long wakeup delay as an inevitable consequence of maximizing the reduction of total storage size for

state retention while they have treated the flip-flops with self-loop as nonoptimizable component, but the flip-flops with self-loop synthesized from HDL code are prevalent in practice and thus, can in no way be negligible.

1.4 Contributions of This Dissertation

In this dissertation, Chapter 2 and Chapter 3 present novel low power techniques for reducing dynamic power consumption and static power consumption respectively.¹

In Chapter 2, we propose a new clock gating technique by addressing (1) how the supplemental (non-sharing) logic for toggling based clock gating can be elegantly reduced or completely eliminated while maximally reaping the benefit of power saving by the clock gating and (2) how the input toggling based clock gating flow is seamlessly integrated into our clock gating flow to maximally exploit the synergy effect on power saving.

The contributions of this work are summarized as:

1. We analyze the resources of gating logic in input data toggling based clock gating, from which we observe an ineffectiveness in resource utilization and propose a new clock gating technique called *flip-flop state driven clock gating*, which completely eliminates the essential component of XOR gates for detecting input toggling of flip-flops.
2. We provide the *XOR-free gating logic circuitry supporting our flip-flop state driven clock gating*, confirming its safe applicability through a comprehensive timing analysis.
3. We propose, based on the flip-flops' state profile, a *clock gating methodology* that seamlessly integrates our flip-flop state driven clock gating with the application flow of the conventional toggling driven clock gating.

¹Preliminary versions of this work were presented in [19] and [20]

4. A set of experiments with benchmark circuits is performed to assess how much our new clock gating methodology is effective in saving power without timing violation.² In summary, ours is able to achieve on average 7.59% more power saving over the input data toggling based clock gating.

In Chapter 3, we propose a new allocation algorithm of multi-bit retention registers for power gated circuits to overcome the limitations of conventional approaches: (1) the long wakeup delay; (2) the inability to optimize the flip-flops with mux-feedback loop. Precisely, for solving (1), we show that the use of MBRFFs with up to two bits, consequently, constraining the wakeup delay to no more than two clock cycles, is enough to maintain the high reduction of total retention storage and for solving (2), we devise a 2-phase retention control mechanism for a pair of flip-flops, one of which has self-loop, by which just a single retention bit can be used to restore state of the two flip-flops, and propose an independent set based algorithm for maximally extracting the non-conflict pairs from circuits.

The contributions of this work are summarized as:

1. Unlike the conventional approaches, which have tried to reduce the retention storage at the expense of (long) wakeup delay, we develop an effective algorithm for *MBRFF allocation that is specialized to the wakeup delay constrained to two clock cycles*.
2. While the conventional approaches have invariably taken into no consideration of the retention storage optimization for the flip-flops with self-loop together with their neighbor flip-flops, we propose *a 2-phase retention control scheme, so that just a single retention bit can be used to restore state of a flip-flop with self-loop and one of its neighbors*.

²Our proposed clock gating in this work is carried out after the completion of cell placement, from which the location information of the flip-flops is available. However, if the wire delay perturbation is not a serious concern, our clock gating is also applicable to the logic synthesis stage.

3. Based on the proposed control scheme, we formulate the retention storage reduction problem into an *independent set based problem* and we develop an *effective heuristic* that maximally extracts non-conflict pairs of flip-flops from circuits.
4. We propose a new design of a multi-bit retention flip-flop and its multi-bit flip-flop extension to reduce the area and power overhead of multi-bit retention flip-flops and control paths for state retention powered by an always-on power supply.
5. Experimental data are provided to show how much our proposed approach reduces the total retention storage size for practical designs while the wakeup delay is constrained to up to two cycles. In short, our approach is able to use the retention storage by 9.8% less on average over that used by the state-of-the-art MBRFF method.

The rest of the paper is organized as follows. Chapter 2 describes the concept, design, and methodology of our flip-flop state driven clock gating. Then, Chapter 3 describes a new allocation algorithm and design of multi-bit retention flip-flops for power gated circuits. Finally, Chapter 4 presents the conclusions of this dissertation.

Chapter 2

FLIP-FLOP STATE DRIVEN CLOCK GATING: CONCEPT, DESIGN, AND METHODOLOGY

2.1 Motivations

2.1.1 Toggling based Clock Gating

Flip-flop's input data toggling based clock gating shuts off the clock signal to a flip-flop when the state of the flip-flop is not subject to change at the next clock cycle. A block-level circuit structure supporting the clock gating for a group of k flip-flops ($R = \{f_1, f_2, \dots, f_k\}$) is shown in Fig. 2.1(a) [8], in which the newly added logic blocks (i.e., *Clock Disable* (CD) and *Integrated Clock Gating* (ICG) [21]) to the original circuit are marked with yellow color.

The logic operations of CD and ICG are:

- CD (= XORs + OR-tree): Boolean equation for CD can be expressed as:

$$g = (D_1 \oplus Q_1) + (D_2 \oplus Q_2) + \dots + (D_k \oplus Q_k) \quad (2.1)$$

Thus, the implementation requires k 2-input XOR gates, one for each flip-flop, and an OR-tree consisting of $k - 1$ 2-input OR gates. The equation indicates that if g is 1, there is at least one flip-flop in R that will change its state at the next

clock cycle. Thus, the condition of disabling clock signal to *all* k flip-flops in R at the next clock cycle is \bar{g} .

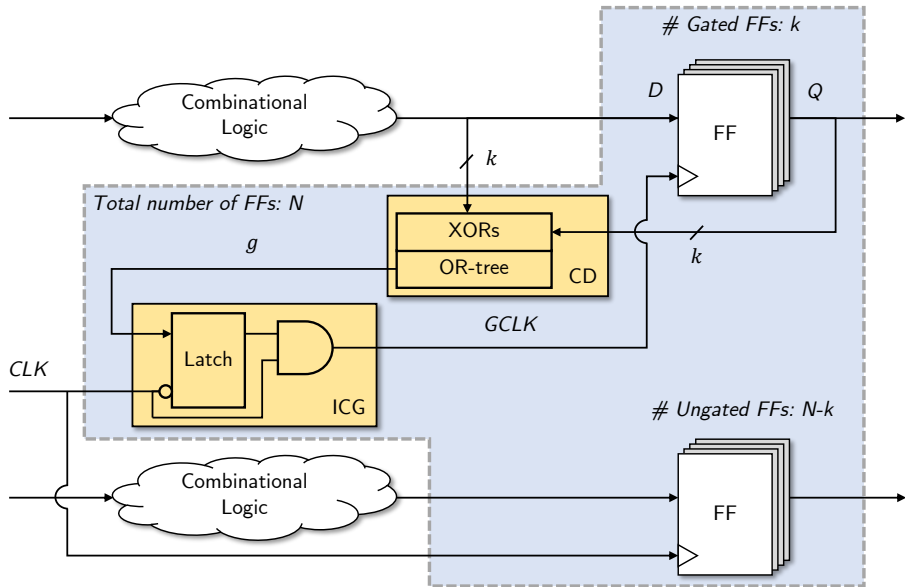
- ICG (= Latch + AND): It receives the logic value of g from CD and decides if the clock signal is to be disabled or not while synchronizing it to the rising edge of CLK .

The effectiveness of the toggling based clock gating on reducing power closely relies on the number of occurrences of the clock cycles at which g is false (i.e., all flip-flops simultaneously untoggling).

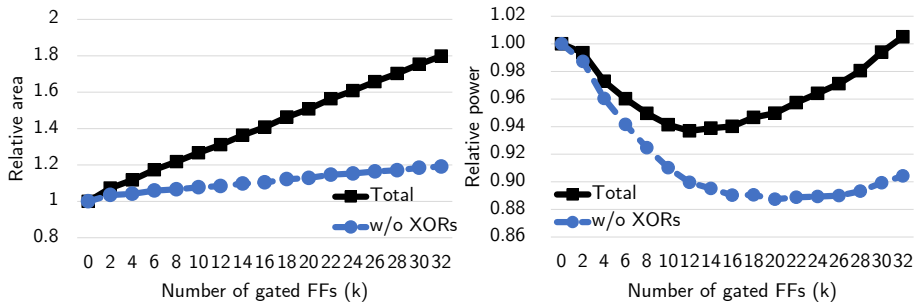
One strong evidence of the usefulness of the toggling based clock gating can be found in [3], which measured statistical toggling data of more than 22,000 flip-flops in a DSP core and found that on average 95% of clock signals to the flip-flops in a clock cycle was untoggled. On the other side, this evidence has cast a strong doubt towards us on “*is it really necessary to allocate an expensive 2-input XOR for every flip-flop in toggling driven clock gating?*” for the circuits with such extremely biased toggling distribution. That is, there is an important factor that has not been carefully addressed by the existing works on the toggling based clock gating. Precisely, the resulting clock gating logic components (XORs and OR-tree in CD and a latch in ICG) incur area overhead, which causes nontrivial impact on power. The following subsection describes in-depth analyses of the gating overhead.

2.1.2 Area and Power by Clock Gating

We performed power analysis using Synopsys *PrimeTime-PX* to the blocks enclosed by the dotted line in Fig. 2.1(a), varying the flip-flop group size $k = 0, 2, 4, \dots, 32$ for clock gating, assuming that the circuit has a total of 32 ($= N$) flip-flops, and each flip-flop toggles independently with toggling probability of 0.05. The curves in Figs. 2.1(b) and 2.1(c) show the changes of the area and the power consumption of the blocks



(a) Toggling based clock gating.



(b) Gating area including flip-flops w/ and (c) Power consumption w/ and w/o XOR gates.

Figure 2.1: (a) Block-level structure of the input data toggling based clock gating [8], in which the logic blocks added for clock gating are marked with yellow color. (b) The changes of area including XORs (i.e., FFs + XORs + OR-tree + ICG) and excluding XORs (i.e., FFs + OR-tree + ICG) as the flip-flop grouping size k changes. (c) The changes of power consumption including (i.e., FFs + XORs + OR-tree + ICG) and excluding XORs (i.e., FFs + OR-tree + ICG) as the grouping size changes.

enclosed by the dotted line in Fig. 2.1(a) as the grouping size k of flip-flops varies.

Observation 1: The total quantity, $Area(CG)$, of area overhead incurred by clock gating for a group of k flip-flops is:

$$Area(CG) = k \cdot Area(XOR) + (k - 1) \cdot Area(OR) + Area(ICG) \quad (2.2)$$

$$Area(CG + FFs) = \begin{cases} N \cdot Area(FF) + Area(CG) & \text{if } k > 0 \\ N \cdot Area(FF) & \text{if } k = 0 \end{cases} \quad (2.3)$$

The black curve in Fig. 2.1(b) shows the changes of the value of $Area(CG+FFs)$ in terms of the cell size as the value of k changes while the blue curve in Fig. 2.1(b) shows the changes of the value of $Area(CG + FFs) - k \cdot Area(XOR)$. The big gap between the two curves in Fig. 2.1(b) indicates that *XORs occupy a significant portion of $Area(CG + FFs)$ as k increases.*

Observation 2: The numbers on the black power curve in Fig. 2.1(c) are obtained by summing the power consumption of (1) the clock gating logic (i.e., XORs, OR-tree, and ICG), (2) the k flip-flops selected for clock gating, and (3) the remaining $32-k$ flip-flops. (The details of the calculation of their power consumption will be described in Sec. 3.10.) The power curve indicates that the power consumption gradually decreases as k increases, and then grows back. It means that the total *power saving heavily depends on the value of group size k* , which in turn closely relies on the joint toggling probabilities among the grouped flip-flops.

On the other hand, the blue curve in Fig. 2.1(c) indicates the changes of power consumption by (1), (2) and, (3) excluding that by XORs. The comparison of the two curves implies that *XORs occupy a considerable portion of power consumption.* (Note that if XORs were all removed, clearly the timing of clock gating would also be shortened by the amount of 2-input XOR delay, though it's a small constant saving.)

Based on the observations, our strategy targets two directions: (i) investigating a new clock gating technique that does not need the expensive XORs required for detecting input toggling while maximally reaping the benefit of toggling driven clock

gating (Sec. 2.2) and (ii) seamlessly integrating our clock gating flow into the existing flow of toggling driven clock gating (Sec. 2.2.3).

2.2 The Proposed Clock Gating

2.2.1 Concept of Flip-flop State Driven Clock Gating

Our idea of an XOR-free clock gating is based on the following observation.

Observation 3: We performed circuit simulation as does in observations 1 and 2 to a set of IWLS benchmark circuits [22] for a sufficiently long period of time and measured the probability of each flip-flop being state-1. Fig. 2.2 shows the flip-flop distribution with respect to the state-1 probability. For example, the length of the bars on the x -coordinate $[p_x, p_x+0.05]$ represents the number of flip-flops whose probability of being state-1 is in between p_x and $p_x+0.05$. The shape of flip-flop distribution in Fig. 2.2 clearly indicates that a large portion ($> 39\%$) of the flip-flops in circuits tends to be stuck at state-0 or state-1 most of the simulation time, which means the input toggling on those flip-flops rarely happens. Thus, the role (i.e., detecting toggling) of XORs is actually not fully utilized on those flip-flops, although the flip-flops shall in fact very likely be selected for toggling driven clock gating.

Observation 3 motivates us to elaborate the toggling based clock gating by devising a new strategy of clock gating.

Definition 1. (State driven clock gating) *For a flip-flop which is classified as stuck at state-0 (state-1) most of time, disable the clock signal to the flip-flop when the coming input at the next clock cycle is 0 (1).*

For a flip-flop which is classified as unstuck at state-0 or state-1 most of time, we will apply the toggling driven clock gating to the flip-flop. (The selection of flip-flops for applying state driven clock gating and the integration into the toggling based clock gating will be described in Sec. 2.2.3.) For example, Fig. 2.3(a) shows the state waveforms of flip-flops f_1, f_2, \dots, f_5 . Flip-flops f_1, f_2 , and f_3 are classified as stuck at

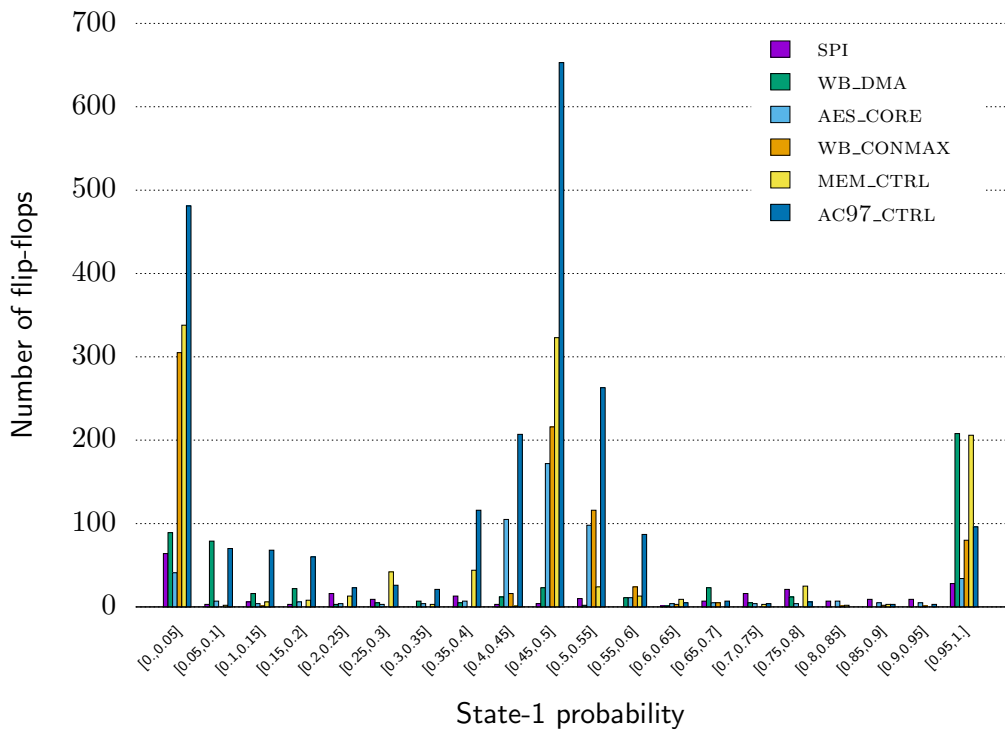


Figure 2.2: Flip-flop distribution with respect to the state-1 probability measured by simulation for a set of IWLS benchmark circuits.

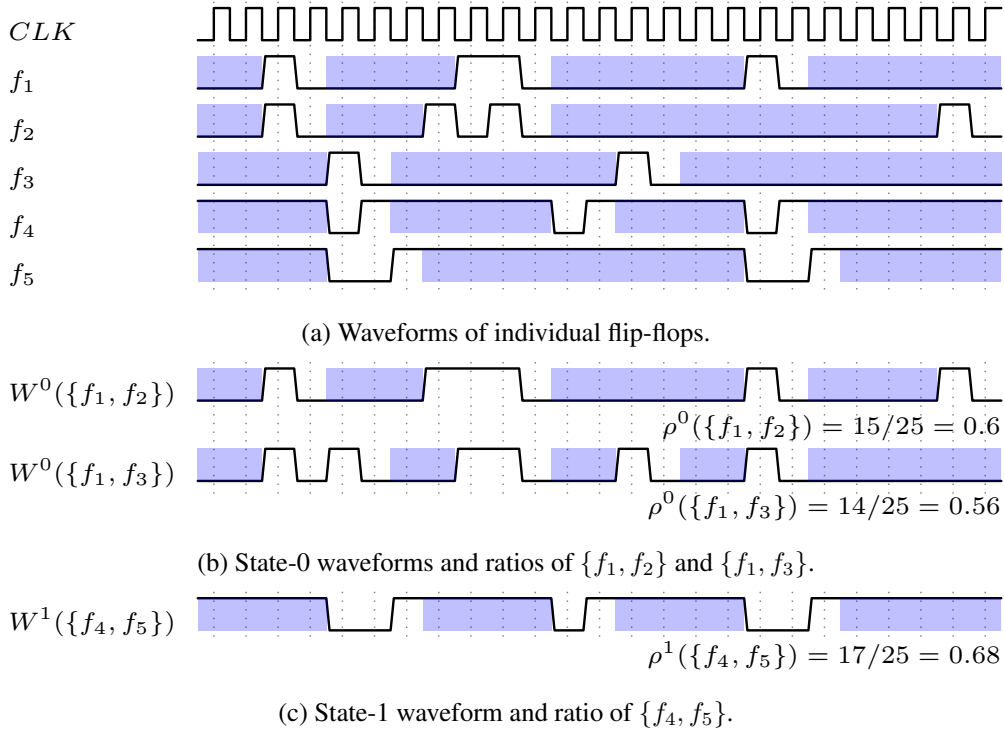


Figure 2.3: (a) State-0/1 waveforms of flip-flops f_1, f_2, \dots, f_5 . in which f_1, f_2 and f_3 are stuck at state-0 most of time while f_4 and f_5 are stuck at state-1 most of time. (b) State-0 waveforms and ratios in Definition 2 of $\{f_1, f_2\}$ and $\{f_1, f_3\}$ derived from (a). (c) State-1 waveform and ratio in Definition 3 of $\{f_4, f_5\}$ derived from (a).

state-0 and f_4 and f_5 as state-1, in which the blue intervals are the time on which our state driven clock gating safely disables the clock signal. Thus, as long as f_1 , f_2 , and f_3 (f_4 and f_5) are in state-0 (state-1), their clock signal can be disabled with no help of XOR. However, since applying clock gating to the flip-flops individually will still increase the gating overhead, because of no sharing of ICG, a careful grouping of flip-flops for clock gating is essential.

Definition 2. ($W^0(S, t), \rho^0(S)$) Let $w(f, t)$ be the state value on the simulation waveform of flip-flop f at simulation time t , $0 \leq t \leq T_{max}$. Then, for a set S of flip-flops and simulation waveforms of the flip-flops in S , $W^0(S, t)$ called **state-0 waveform of S** is defined to a waveform which satisfies, for every t in $[0, t_{max}]$,

1. $W^0(S, t) = 0$, if $w(f_i, t) = 0$, for all $f_i \in S$.
2. $W^0(S, t) = 1$, otherwise.

Then, $\rho^0(S)$ called **state-0 ratio** is defined to the portion of the simulation times at which $W(S, t) = 0$, $0 \leq t \leq T_{max}$.

For example, Fig. 2.3(b) shows the generation of state-0 waveforms of $S1 = \{f_1, f_2\}$ and $S2 = \{f_1, f_3\}$ from the waveforms of f_1 , f_2 , and f_3 in Fig. 2.3(a), and the values of $\rho^0(\cdot)$. The blue bars in Fig. 2.3(b) indicate the clock cycle times at which the clock signals to the flip-flops in $S1$ and $S2$ are disabled if our state driven clock gating were applied to flip-flop groupings $S1$ and $S2$. Thus, for the two grouping $S1$ and $S2$ of flip-flops, containing the same number of flip-flops, since $S1$ has the higher $\rho^0(\cdot)$ value than that of $S2$, selecting grouping $S1$ rather than $S2$ is a better choice for applying our state driven clock gating.

Definition 3. ($W^1(S, t), \rho^1(S)$) $W^1(S, t)$ called **state-1 waveform of S** is defined to be a waveform that satisfies, for every t in $[0, t_{max}]$,

1. $W^1(S, t) = 1$, if $w(f_i, t) = 1$, for all $f_i \in S$.
2. $W^1(S, t) = 0$, otherwise.

Then, $\rho^1(S)$ called **state-1 ratio** is defined to the portion of the simulation times at which $W^1(S, t) = 1$.

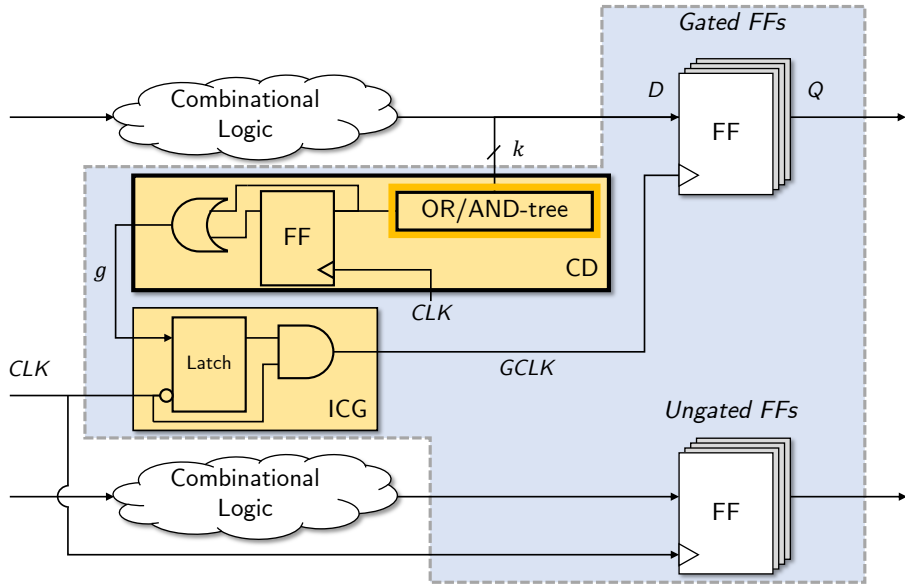
For example, Fig. 2.3(c) shows the generation of state-1 waveform for $S3 = \{f_4, f_5\}$ from the waveforms of f_4 , and f_5 in Fig. 2.3(a), and the value of $\rho^1(S3)$. Note that the generation of $W^0(S)$ and $W^1(S)$ can be performed incrementally as S gradually increases, thus, the time complexity is bounded by $O(|S| \cdot T_{max})$.

2.2.2 Design of Gating Logic Circuitry

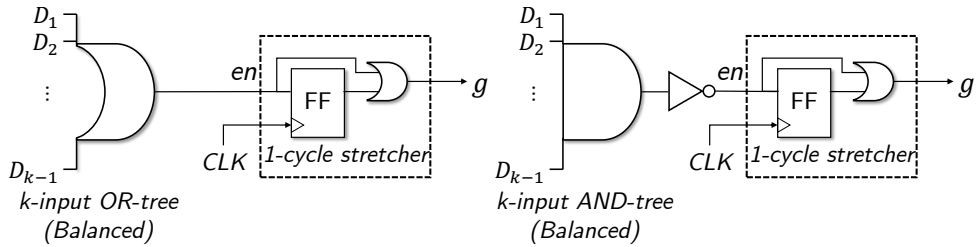
Fig. 2.4(a) shows the block-level structure of our proposed state driven gating and Figs. 2.4(b) and (c) show the internal structure of *two types* (i.e., OR-tree type or AND-tree type) of *Clock Disable* (CD) block. Compared with the structure of toggling driven clock gating shown in Fig. 2.1, ours *never include the most expensive XORs* in CD, one for every gated flip-flop in toggling driven clock gating. The supporting new or updated logic circuitry is the followings:

1. Unlike the toggling driven clock gating which consistently allocates an OR-tree for a group of flip-flops, ours allocates an *AND-tree*, shown in Fig. 2.4(c), for a flip-flop group that has a high probability of state-1 while allocating an *OR-tree*, shown in Fig. 2.4(b), for a flip-flop group that has a high probability of state-0.
2. Our clock gating includes a *signal stretcher*¹ composed of a *flip-flop* and an *2-input OR* gate, for each group of flip-flops as shown in Figs. 2.4(b) and (c).
3. An inverter is needed in the AND-tree type CD block before the signal stretcher because it is assumed that the clock enable signal (*en*) is active high.

¹Note that just one *distinct signal stretcher* is required for every group of gated flip-flops in the state driven clock gating while *one distinct XOR* is required for every gated flip-flop in the toggling driven clock gating.



(a) Block-level structure of our state driven clock gating.



(b) CD block of OR-tree type.

(c) CD block of AND-tree type.

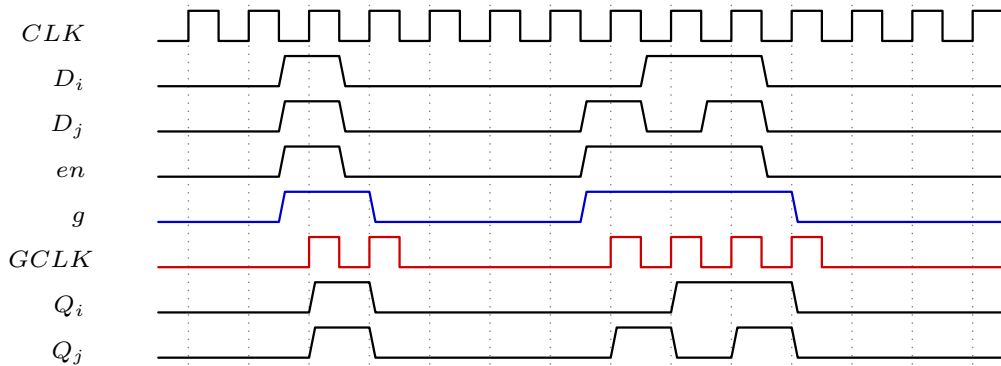
Figure 2.4: Circuitry of the proposed state driven clock gating. (a) Block-level structure, in which the logic blocks added for clock gating are marked with yellow color. (b) Clock Disable (CD) block including OR-tree to assert the clock enable for flip-flops that have a high probability of state-0. (c) Clock Disable (CD) block including AND-tree to assert the clock enable for flip-flops that have a high probability of state-1.

The role of *signal stretcher* to expand clock enable signal en from CD block and guarantee one more clock toggling at the next cycle after en is de-asserted. For example, in the case where a group of flip-flops has a high probability of state-0, the OR-tree type is used and en is asserted to high when at least one of the input data (D) is changed to state-1 and the clock is enabled. When every value of the input data (D) goes to state-0, en is de-asserted but one more clock toggling *is required to capture the last state transition* from state-1 to state-0.

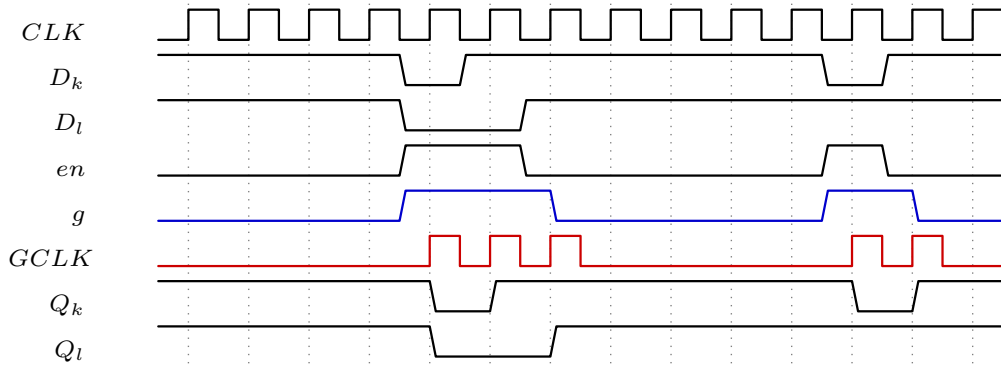
Fig. 2.5 demonstrates the timing waveforms of the signals on CD blocks in Figs. 2.4(b) and (c), in which the enable signal for ICG (g) is stretched by one cycle compared with en and the gated clock ($GCLK$) is toggling during consecutive cycles during g is asserted.

We performed HSPICE simulation to extract the actual timing delay on the gated flip-flops and CD blocks in Fig. 2.1 and Fig. 2.4. Fig. 2.6 compares the timing waveforms for the clock-to-Q (T_{CLKQ}) on a gated flip-flop in the toggling driven clock gating and a gated flip-flop in the proposed state driven clock gating. Compared with the toggling driven clock gating which has an XOR gate on the output of the flip-flop (Q_{tg} , the red curves in Fig. 2.6), the clock-to-Q delay (Q_{sdcg} , the blue curves in Fig. 2.6) of the flip-flop in the state driven clock gating is decreased by 6.1 ps for rise transition and 6.23 ps for fall transition. Contrary to the conventional toggling based gating, the proposed clock gating requires no additional output load on the output of the gated flip-flop, thereby not increasing the clock-to-Q delay in comparison with a regular ungated flip-flop.

The delay of CD block varies depending on the flip-flop grouping size k . Fig. 2.7 shows the timing waveforms of the outputs of CD block in Fig. 2.1 and Fig. 2.4 varying $k = 2, 4, \dots, 16$. For the simulation, we used 2-, 3-, and 4-input OR gates and 2-, 3-input XOR gates which are available in the cell library. In comparison with the delay of CD block (red curves) in the toggling based gating, the delay (blue curves) in the state driven clock gating is reduced by 8.11% \sim 29.79%.



(a) CD block of OR-tree type.



(b) CD block of AND-tree type.

Figure 2.5: Timing waveforms of the signals on (a) CD block of OR-tree type in Fig. 2.4(b) and (b) CD block of AND-tree type in Fig. 2.4.

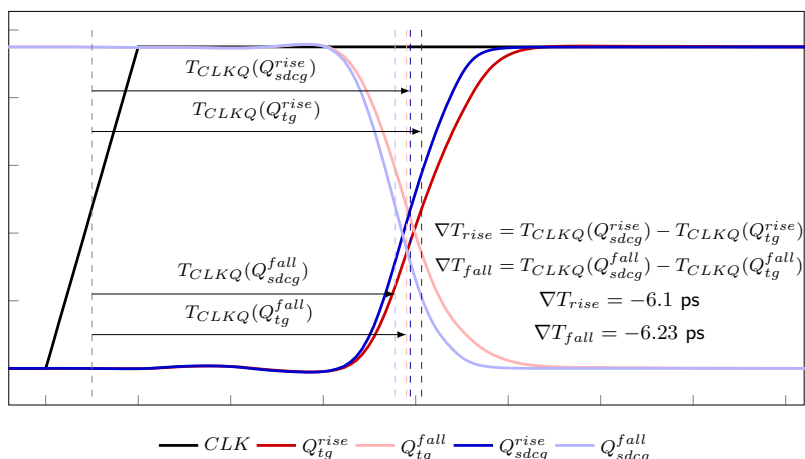


Figure 2.6: Timing waveforms by HSPICE simulation for the clock-to-Q delay (T_{CLKQ}) on a flip-flop in the toggling driven clock gating (Q_{tg} : red curves) and a flip-flop in our state driven driven clock gating (Q_{sdcg} : blue curves).

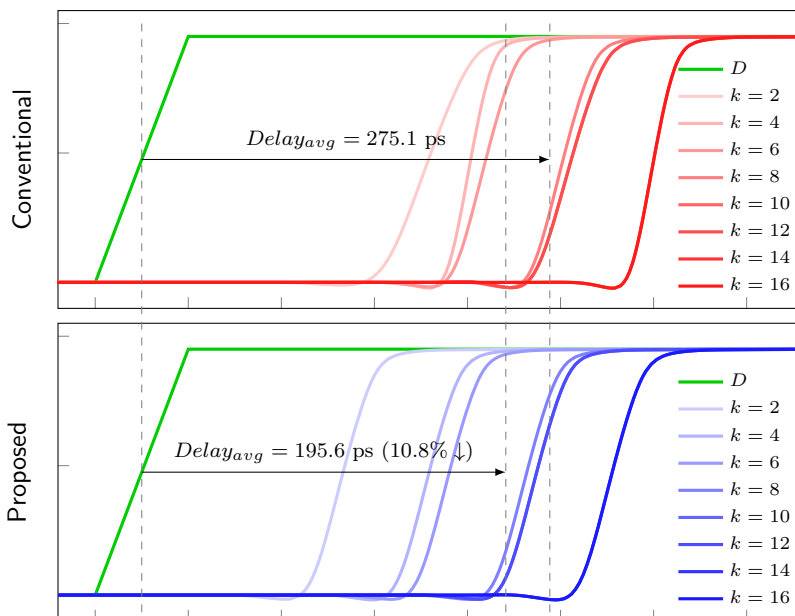


Figure 2.7: Timing waveforms by HSPICE simulation for the delay of CD blocks in the toggling driven clock gating (red curves) and our state driven clock gating (blue curves). The flip-flop group size is $k = 2, 4, \dots, 16$.

The timing analyses shown in Fig. 2.6 and Fig. 2.7 clearly ensure that the application of our state drive clock gating is more safe in terms of timing as well as more economical in terms of area/power over the toggling driven clock gating.

2.2.3 Integrated Clock Gating Methodology

Our clock gating methodology applies the proposed state driven clock gating to two disjoint subsets of flip-flops which have a high probability of state-0 or state-1, followed by applying the conventional toggling driven clock gating to the rest of flip-flops. Precisely, we carry out the clock gating methodology to an input circuit \mathcal{C} in the following three steps.

• **Step 1:** For the circuit \mathcal{C} , the flip-flop set \mathcal{F} in \mathcal{C} with the placement information, and simulation waveform $w(f_i, t)$ and its $W^0(\cdot)$, $W^1(\cdot)$, $\rho^0(\cdot)$, $\rho^1(\cdot)$ for every $f_i \in \mathcal{F}$,

1.1 Set $\mathcal{F}^0 = \{f_i | \rho^0(\{f_i\}) > p_{HIGH}\}$.²

1.2 Set $\mathcal{F}^1 = \{f_i | \rho^1(\{f_i\}) > p_{HIGH}\}$.

1.3 Apply Step 2 to \mathcal{F}^0 .

1.4 Apply Step 2 to \mathcal{F}^1 .

1.5 Apply Step 3 to $\mathcal{F} - (\mathcal{F}^0 \cup \mathcal{F}^1)$.

• **Step 2 (State driven clock gating):** For an input flip-flop set \mathcal{F} , D_{max} (in terms of Manhattan distance), and cost $\nabla P(S, f_i)$ which represents the amount of power consumption to be saved additionally by the inclusion of flip-flop f_i to flip-flop group S of clock gating. (The details on the cost formulation will be described in Sec. 3.10.)

2.1 Set $i = 1$.

2.2 If \mathcal{F} is empty, return. Otherwise, pick a seed, f_s , which has the largest number of flip-flops in \mathcal{F} within D_{max} .

² p_{HIGH} is a user controlled parameter, and set to 0.95 ~ 0.99 in our experiments.

2.3 Set $S_i = \{f_s\}$.

2.4 Expand flip-flop group S_i by iteratively including $f_j \in \mathcal{F}$ such that (*condition 1*) the half perimeter on $S_i \cup \{f_j\}$ is within D_{max} and (*condition 2*) its $\nabla P(S_i, f_j)$ is the largest positive value.

2.5 Implement a clock gating logic for S_i .

2.6 Update $i = i + 1$, $\mathcal{F} = \mathcal{F} - S_i$, and go to Step 2.2.

- **Step 3** (*Toggling based clock gating*): Apply a conventional toggling based clock gating to the rest of flip-flops unprocessed in Step 2.

Fig. 2.8 shows an illustrative example of generating flip-flop group for clock gating in Step 2.4, in which starting from seed f_1 picked in Step 2.2, flip-flops f_3 , f_2 , and f_5 are selected in the first, second, and third iterations, respectively.

2.2.4 Cost Formulation

Let S be a group of flip-flops selected for state-0 driven clock gating in Step 2 and $k = |S|$. (The power saving cost in state-1 driven clock gating can be similarly formulated.) The amount of power consumption decreased by including flop-flop f_j to S for a clock gating with respect to a clock gating to S (without f_j) is:

$$\nabla P_{saving}(S, f_j) = (P_g(S) + P_{ug}(f_j)) - P_g(S \cup \{f_j\}) \quad (2.4)$$

$$P_g = \sum_{i=1}^k P_{gFF}(\rho^0(S)) + P_{GL}^k(\rho^0(S)) \quad (2.5)$$

$$P_{GL}^k(\rho^0(S)) = P_{OR-tree}^k(\rho^0(S)) + P_{\sim const} \quad (2.6)$$

$$P_{\sim const} = P_{ICG}(\rho^0(S)) + P_{FF}(\rho^0(S)) + P_{OR}(\rho^0(S)) \quad (2.7)$$

- $P_{gFF}(\rho^0(S))$: the total power consumed by the flip-flops when state-0 clock gating is applied to the group S of flip-flops including the power consumed by the internal clock inverters in the flip-flops.

Grouping iteration (k)	S_i	$\nabla P(S_i, f_j)$	Remark
1	$\{f_s = f_1\}$	$\nabla P(S_i, f_2) = 50$ $\nabla P(S_i, f_3) = 150$ $\nabla P(S_i, f_4) = 10$ $\nabla P(S_i, f_5) = 80$ $\nabla P(S_i, f_6) = 10$	Selected
2	$\{f_1, f_3\}$	$\nabla P(S_i, f_2) = 30$ $\nabla P(S_i, f_4) = 10$ $\nabla P(S_i, f_5) = 20$ $\nabla P(S_i, f_6) = 5$	Selected
3	$\{f_1, f_2, f_3\}$	$\nabla P(S_i, f_4) = 2$ $\nabla P(S_i, f_5) = 8$ $\nabla P(S_i, f_6) = -5$	Selected
4	$\{f_1, f_2, f_3, f_5\}$	$\nabla P(S_i, f_4) = -5$ $\nabla P(S_i, f_6) = -20$	No positive value

$$S_i = \{f_1, f_2, f_3, f_5\}$$

$$i = i + 1, \mathcal{F} = \{f_4, f_6\} \rightarrow \text{Step 2.2}$$

Figure 2.8: Example of generating a flip-flop group S_i for clock gating. The input flip-flop set \mathcal{F} is $\{f_1, f_2, f_3, f_4, f_5\}$ and f_1 is picked as a seed in Step 2.2.

- $P_{GL}^k(\rho^0(S))$: the total power consumed by the state-0 clock gating logic i.e., OR-tree, ICG, a signal stretcher (flip-flop and a 2-input OR gate) where the power consumed by OR-tree increases as the group size k increases whereas the power consumed by ICG and a stretcher is almost constant regardless of the value of k .
- $P_{ug}(f_j)$: the power consumed by ungated flip-flop f_j including the power consumed by the internal clock inverters in the flip-flop.

Note that a fast calculation of the power costs, which will be iteratively required in Step 2.4, is enabled by simply referring the pre-computed data in a lookup table (LUT) forms.

2.3 Experiments

2.3.1 Experimental Setup

To evaluate the effectiveness of our state driven clock gating, we tested our method and the conventional clock gating methods for circuits taken from IWLS benchmarks [22]. The benchmark circuits were synthesized and physically implemented by using Synopsys *Design Compiler* and *IC Compiler*. The operating clock frequency was set to 200 MHz for all circuits and the initial layout utilization was 70%. We used Synopsys 32/28 nm Generic Library and a slow PVT corner to guarantee the worst case performance. In addition, for power analysis we performed RTL simulations to get the switching activity information of the benchmark circuits and used *PrimeTime PX* for power estimation.

To compare our clock gating called **State-driven CG** with the existing clock gating methods, we used the conventional idle logic driven clock gating in RTL (**Logic-driven CG**) and toggle driven clock gating (**Toggling-driven CG**) provided by *Design Compiler* and *IC Compiler*. We tested three clock gating flows for the experi-

ment as shown in Fig. 2.9. Those are (1) flow of Logic-driven CG only, (2) flow of Logic-driven CG followed by Toggling-driven CG, and (3) flow of Logic-driven CG followed by our State-driven CG and finally Toggling-driven CG as described in Sec. 2.2.3.

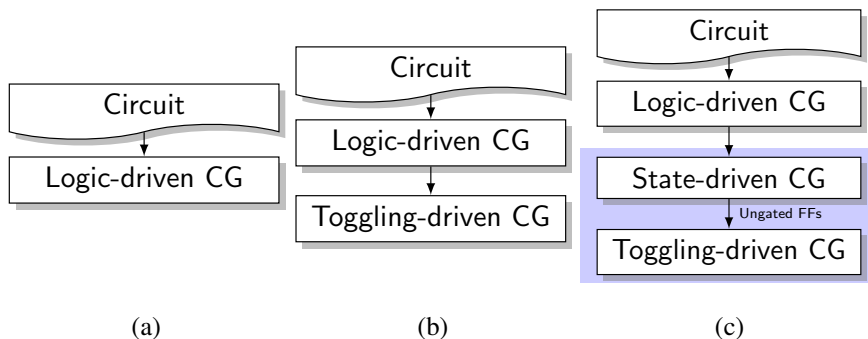


Figure 2.9: Tested clock gating flows. (a) Logic-driven CG only. (b) Logic-driven CG followed by Toggling-driven CG. (c) Logic-driven CG followed by our State-driven CG and finally Toggling-driven CG.

Table 2.1 shows the information of benchmark circuits including the number of flip-flops, the number of gates, the percentage of the number of flip-flops whose state-1 probability is smaller than 0.05 over the total number of flip-flops (i.e., state-0 probability > 0.95), and the percentage of the number of flip-flops whose state-1 probability is bigger than 0.95. To extract the switching activity information, we performed simulations of typical operation modes for a sufficiently long period of time.

2.3.2 Experimental Results

Table 2.2 summarizes the results produced by the conventional idle logic driven and toggling driven clock gating methods provided by a commercial electronic design automation (EDA) tool and our state driven clock gating method combined with the conventional methods. The Logic-driven CG, which has been commonly applied to practical low power designs, is used as a baseline for the comparison. We compared the

Table 2.1: Benchmark circuit information.

Circuit	# of FFs	# of gates	% of state-1 prob. < 0.05	% of state-1 prob. > 0.95
SPI	229	1612	27.90%	12.20%
WB_DMA	523	2571	17.00%	39.77%
AES_CORE	530	8422	7.74%	6.42%
WB_CONMAX	770	19600	39.61%	10.39%
MEM_CTRL	1065	5078	31.74%	19.34%
AC97_CTRL	2199	8471	21.87%	4.37%
VGA_LCD	17053	62187	14.95%	43.47%

clock gating ratio (i.e., the ratio of the number of gated flip-flops to the total number of flip-flops), the number of clock gating cells, and the power consumption of clock tree (P_{clk}), flip-flops (P_{ff}), combinational logics (P_{combi}), the total power consumption (P_{total}), the percentage of reduction of total power consumption ³.

Compared with the conventional clock gatings, the flip-flop power P_{ff} was reduced consistently and effectively by our method for all test cases while there were fluctuations in the clock power P_{clk} due to the load changes on the driving buffers to the groups of gated flip-flops. In addition, the combinational logic power P_{combi} was increased because more flip-flops were gated by our method, which increased the corresponding supplemental clock gating logic. However, the flip-flop power P_{ff} and the clock power P_{clk} dominated the overall power, therefore decreasing the total power consumption over the conventional clock gating methods. In summary, our method reduced the total power consumption by 10.81% on average (up to 28.23%) while the conventional toggling driven clock gating reduced the total power consumption by 3.22% on average (up to 9.93%).

It is worthy to note that the power consumption of AES_CORE was increased by

³A negative sign indicates increase.

Table 2.2: Comparison of the clock gating ratio, the number of clock gating cells (i.e., ICGs), the clock tree power P_{clk} , the flip-flop power P_{ff} , the combination logic power P_{combi} including the power by clock gating logic, the total power P_{total} produced (1) by the conventional idle logic driven clock gating (Logic-driven CG), produced (2) by Logic-driven CG followed by the conventional toggling driven clock gating (Toggling-driven CG), and produced (3) by Logic-driven CG followed by our state driven clock gating (State-driven CG), finally applying Toggling-driven CG.

[†]Excluding the power consumed by the internal clock inverters in the flip-flops. [‡]Including the power consumed by the internal clock inverters in the flip-flops.

Circuit	Logic-driven CG				Logic-driven + Toggling-driven CG				Logic-driven + our State-driven + Toggling-driven CG										
	CG ratio	# of ICGs	$\dagger P_{clk}$	$\dagger P_{ff}$	P_{combi}	P_{total}	CG ratio	# of ICGs	$\dagger P_{clk}$	$\dagger P_{ff}$	P_{combi}	P_{total}	CG ratio	# of ICGs	$\dagger P_{clk}$	$\dagger P_{ff}$	P_{combi}	P_{total}	
SPI	76.4%	8	85.8	232.4	93.5	411.7	79.9%	12	82.3	212.0	96.7	391.0	85.5%	15	75.0	166.8	123.7	365.5	
WB_DMA	60.6%	14	133.1	448.8	67.0	648.9	85.7%	31	168.1	380.1	77.5	625.6	88.3%	43	129.1	256.0	80.6	465.7	
AES_CORE	24.9%	2	155.7	797.0	963.0	1916.0	26.0%	20	174.6	793.6	971.5	1940.0	27.5%	7	175.7	769.2	982.1	1927.0	
WB_CONMAX	49.9%	24	317.2	643.4	875.3	1836.0	73.4%	50	370.0	559.7	894.7	1824.0	76.1%	50	355.4	462.5	913.6	1731.0	
MEM_CTRL	73.3%	41	198.1	594.3	85.6	877.9	87.0%	76	252.4	438.5	99.7	790.7	90.7%	93	225.1	416.4	104.7	746.2	
AC97_CTRL	72.4%	62	257.2	753.5	91.3	1102.0	91.4%	118	315.0	619.0	115.8	1050.0	86.7%	106	337.0	577.7	106.5	1021.0	
VGA_LCD	98.7%	701	1400.0	2661.0	751.4	4835.0	98.9%	736	1401.0	2660.0	754.9	4840.0	99.1%	765	993.9	2646.0	745.3	4414.0	
Avg. reduction						—						3.22%							10.81%

both the toggling driven clock gating and our method compared with that of the logic driven clock gating. In the case of designs in which most flip-flops exhibit high switching activity like AES_CORE, our method and toggling driven clock gatings are all unsuitable to the designs due to a high demand of supplemental clock gating logic. On the other hand, for the other circuits in which a large portion of flip-flops tends to be stuck at state-0 or state-1 most of time, our method is able to reduced the power consumption very effectively.

Table 2.3 shows the area used by the conventional idle logic driven and toggling driven clock gating methods and our state driven clock gating method with the conventional methods. Compared with the logic driven clock gating, the area was increased by 5.4% on average by our approach but it was almost the same as that of the toggling driven clock gating.

Fig. 2.10 shows the layouts of WB_DMA produced by Logic-driven CG, Toggling-driven CG, and our State-driven CG. The colored rectangles indicate flip-flops and the white flip-flops are ungated flip-flops. Compared with the conventional methods in Figs. 2.10(a) and (b), more flip-flops were gated by our method.

Table 2.3: Comparison of total area and cell area used (1) by the conventional idle logic driven clock gating (Logic-driven CG), used (2) by Logic-driven CG followed by the conventional toggling driven clock gating (Toggling-driven CG), and used (3) by Logic-driven CG followed by our state driven clock gating (State-driven CG), finally applying Toggling-driven CG.

Circuit	Logic-driven CG		Logic-driven + Toggling-driven CG		Logic-driven + our State-driven + Toggling-driven CG	
	Cell area (μm^2)	Total area (μm^2)	Cell area (μm^2)	Total area (μm^2)	Cell area (μm^2)	Total area (μm^2)
SPI	4175.8	5359.6	4308.8 (3.2%)	5475.9 (2.2%)	4393.9 (5.2%)	5548.2 (3.5%)
WB_DMA	6811.3	8516.0	7661.4 (12.5%)	9379.1 (10.1%)	7986.0 (17.2%)	9744.0 (14.4%)
AES_CORE	22358.3	29442.5	23220.4 (3.9%)	30299.0 (2.9%)	22511.1 (0.7%)	29588.1 (0.5%)
WB_CONMAX	52619.2	85782.3	53952.0 (2.5%)	87096.0 (1.5%)	53545.3 (1.8%)	86103.5 (0.4%)
MEM_CTRL	13088.2	16216.8	14668.2 (12.1%)	17907.4 (10.4%)	15026.8 (14.8%)	18268.2 (12.6%)
AC97_CTRL	21722.7	28142.9	24398.1 (12.3%)	30798.3 (9.4%)	23406.4 (7.8%)	29811.5 (5.9%)
VGA_LCD	166265.1	263973.9	167287.8 (0.6%)	264698.8 (0.3%)	167239.0 (0.6%)	265001.9 (0.4%)
<i>Avg. increase</i>			6.7%	5.3%	6.9%	5.4%

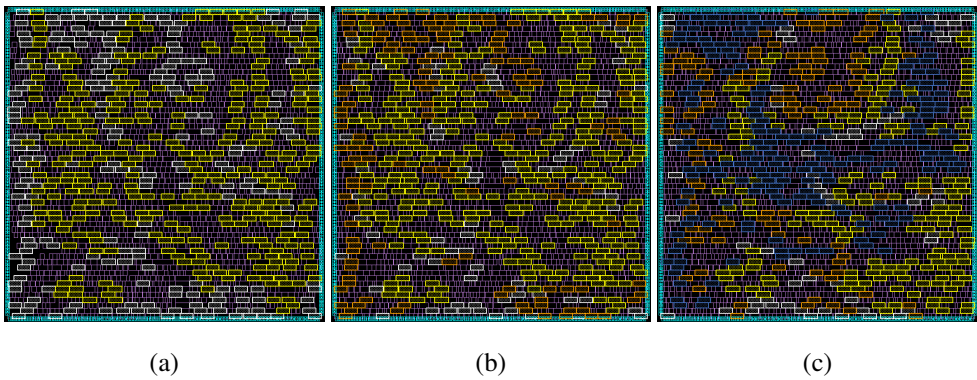


Figure 2.10: Layouts for WB_DMA. The colored rectangles represent flip-flops: ungated (white), gated by Logic-driven CG (yellow), gated by Toggling-driven CG (orange), gated by our State-driven CG (blue). (a) Layout produced by Logic-driven CG. (b) Layout produced by Toggling-driven CG. (c) Layout produced by our State-driven CG.

Chapter 3

ALGORITHM AND DESIGN OPTIMIZATION OF ALLOCATING MULTI-BIT RETENTION FLIP-FLOPS FOR POWER GATED CIRCUITS

3.1 Motivations

3.1.1 Flip-flops with Mux-feedback Loop

Fig. 3.1(a) and Fig. 3.1(b) show a section of Verilog code that commonly appears in the description of design behavior and its synthesized structure, respectively, from which we can see that each of the eight flip-flops contains combinational (i.e., mux-feedback) loop. We call such a flop-flop that has a mux-feedback loop a *self-loop FF* while we call a flip-flop which has no self-loop an *ordinary FF*.

Table 3.1 summarizes the number of self-loop FFs in the circuits synthesized from IWLS benchmark code [22]. It is shown that the proportion of self-loop FFs is in 24%~77% of the total number of flip-flops in circuits, which clearly indicates that a careful treatment should be taken into account when allocating state retention flip-flops to circuits with a high portion of self-loop FFs.

It should be noted 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

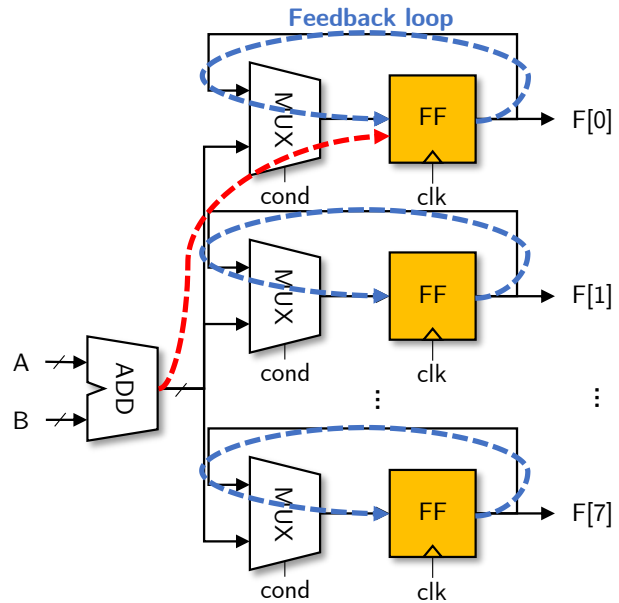
```

...
reg [7:0] F;

always @( posedge clk )
begin
...
if ( cond )
    F <= A + B;
...
end

```

(a)



(b)

Figure 3.1: Synthesis of flip-flops with mux-feedback loop. (a) a Verilog HDL code (b) a synthesized structure for the code in (a).

Table 3.1: Proportion of self-loop FFs in IWLS benchmark circuits.

Circuit	# of FFs	# of self-loop FFs	% of self-loop FFs
s15850	128	40	31.25%
s13207	214	100	46.73%
wb_dma	523	324	61.95%
aes_core	530	132	24.91%
mem_ctrl	1118	864	77.28%
Avg.	503	292	48.42%

whether the flip-flop state, when waking up, comes from the mux-feedback loop or driving flip-flops other than itself (e.g., the red signal path in Fig. 3.1). Unfortunately, to our knowledge, all conventional approaches [14, 18, 25] are not fully aware of the abundance of self-loop FFs, and simply performed the allocation of retention flip-flop to self-loop FFs as a pre-processing or post-processing task. Specifically, they could not but adopt one of the following two flow options:

Flow option 1: (1.1) Generating a flip-flop dependency graph G' by removing all self-loops in the original dependency graph G of input circuit; (1.2) applying any conventional approach of MBRFF allocation to G' ; (1.3) additionally, allocating an SBRFF for every self-loop FF if it has not been replaced with a retention flip-flop during step 1.2.

Example: Fig. 3.2(b) shows graph G' in *Flow option 1*, obtained from the original flip-flop dependency graph G in Fig. 3.2(a). Then, the upper part in Fig. 3.2(d) shows the MBRFF allocation produced by applying the ILP-based method in [18] to G' in Fig. 3.2(b) while constraining the maximum wakeup latency to 3 clock cycles. Finally, the lower part in Fig. 3.2(d) is obtained by simply adding two SBRFFs to self-loop FFs f_5 and f_6 .

Flow option 2: (2.1) Generating a set S of flip-flop dependency subgraphs by decomposing the original graph G , so that every self-loop FF in the decomposed maximal subgraph should have no driving flip-flops (i.e., no predecessors); (2.2) applying any conventional approach of MBRFF allocation to all subgraphs in S independently while ensuring an MBRFF/SBRFF allocation for every self-loop FF.

Example: Fig. 3.2(c) shows set S , in *Flow option 2*, of three connected components, obtained from G in Fig. 3.2(a). Then, Fig. 3.2(e) shows the MBRFF allocation produced by applying the ILP-based method in [18] to the connected subgraphs in S in Fig. 3.2(c) while constraining the maximum wakeup delay to 2 clock cycles. Thus, total of 8 retention bits are allocated.

However, for G in Fig. 3.2(a) it is possible to save two more retention bits, as

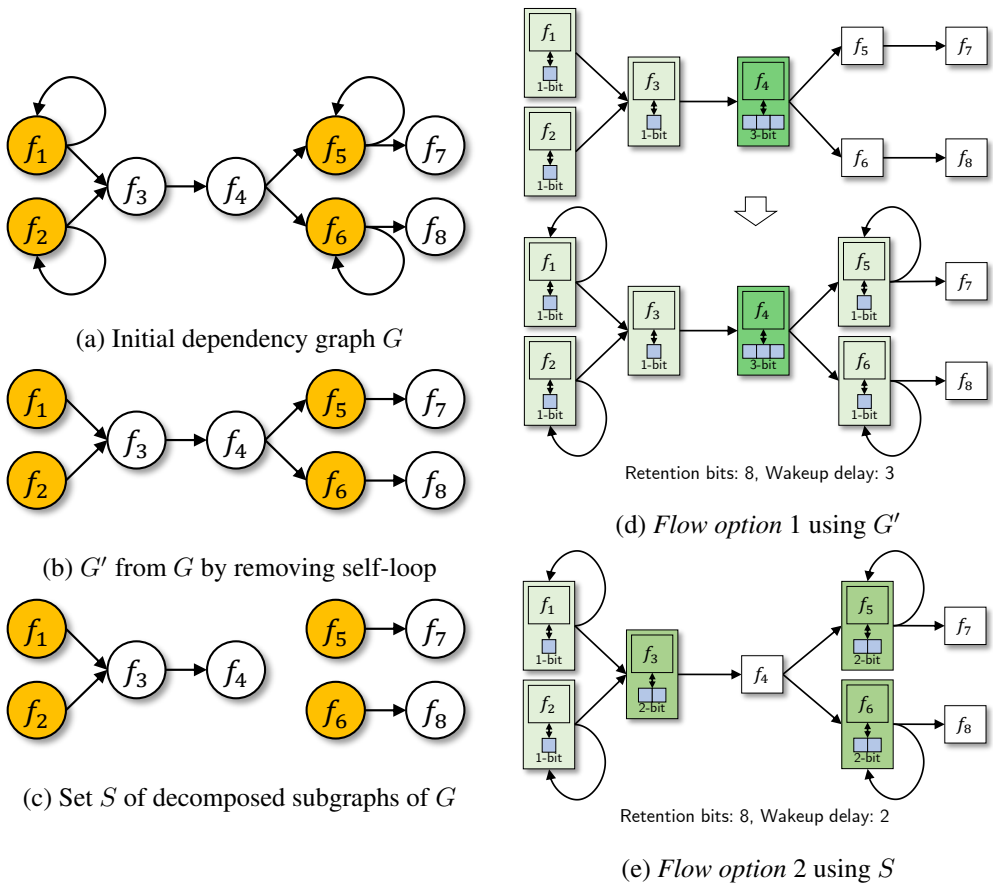
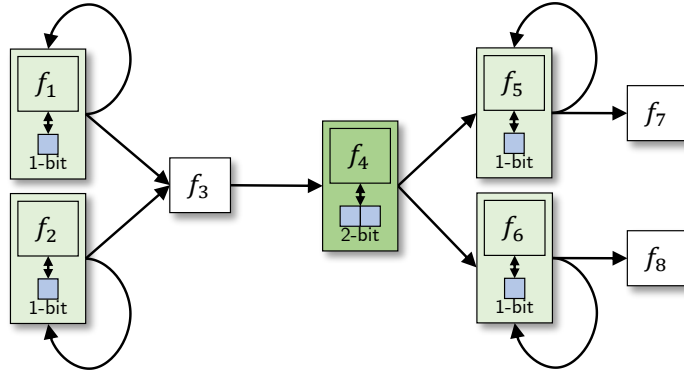


Figure 3.2: Two flow options used by the conventional approaches of allocating retention flip-flops to circuits with self-loop FFs.



Retention bits: 6, Wakeup delay: 2

(a)

Cycle	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
t_0	X	X	X	X	X	X	X	X
	$\{d_1^1\}$	$\{d_2^1\}$		$\{d_4^1, d_4^2\}$	$\{d_5^1\}$	$\{d_6^1\}$		
t_1	d_1^1	d_2^1	X	d_4^1	d_5^1	d_6^1	X	X
				$\{d_4^2\}$				
t_2	d_1^2	d_2^2	d_3^2	d_4^2	d_5^2	d_6^2	d_7^2	d_8^2

(b)

Figure 3.3: (a) A better allocation of retention bits over that in Fig. 3.2. (b) The state restoration of the wakeup sequence for the retention bit allocation in (a). ($\{\dots\}$ indicates the state(s), to be used when waking up, stored in the retention bit(s) and d_i^j indicates the state of flip-flop f_i at time t_j , $j = 1, 2$.)

shown in Fig. 3.3, over the allocation results produced by *Flow options* 1 and 2. Specifically, Fig. 3.3(b) illustrates the state restoration when waking up for the retention storage allocation in Fig. 3.3(a): At the first clock cycle, i.e., at time t_1 , states of flip-flops f_1 , f_2 , f_4 , f_5 , and f_6 will be set by using their own retention bits. Then, at the next cycle, i.e., at time t_2 , states of f_7 and f_6 will be set by using the states of f_5 and f_6 , respectively; the states (i.e., d_5^2 and d_6^2) of f_5 and f_6 are set by using either their states (i.e., d_5^1 and d_6^1) or the state (i.e., d_4^1) of f_4 ; state of f_4 is set by its own retention bit (i.e., d_4^2); state of f_3 is set by the states of f_1 and f_2 ; state of f_1 and f_2 are set by the driving inputs.

3.1.2 Impact of Wakeup Delay

Intuitively, it is obvious that a long wakeup delay enables to provide an increased opportunity of reducing total size of retention storage at the expense of the loss of circuit performance. To observe the impact of the wakeup delay on the size of retention storage, we measured the reductions of retention storage size allocated by *Flow options* 1 and 2 using the conventional MBRFF algorithm [18] by varying the constraint of wakeup delay for IWLS benchmark circuits in Table 3.1. The bars in Fig. 3.4 show the average changes of reduction rate of retention storage size for *Flow options* 1 and 2 as the wakeup delay constraint changes, which strongly points out that *wakeup latency of 2 clock cycles (the purple bars in Fig. 3.4) suffices in practice.*

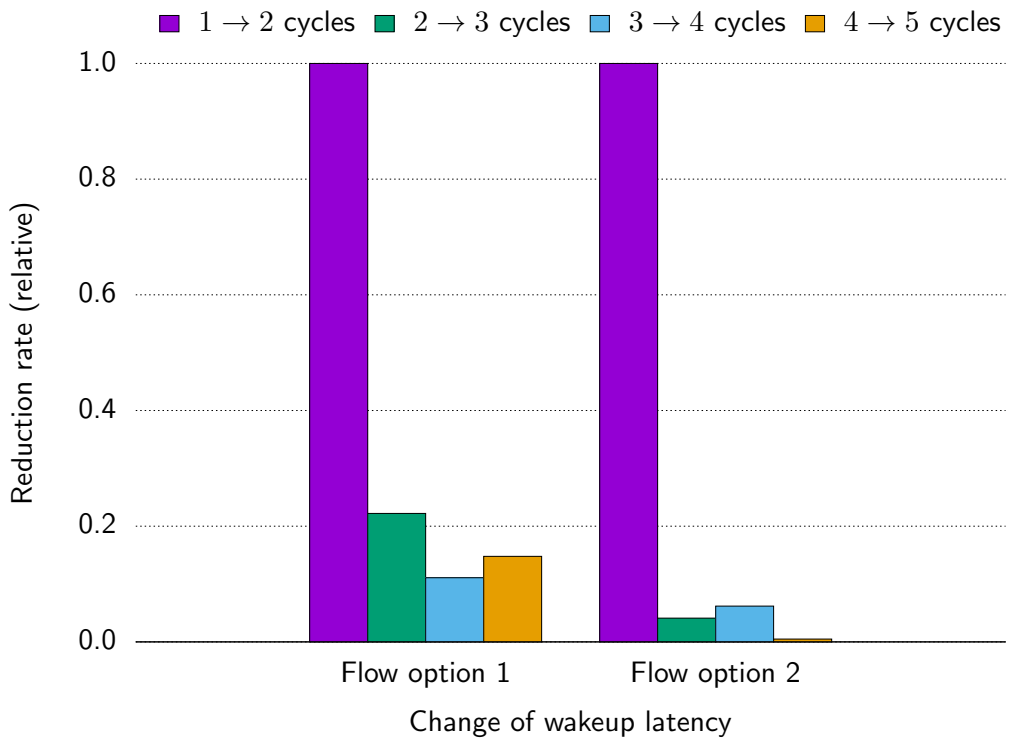


Figure 3.4: Average changes of the reduction rate of the total size of retention storage by *Flow options* 1 and 2 using [18] as the wakeup latency changes for IWLS benchmark circuits in Table 3.1.

3.2 The Proposed Allocation Algorithm

We propose a practically effective algorithm for allocating state retention storage that is able to perform a global minimization of total size of retention storage for self-loop FFs as well as ordinary FFs in circuits under the wakeup delay is strictly limited to two clock cycles. We start with a few definitions and concepts.

Definition 1. (2-phase retention control scheme): The state retention flip-flops to be used by our allocation algorithm can be classified by three types: (1) *1st-phase SBRFF* that retains or restores at the *first clock cycle* of power-down or wakeup sequence, (2) *2nd-phase SBRFF* that retains or restores at the *second clock cycle* of power-down or wakeup sequence, and (3) *2-bit MBRFF*.

Lemma 1. *For a flip-flop dependency $f_i \rightarrow f_j$ in G , if f_i is replaced by a 2nd-phase SBRFF, f_j cannot be replaced by a 1st-phase SBRFF. (It is clear by observing that the input state of f_j is unknown at the 2nd phase of the wakeup sequence.)*

Table 3.2 describes the detailed operation of the three types of retention flip-flop at power down and wakeup modes.

Table 3.2: State retention and restoration for the three types of retention flip-flop. (The notations in parentheses represent the values in the retention storage of the corresponding SBRFFs/MBRFF and cycle times.)

Cycle	Power state	1st-phase SBRFF (f_i)	2nd-phase SBRFF (f_j)	2-bit MBRFF (f_k)
t_l	<i>on</i>	d_i^l	d_j^l	d_k^l
t_{l+1}	<i>power down</i>	d_i^{l+1}	d_j^{l+1}	d_k^{l+1}
		$\{d_i^{l+1}\}$	$\{\}$	$\{d_k^{l+1}\}$
t_{l+2}	<i>power down</i>	d_i^{l+2}	d_j^{l+2}	d_k^{l+2}
		$\{d_i^{l+1}\}$	$\{d_j^{l+2}\}$	$\{d_k^{l+1}, d_k^{l+2}\}$
\dots	<i>sleep</i>	X	X	X
		$\{d_i^{l+1}\}$	$\{d_j^{l+2}\}$	$\{d_k^{l+1}, d_k^{l+2}\}$
t_m	<i>sleep</i>	X	X	X
		$\{d_i^{l+1}\}$	$\{d_j^{l+2}\}$	$\{d_k^{l+1}, d_k^{l+2}\}$
t_{m+1}	<i>wakeup</i>	d_i^{l+1}	X	d_k^{l+1}
		$\{\}$	$\{d_j^{l+2}\}$	$\{d_k^{l+2}\}$
t_{m+2}	<i>wakeup</i>	d_i^{l+2}	d_j^{l+2}	d_k^{l+2}

Definition 2. (Retention pair): A retention pair in a flip-flop dependency graph G refers to a pair of nodes f_i and f_j in G that satisfies (1) f_i directly drives f_j and (2) f_j is an ordinary FF, i.e., not a self-loop FF as shown in Fig. 3.5. A retention pair is called a *type 1 retention pair* if f_i is an ordinary FF and is called a *type 2 retention pair* if f_i is a self-loop FF. We use notation $p(f_i, f_j)$ to refer the retention pair. For example, the pairs of flip-flops connected by red and blue colors in Fig. 3.6 indicate type 1 and type 2 retention pairs, respectively.

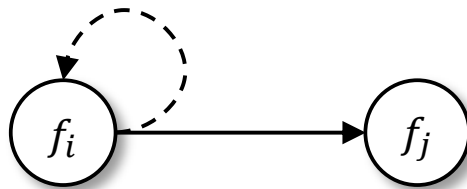


Figure 3.5: Retention pair $p(f_i, f_j)$.

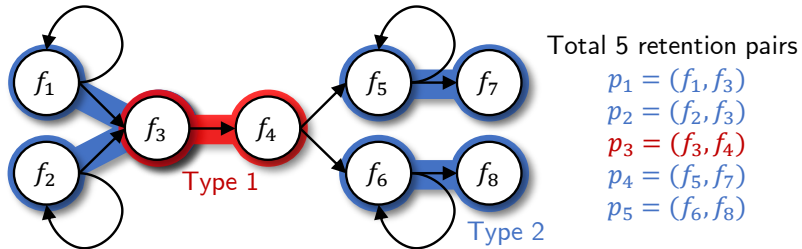
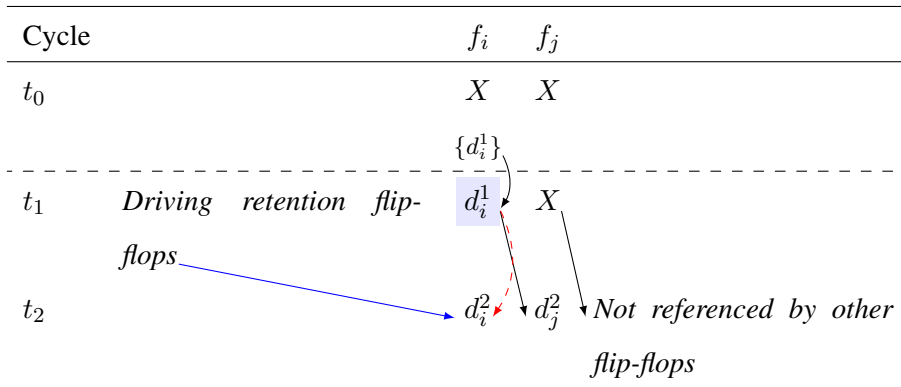


Figure 3.6: Retention pairs of type 1 (in red color) and type 2 (in blue color) in a flip-flop dependency graph G .

Table 3.3: State restoration of flip-flops in a retention pair $p(f_i, f_j)$ where f_i is minimally allocated by a 1st-phase SBRFF. The state of f_i at the second cycle will be set by one of the blue solid and red dotted arrows if f_i is a self-loop FF (i.e., type 2 retention pair) and otherwise (i.e., type 1 retention pair) will be set only by the blue arrow.



Definition 3. (Conflict between retention pairs): A *conflict* between two retention pairs $p(f_i, f_j)$ and $p(f_k, f_l)$ in a flip-flop dependency graph G exists if any of the following conditions hold:

- *Condition 1:* $f_j = f_k$.
- *Condition 2:* f_j directly drives f_k .
- *Condition 3:* f_j directly drives f_l .

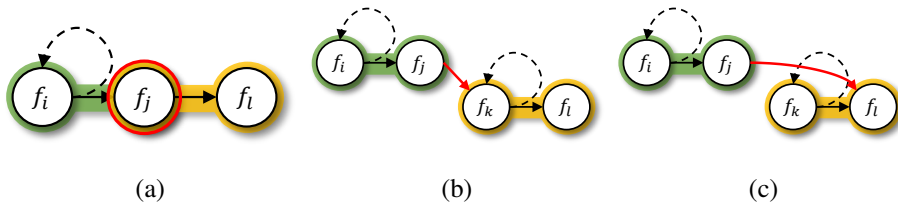


Figure 3.7: Conditions of conflict between retention pairs. (a) *Condition 1* (b) *Condition 2* (c) *Condition 3*.

Lemma 2. *The following allocations are required **minimally** for a retention pair $p(f_i, f_j)$ and its driven and driving flip-flops. (The allocation requirement can be easily checked by examining the activity in the two-cycle wakeup sequence.)*

1. f_i minimally requires a 1st-phase SBRFF.
2. f_j minimally requires no retention.
3. A flip-flop that drives f_i or f_j minimally requires a 1st-phase SBRFF.
4. A flip-flop that is driven by f_j minimally requires a 2nd-phase SBRFF.
5. A flip-flop that is driven by f_i minimally requires no retention.

Fig. 3.8 summarizes the minimally required retention bits for the flip-flops in a retention pair and its driving and driven flip-flops. For example, Table 3.3 shows the state

restoration for a pair $p(f_i, f_j)$ of either type 1 or type 2, in which f_i is minimally allocated with a 1st-phase SBRFF. In addition, Fig. 3.9 shows examples of the conflicts between retention pairs by *conditions 1, 2, and 3* and the minimally required retention bits for the green pairs when the yellow pairs are minimally allocated according to the allocation rules in *Lemma 2* and Fig. 3.8.

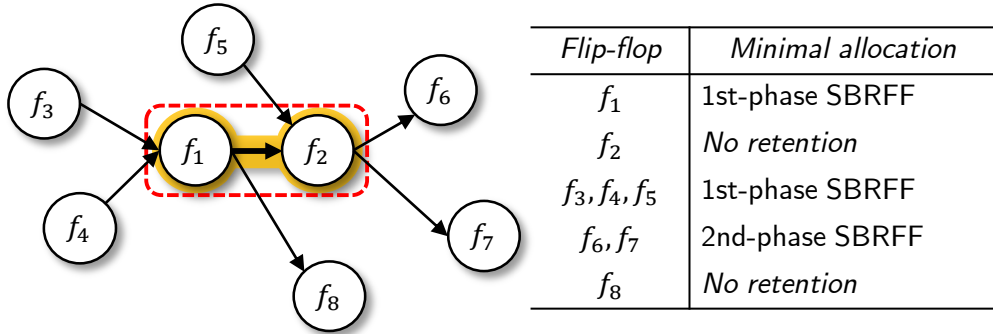


Figure 3.8: Minimally required retention bits for the flip-flops in a retention pair $p(f_1, f_2)$ and its driving and driven flip-flops.

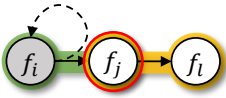
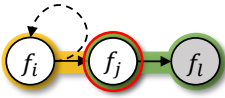
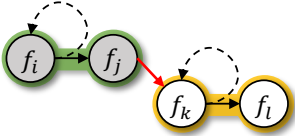
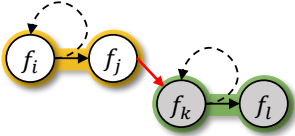
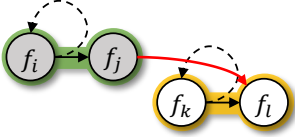
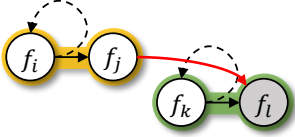
	Inbound	Minimal cost	Outbound	Minimal cost
<i>Cond1</i>		1		1
<i>Cond2</i>		2		2
<i>Cond3</i>		2		1

Figure 3.9: Two retention pairs that cause conflicts due to each of *conditions 1, 2, and 3*, and the minimally required retention bits for the green pair when the yellow pair is allocated by the minimally required retention bits.

Thus, our allocation strategy is: we maximally extract a set, R , of retention pairs with no conflict each other, and apply *Lemma 2* to R with the objective of minimally allocating retention bits to G . (Note that due to an existence of conflicts among pairs, some flip-flops shall be allocated with 2-bit MBRFFs to resolve the conflicts.)

Our allocation approach consists of three steps: (Step 1) generating a flip-flop dependency graph G from an input circuit \mathcal{C} and then a conflict graph G_p of retention pairs from G , (Step 2) extracting a maximal retention pairs with non-conflict from G_p by applying an iterative heuristic, and (Step 3) resolving the allocation conflicts among the flip-flops with a minimal additional allocation of retention bits.

Step 1. *Generating flip-flop dependency graph and its conflict graph:* We first convert an input circuit \mathcal{C} into a flip-flop dependency graph $G(V, E)$. $G(V, E)$ is a directed graph where nodes in V represent distinct flip-flops in \mathcal{C} and there exists an edge $f_i \rightarrow f_j$ in E if and only if the flip-flop of f_i drives the flip-flop of f_j through a combinational logic path in \mathcal{C} . We then extract a set, P , of all retention pairs from G in $O(|V|^2)$ time. Finally, we construct a conflict graph $G_p(V_p, E_p)$ from P , in which nodes in V_p represent distinct retention pairs in P and there is an edge between nodes p_i and p_j in G_p if and only if a conflict exists between the retention pairs of the two nodes.

Step 2. *Extracting a maximal retention pairs:* We transform the problem of finding a maximal retention pairs in G_p into a maximum independent set problem [23] on G_p , and solve it by applying a greedy heuristic: At each iteration, we select the node in G_p which has the least number of adjacent nodes (i.e., the smallest degree). If there are ties, we choose the node p_i that has the smallest value of $Cost(\cdot)$:

$$Cost(p_i) = 1 \cdot N_{In}^1 + 1 \cdot N_{Out}^1 + 2 \cdot N_{In}^2 + 2 \cdot N_{Out}^2 + 2 \cdot N_{In}^3 + 1 \cdot N_{Out}^3 \quad (3.1)$$

where N_{In}^i and N_{Out}^j , $j = 1, 2, 3$ represent the the numbers of the inbound and out-bound retention pairs, as defined in Fig. 3.9 with respect to p_i for *condition j*, respectively. Thus, $Cost(p_i)$ indicates a minimally required cost in terms of the number of

retention bits caused by the selection of the p_i . Fig. 3.10 illustrates how the tie-breaking is applied by using the cost in Eq.(3.1).

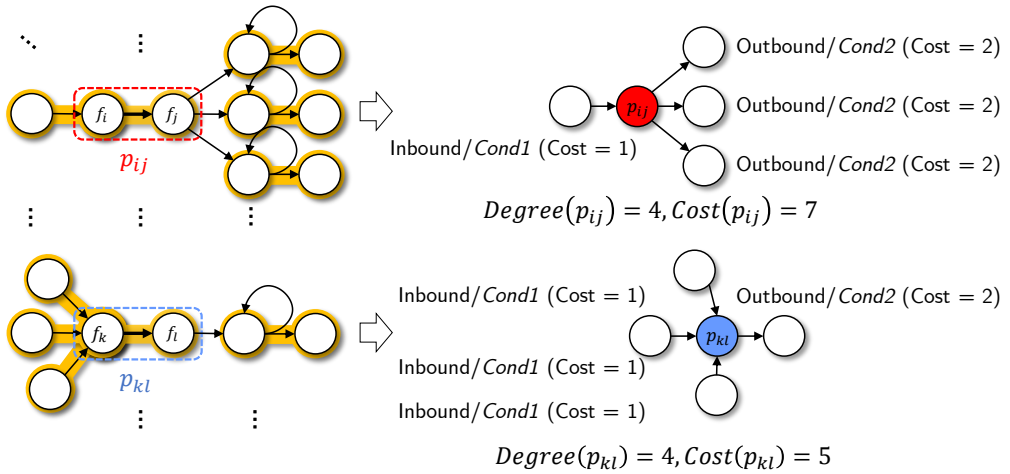


Figure 3.10: Tie-breaking based on the cost formulation in Eq.(3.1).

Step 3. Resolving conflicts with a minimal allocation of retention bits: This step resolves the conflicts caused by the allocation performed in step 2 according to the rules of minimal retention bit allocation in Lemma 1, Lemma 2, and Fig. 3.9. Thus, in case where a particular flip-flop is minimally required to be both of a 1st-phase SBRFF and a 2nd-phase SBRFF, it should be replaced with a 2-bit MBRFF.

3.3 Design of Multi-Bit Retention Flip-Flop and Multi-Bit Extension

3.3.1 Multi-Bit Retention Flip-Flop

A MBRFF design was proposed by Chen *et al.* [14]. In this design, a pulse-driven latch array is used instead of the conventional shift registers to reduce the area overhead. Fig. 3.11 shows the conceptual schematic of this pulse-driven latch array. Each retention latch has an extra three-stage inverter chain to provide the necessary delay for shift operation.

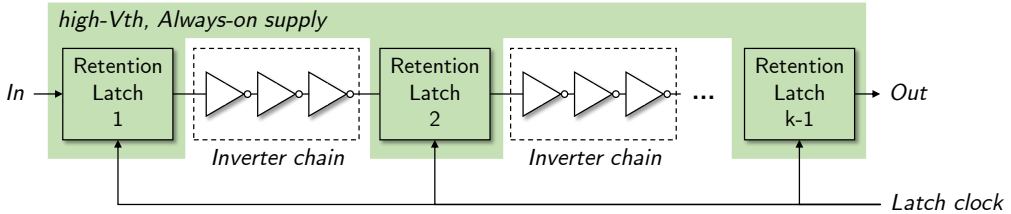


Figure 3.11: Conceptual schematic of the pulsed latch array of the k -bit retention register in [14]. High-V_t transistors powered by an always-on power supply are used for the retention latches (in green color).

However there are two critical issues in this design [14]. First, the width of the pulse signal for the retention latches should be smaller than the delay of the inverter chain and the global network of this pulse signal also should be designed carefully. But it is not easy to implement the narrow pulse network physically to meet this constraint. In addition, the number of the stage of the inverter chain might need to be increased to consider on-chip variation and it can increase the area overhead further.

Second, it is possible the overall area might increase rather by using multi-bit retention flip-flops due to the area overhead of the inverter chain. Each retention latch has the extra inverter chain therefore this overhead is inevitable and the overall area can be increased even the total storage size of the retention registers is decreased.

To address these issues, we devise a new design for the MBRFF especially where

the size of the retention storage is two bits which is the maximum size required by our approach. Fig. 3.12 shows the schematic of our 2-bit MBRFF. Compared with the conventional design of [14], there is no inverter chain in the retention latches. Instead, an extra pin *SHIFT* for the clock of the second retention latch is used. Fig. 3.13 shows the shift operation of the proposed design of retention latches.

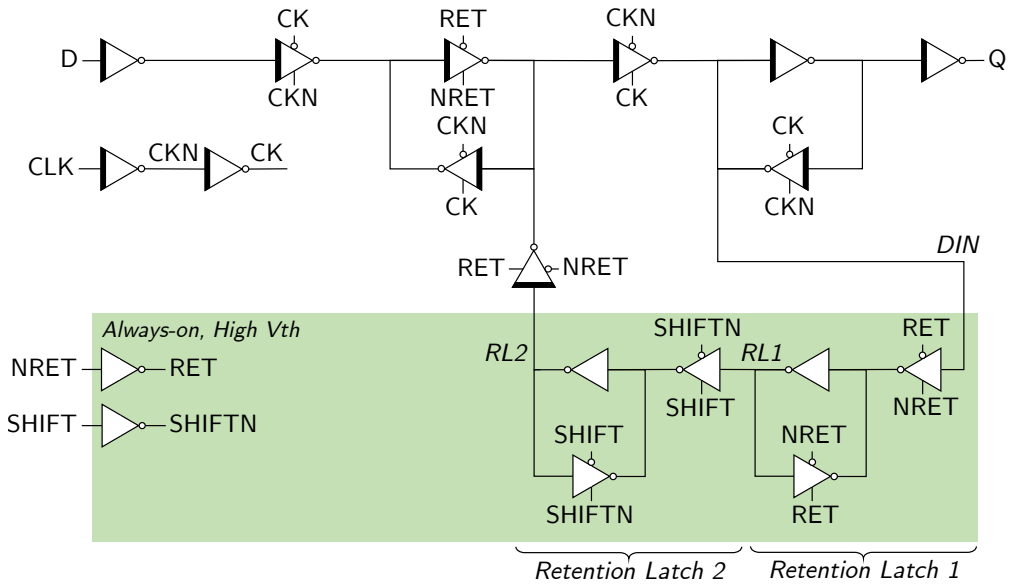


Figure 3.12: Schematic of our 2-bit MBRFF. The always-on supply region in green color consists of two retention latches and two input inverters for control pins *NRET* and *SHIFT*.

To verify the functionality of the retention operation, we implemented a simple power gated design composed of a power switch cell and an our 2-bit MBRFF and then simulated it using Synopsys HSPICE for all process corners supported by the library. Fig. 3.14 illustrates the spice simulation waveforms of our 2-bit MBRFF design at the typical operating condition (TT/1.05V/25°C). During the power down process, the input data are shifted sequentially into the retention latches by the latch clock signals *RET*N and *SHIFT*. When the control signal *NSLEEP* of the power switch cell goes down, the gated power supply *VVDD* is turned off while the retention latches in

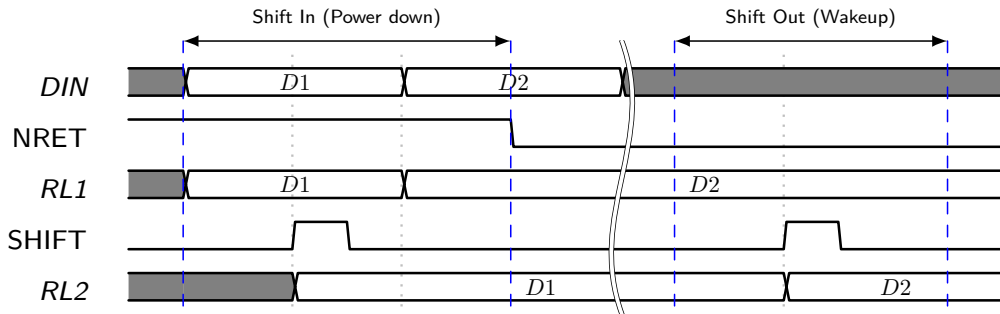


Figure 3.13: Waveforms of the 2-bit shift register using two control signals.

the always-on supply region are powered by the always-on power supply *RVDD* and retain the data. After the gated power supply *VVDD* is turned on again according to the power control signal *NSLEEP*, the retained data is shifted out from the retention latches during the consecutive two clock cycles.

It is worth to note that the proposed design has no area overhead due to the inverter chain while the extra pin *SHIFT* is added newly and it requires an additional buffer network powered by the always-on power supply.

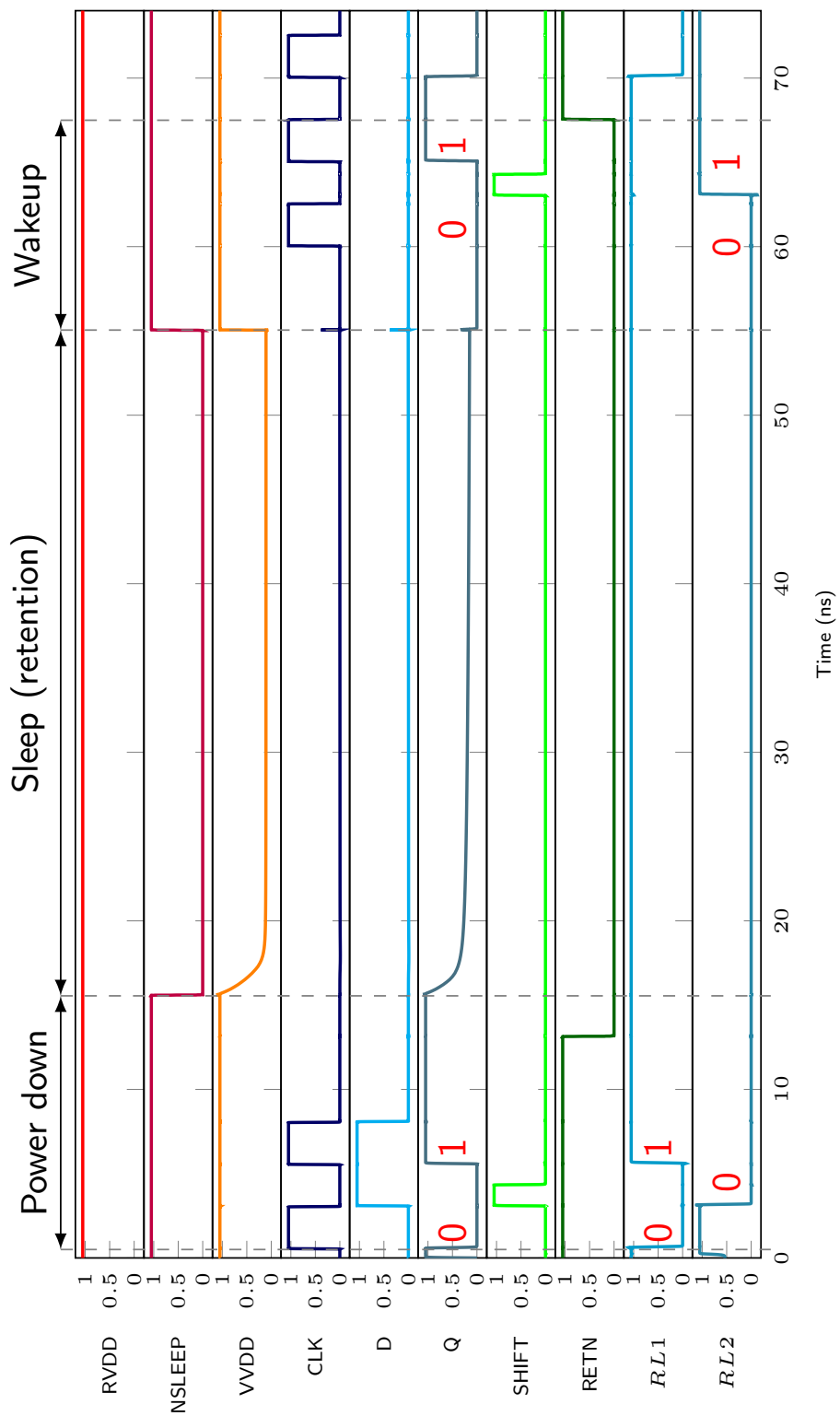


Figure 3.14: Spice simulation waveforms of a 2-bit MBRFF.

3.3.2 Multi-Bit Flip-Flop Extension

A multi-bit flip-flop (MBFF) is a cell which contains several single-bit flip-flops and shares common inverters mainly to save the clock power consumption [24].

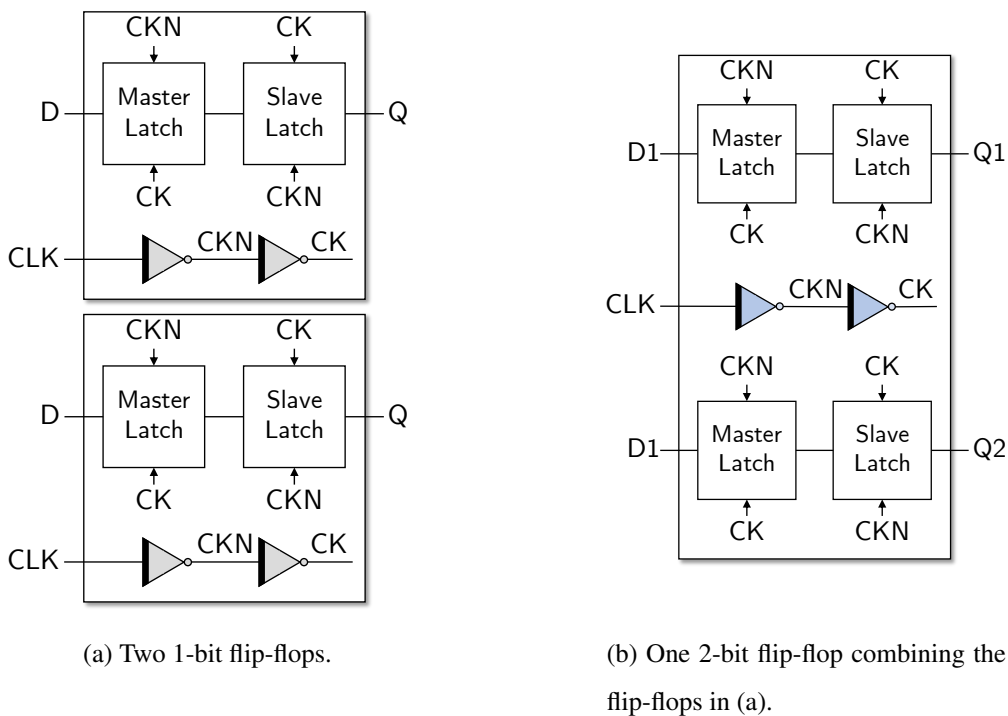


Figure 3.15: Comparison of the internal structures of the 1-bit master-slave based flip-flop and the 2-bit master-slave flip-flop.

Fig. 3.15 shows the internal structures of the transitional 1-bit master-slave flip-flop and the 2-bit MBFF which plays same role of two 1-bit flip-flops. Each 1-bit flip-flop has two inverters for the clock signal respectively while the 2-bit MBFF has only two inverters for driving all master and slave latches. Therefore we can reduce the clock power consumption by adopting the MBFF.

By the similar way, the internal always-on inverters and the always-on network for *SHIFT* of multiple MBRFFs can be shared by merging those MBRFFs to a single multi-bit retention multi-bit flip-flop (MBR-MBFF).

Fig. 3.16 shows the schematic of the proposed MBR-MBFF where two 2-bit MBRFFs are merged.

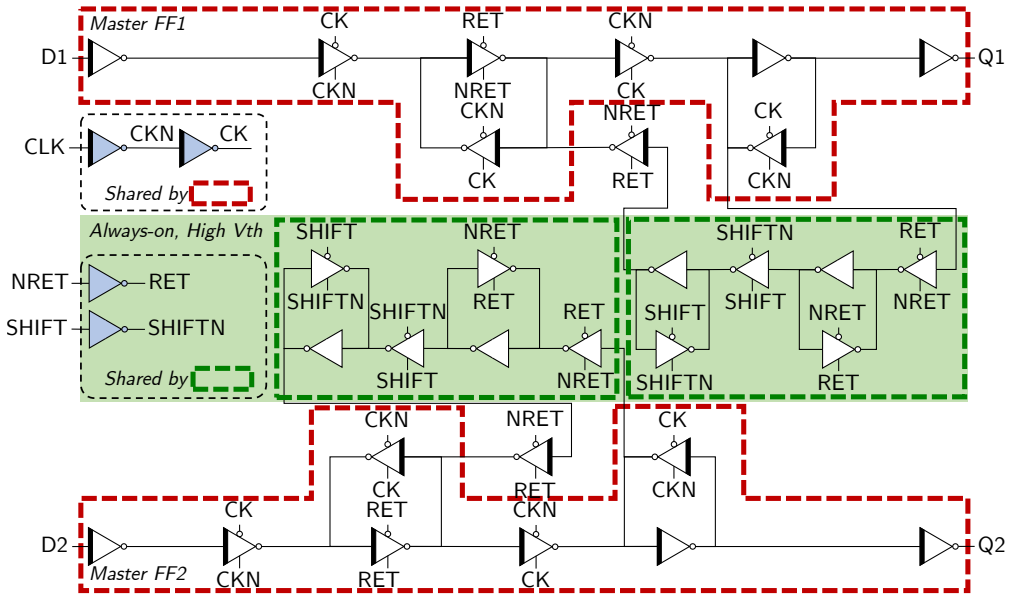


Figure 3.16: Schematic of a 2-bit/2-bit MBR-MBFF. The cells in light blue color are shared between two master flip-flops $FF1$ and $FF2$.

By merging two 2-bit MBRFFs, two always-on inverters for RET and $SHIFT$ are shared as well as the common inverters for the clock input CLK . Therefore it reduces not only the dynamic power consumption due to the internal clock inverters but also the leakage power consumption due to the always-on inverters by the extra pin $SHIFT$ during the sleep mode.

Fig. 3.17 shows the comparisons of the sleep power consumption and the area of a MBR-MBFF between the conventional design of [14] and the proposed design. The size of the retention storage is 2-bit while the the size of the master flip-flop is increased from 1-bit to 8-bit.

Compared with the conventional design, the sleep power consumption of the proposed design is almost the same regardless of the multi-bit size. The proposed design requires one more always-on inverter for the extra pin $SHIFT$ however the impact of

the additional inverter is negligible for the sleep power consumption. On the other hand, in the case of the area, the proposed design is 10.5% less smaller on average than the conventional design that includes the inverter chain.

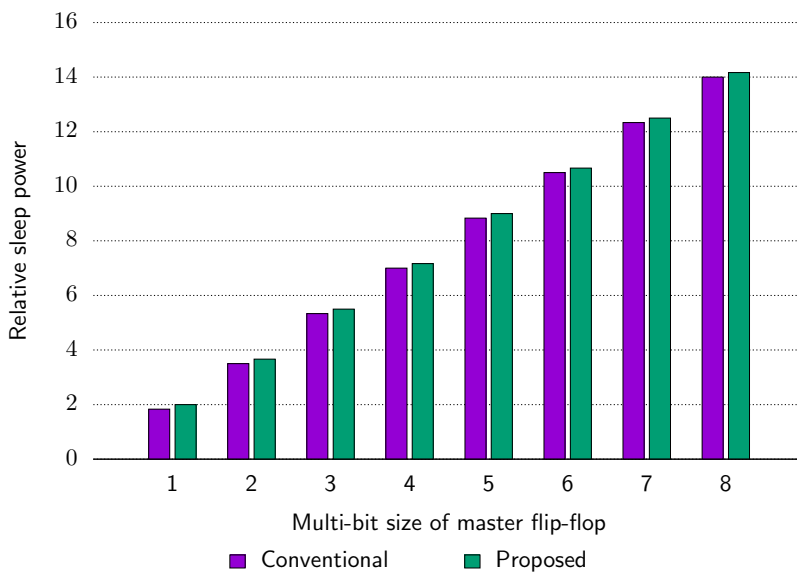
Fig. 3.18 shows the overall flow of our design methodology using MBR-MBFFs. For the given gate-level netlist \mathcal{C} , our three-step MBRFF allocation algorithm is performed with the MBRFF library cells. And then with the generated netlist \mathcal{C}' and our MBR-MBFF library cells, multiple MBRFFs are grouped into a single MBR-MBRFF like the traditional MBFF flow at the physical implementation stage.

3.4 Experiments

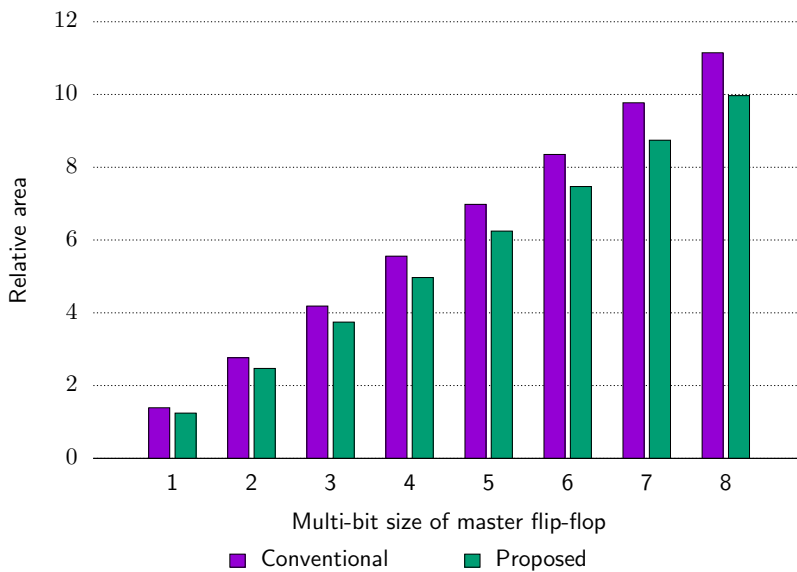
3.4.1 Experimental Setup

We implemented the proposed approach in C++ and Python with the *igraph-python* package [26] for graph analysis. In addition, to compare our results with that produced by the previous ILP-based MBRFF methods [18], we implemented them with the Gurobi Optimizer [27] as an ILP solver. We tested our allocation approach and the conventional approaches using IWLS benchmark circuits [22]. The benchmark circuits were synthesized and implemented by Synopsys *Design Compiler* and *IC Compiler* with Synopsys 32/28 nm Generic Library. We set the operating clock frequency to 200 MHz for all circuits and the target utilization ratio for the core area of those placements was 70%. We also implemented the logic and physical library of our 2-bit MBRFF and the corresponding multi-bit flip-flops. The maximum number of single flip-flops to group into multi-bit flip-flops was 8. To replace single-bit flip-flops with multi-bit flip-flops, we used the conventional placement-aware multi-bit register banking flow provided by *IC Compiler*.

Table 3.4 shows the detailed information of benchmark circuits including the number of flip-flops (“# of FFs”), the number of dependencies among flip-flops (“# of edges”), and the percentage of self-loop FFs (“% of self-loops”).



(a) Power



(b) Area

Figure 3.17: Comparison of power and area of a MBR-MBFF between the conventional design and the proposed design. The size of the retention storage is 2-bit.

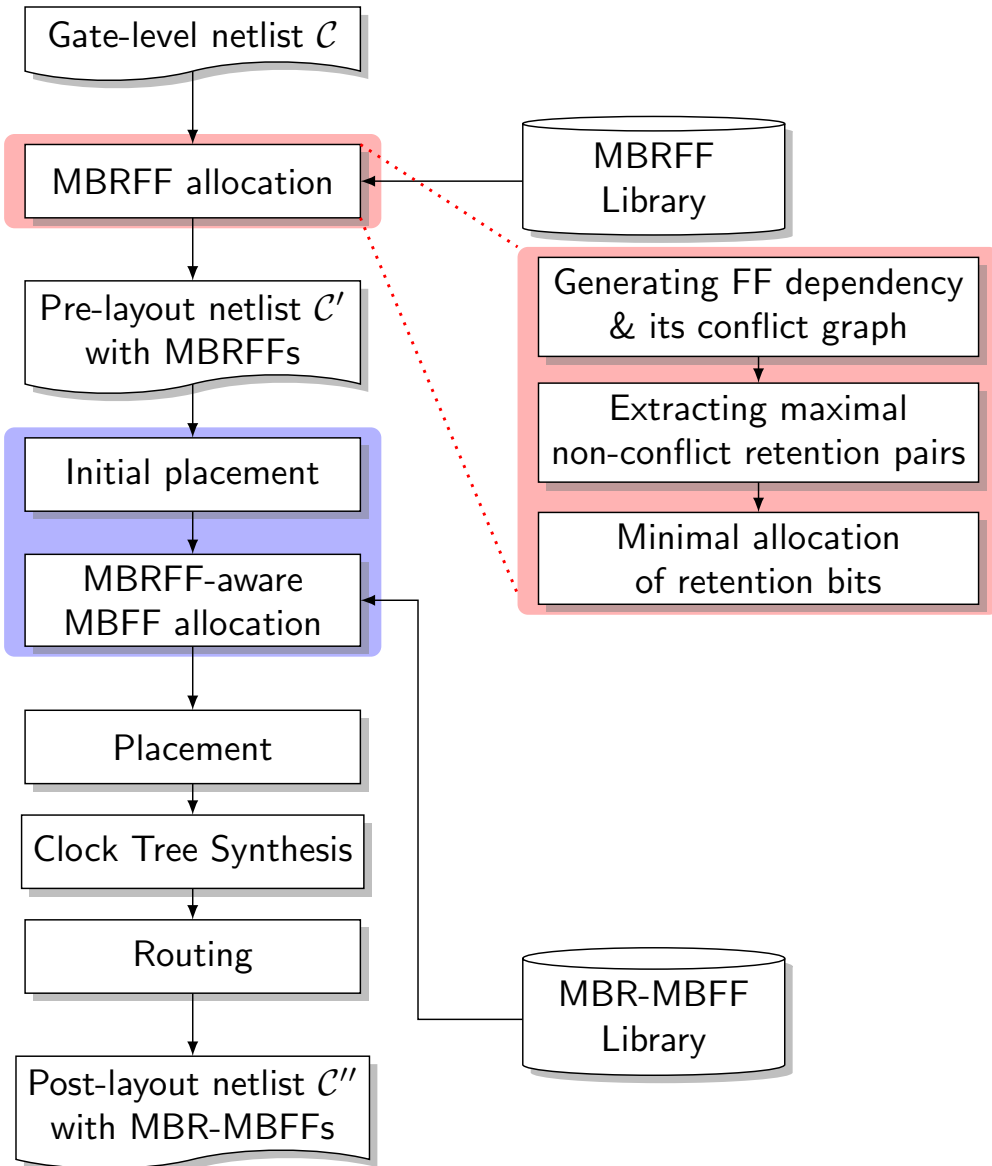


Figure 3.18: Design flow of the proposed MBR-MBFF approach.

Table 3.4: Benchmark circuit information.

Circuit	# of FFs	# of edges	% of self-loops
SPI	229	3690	84.72%
WB_DMA	523	7351	61.95%
AES_CORE	530	7198	24.91%
WB_CONMAX	818	12174	50.86%
MEM_CTRL	1118	59954	77.28%
USB_FUNCT	1739	19876	69.29%
AC97_CTRL	2199	14891	77.72%

3.4.2 Experimental Results

Table 3.5 shows the number of retention flip-flops (SBRFFs or MBRFFs) (“# RFFs”) and the total storage size for state retention (“# RetentionBits”), the wakeup latency (“Latency”), and the percentage of reduction of retention bits over the total bits of all flip-flops (“% Reduction”) of Fan *et al.*’s ILP based approach [18] with *Flow option 1* and *Flow option 2*, and our proposed approach. Note that we ran the previous approach with various sizes of the wakeup latency for the given circuits and used the best results while the wakeup latency of our approach is less than or equal to two cycles. In addition, it should be noted that the ordinary flip-flops which are driven only by the primary inputs are not counted for a fair comparison. When comparing with the non-uniform MBRFF approach [18] with *Flow option 1*, our proposed approach reduced the retention storage by 11.9% on average. Furthermore, compared with the non-uniform MBRFF approach [18] with *Flow option 2*, our approach reduced the retention storage by 9.82% (= 12.97% - 3.15%) on average.

We also compared the power consumption and the area used by the conventional SBRFF approach and our proposed MBRFF approach, and their MBFF extensions.

Table 3.5: Comparison of the previous non-uniform MBRFF approaches ([18]) with flow options 1 and 2, and our proposed approach.

Circuit	<i>Non-uniform</i> MBRFF replacement [18] with Flow option 1				<i>Non-uniform</i> MBRFF replacement [18] with Flow option 2				Our MBRFF approach			
	# RFFs	# RetentionBits	Latency	% Reduction	# RFFs	# RetentionBits	Latency	% Reduction	# RFFs	# RetentionBits	Latency	% Reduction
spi	197	223	5	2.62%	229	229	2	0.00%	197	197	2	13.97%
wb_dma	372	504	4	3.63%	372	473	4	9.56%	392	423	2	16.63%
aes_core	530	530	6	0.00%	521	525	3	0.94%	393	425	2	19.81%
wb_conmax	818	818	4	0.00%	690	818	3	0.00%	690	708	2	13.45%
mem_ctrl	1118	1118	6	0.00%	980	1049	5	6.17%	985	1022	2	8.59%
usb_funcnt	1739	1739	10	0.00%	1481	1688	4	2.93%	1515	1707	2	1.84%
ac97_ctrl	1937	2171	14	1.27%	2111	2146	14	2.41%	2040	2137	2	2.82%
Avg.	959	1015	7	1.08%	912	990	5	3.15%	889	928	2	12.97%

Table 3.6 indicates the comparisons of the active and the sleep power consumption. The active power is the power consumption in the normal operation mode and the sleep power means the leakage power consumption in the retention mode during the sleep mode.

When the MBFF extension was used, as shown in Table 3.6, the active power of the SBRFF-based designs and the proposed MBRFF-based designs were reduced similarly because, in the active mode, the dynamic power was larger than the sleep power so it dominated the total power consumption and the dynamic power was reduced by the MBFF extension in both approaches. The sleep power was also reduced in both approaches because the number of always-on cells for control signals of retention flip-flops was decreased by merging registers. However, in the case of the sleep power consumption, the MBR-MBFF approach reduced about 31.2% and 8.5% on average over that produced by the conventional SBRFF approach (SBRFF) and its MBFF extension (SBR-MBFF).

Table 3.6: Comparison of active power and sleep power consumption used by SBRFF (conventional single bit retention flip-flop allocation for single bit flip-flops), SBR-MBFF (conventional single bit retention flip-flop allocation for multi-bit flip-flops), MBRFF (our multi-bit retention flip-flop allocation for single-bit flip-flops), and MBR-MBFF (our multi-bit retention flip-flop allocation for multi-bit flip-flops).

Circuit	Active Power (μW)				Sleep Power (μW)			
	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF
SPI	574.3	549.6 (4.30%)	545.3 (5.05%)	539.4 (6.08%)	89.3	84.9 (4.90%)	82.7 (7.40%)	75.5 (15.49%)
WB_DMA	1248.0	904.0 (27.56%)	1215.0 (2.64%)	978.5 (21.59%)	222.0	143.1 (35.54%)	176.3 (20.59%)	135.3 (39.05%)
AES_CORE	3614.0	3395.0 (6.06%)	3389.0 (6.23%)	3162.0 (12.51%)	281.2	256.6 (8.75%)	225.5 (19.81%)	165.6 (41.11%)
WB_CONMAX	6011.0	5504.0 (8.43%)	5855.0 (2.60%)	5529.0 (8.02%)	970.3	701.6 (27.69%)	891.7 (8.10%)	664.5 (31.52%)
MEM_CTRL	1768.0	1493.0 (15.55%)	1631.0 (7.75%)	1379.0 (22.00%)	403.2	320.4 (20.54%)	358.0 (11.21%)	247.5 (38.62%)
USB_FUNCT	2766.0	2459.0 (11.10%)	2521.0 (8.86%)	2371.0 (14.28%)	711.9	649.6 (8.75%)	622.9 (12.50%)	579.1 (18.65%)
AC97_CTRL	3198.0	2378.0 (25.64%)	3134.0 (2.00%)	2394.0 (25.14%)	705.9	457.9 (35.13%)	696.1 (1.39%)	460.7 (34.74%)
Avg.	2739.9	2383.2 (13.02%)	2612.9 (4.64%)	2336.1 (14.74%)	483.4	373.4 (22.75%)	436.2 (9.77%)	332.6 (31.20%)

(a) Total power consumption

Circuit	Active Power (μW)				Sleep Power (μW)			
	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF
SPI	425.3	402.0 (5.48%)	399.1 (6.16%)	409.0 (3.83%)	18.4	17.9 (2.77%)	15.9 (13.94%)	15.6 (15.63%)
WB_DMA	751.3	516.1 (31.31%)	722.6 (3.82%)	568.6 (24.32%)	42.1	39.0 (7.41%)	37.7 (10.43%)	33.6 (20.24%)
AES_CORE	1645.0	1497.0 (9.00%)	1461.0 (11.19%)	1369.0 (16.78%)	42.7	40.9 (4.06%)	35.1 (17.82%)	33.5 (21.59%)
WB_CONMAX	846.4	668.6 (21.01%)	781.5 (7.67%)	718.7 (15.09%)	62.0	57.6 (7.11%)	53.6 (13.52%)	50.0 (19.29%)
MEM_CTRL	1155.0	980.9 (15.07%)	1086.0 (5.97%)	959.1 (16.96%)	85.7	80.7 (5.83%)	79.0 (7.86%)	73.2 (14.58%)
USB_FUNCT	1784.0	1537.0 (13.85%)	1627.0 (8.80%)	1508.0 (15.47%)	140.2	137.1 (2.21%)	142.5 (-1.64%)	139.7 (0.36%)
AC97_CTRL	2288.0	1827.0 (20.15%)	2217.0 (3.10%)	1806.0 (21.07%)	177.0	165.9 (6.27%)	176.1 (0.51%)	164.7 (6.95%)
Avg.	1270.7	1061.2 (16.49%)	1184.9 (6.75%)	1048.3 (17.50%)	81.2	77.0 (5.10%)	77.1 (4.98%)	72.9 (10.19%)

(b) Register-only power consumption

Table 3.7 shows the area used by the conventional SBRFF-based designs and our proposed MBRFF-based designs with their MBFF extensions. Compared with the conventional SBRFF approach without MBFF extension (SBRFF), our MBRFF-based design without MBFF extension (MBRFF) reduced the total area by 4% on average. Furthermore, our MBRFF-based design with MBFF extension (MBR-MBFF) reduced the area by 12.5% on average. It was because that the area of always-on cells for control signals of retention flip-flops were reduced effectively in the MBFF extension.

For the MBR-MBFF based designs, the detailed information of usage of multi-bit flip-flops is shown in Table 3.8. It shows the number of 1-bit FFs (“# 1-bit FFs), 2 to 4-bit MBR-MBFFs, and 5 to 8-bit MBR-MBFFs. The multi-bit banking ratio means the ratio of the number of total bits of MBR-MBFFs to the number of total bits of all flip-flops in the design.

Fig. 3.19 shows the power breakdown used by our MBRFF approach with no MBFF extension and our MBR-MBFF approach during the sleep mode. The power management cells indicate the power switch cells and the isolation cells. The always-on cells means the always-on buffers for control signals of retention flip-flops. By the MBFF extension, the power consumption of the flip-flops was reduced by about 5.5% while the power consumption of the always-on cells was reduced by 33.9% compared to those of the MBRFF approach with no MBFF extension.

In addition, Fig. 3.20 shows the comparison of total area of always-on cells of our MBRFF approach with no MBFF extension and our MBR-MBFF approach. Compared with the MBRFF approach with no MBFF extension, Our MBR-MBFF approach decreased the total area of always-on cells by 9.6% on average.

Consequently, the power and area overhead caused by the extra input pin in our MBRFF design (i.e., *SHIFT*) were decreased effectively by the MBFF extension.

Table 3.7: Comparison of total area and cell area used by SBRFF (conventional single bit retention flip-flop allocation for single bit flip-flops), SBR-MBFF (conventional single bit retention flip-flop allocation for multi-bit flip-flops), MBRFF (our multi-bit retention flip-flop allocation for single-bit flip-flops), and MBR-MBFF (our multi-bit retention flip-flop allocation for multi-bit flip-flops).

Circuit	Total Area (μm^2)				Cell Area (μm^2)			
	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF	SBRFF	SBR-MBFF	MBRFF	MBR-MBFF
SPI	8969.2	8726.5 (2.71%)	8639.7 (3.67%)	8435.9 (5.95%)	7384.2	7444.4 (-0.82%)	7117.3 (3.61%)	7103.8 (3.80%)
WB_DMA	17084.1	14593.7 (14.58%)	15459.1 (9.51%)	14093.5 (17.51%)	13805.6	13045.7 (5.50%)	12520.2 (9.31%)	12179.6 (11.78%)
AES_CORE	39344.1	38070.0 (3.24%)	37912.5 (3.64%)	36201.4 (7.99%)	30535.1	30538.7 (-0.01%)	29266.5 (4.15%)	28639.7 (6.21%)
WB_CONMAX	117481.0	107551.9 (8.45%)	116096.6 (1.18%)	106789.9 (9.10%)	75331.8	76573.6 (-1.65%)	73784.4 (2.05%)	75360.3 (-0.04%)
MEM_CTRL	32329.5	27789.6 (14.04%)	30914.1 (4.38%)	26427.4 (18.26%)	25336.6	24645.9 (2.73%)	24145.0 (4.70%)	23258.5 (8.20%)
USB_FUNCT	53480.2	50893.5 (4.84%)	51213.8 (4.24%)	49223.3 (7.96%)	43013.1	42898.2 (0.27%)	41059.3 (4.54%)	41108.8 (4.43%)
AC97_CTRL	60287.7	48011.5 (20.36%)	59250.5 (1.72%)	47655.7 (20.95%)	45007.6	42446.4 (5.69%)	44194.4 (1.81%)	41889.3 (6.93%)
Avg. reduction	—	(9.75%)	(4.05%)	(12.53%)	—	(1.67%)	(4.31%)	(5.90%)

Table 3.8: Multi-bit information of the MBR-MBFF based designs.

Circuit	# 1-bit FFs	# 2 to 4-bit FFs	# 5 to 8-bit FFs	Total bits of MBFFs	Multi-bit banking ratio
SPI	182	1	6	229	20.5%
WB_DMA	152	24	47	523	70.9%
AES_CORE	270	14	33	530	49.1%
WB_CONMAX	147	100	51	770	80.9%
MEM_CTRL	298	53	91	1065	72.0%
USB_FUNCT	1347	33	43	1741	22.6%
AC97_CTRL	453	117	214	2199	79.4%
Avg.	407	49	69	1008	56.5%

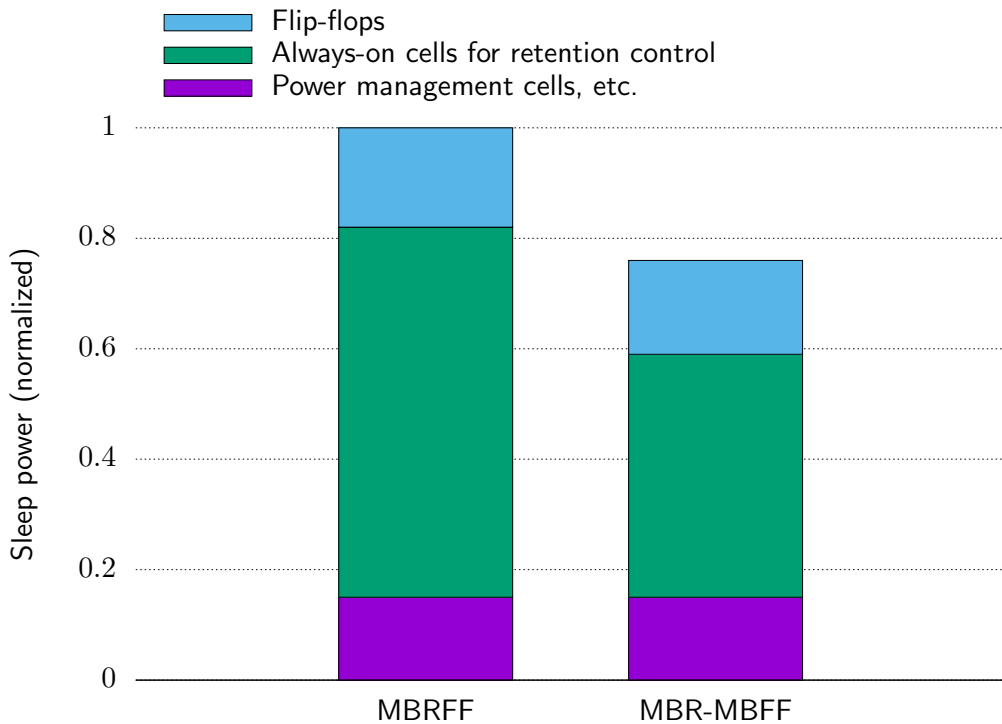


Figure 3.19: Sleep power breakdown of our MBRFF approach without MBFF extension (MBRFF) and MBRFF approach with MBFF extension (MBR-MBFF).

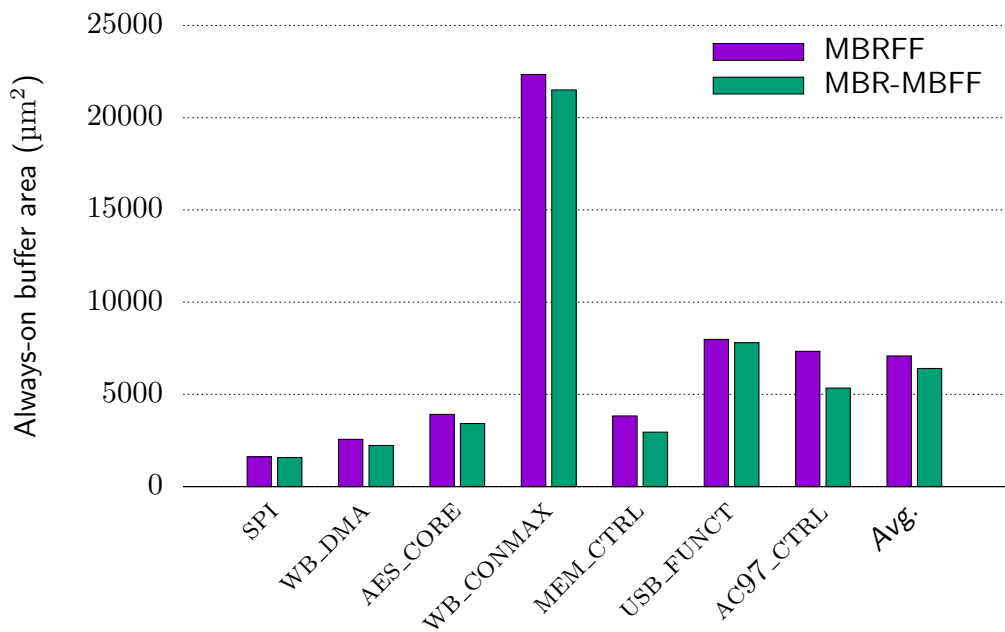


Figure 3.20: Comparison of always-on cell area of our MBRFF approach without MBFF extension (MBRFF) and MBRFF approach with MBFF extension (MBR-MBFF).

Chapter 4

CONCLUSIONS

4.1 Flip-flop State Driven Clock Gating: Concept, Design, and Methodology

This work proposed a novel clock gating method to overcome the inherent and critical limitation of the prevalent input data toggling driven clock gating. Precisely, (1) we proposed a new clock gating method called *flip-flop state driven clock gating* which completely eliminates the essential and expensive component of XOR gates for detecting input toggling of flip-flops; (2) we provided the supporting logic circuitry of our proposed XOR-free clock gating, confirming its safe applicability through a comprehensive timing analysis; (3) we proposed, based on the flip-flops' state profile, a clock gating methodology that seamlessly combines our flip-flop state based clock gating with the toggling based clock gating. Through experiments with benchmark circuits, it was confirmed that our state driven clock gating method is very effective, reducing the power on average by 7.59% further over the toggling driven clock gating.

4.2 Algorithm and Design Optimization of Allocating Multi-bit Retention Flip-flops for Power Gated Circuits

This work proposed a practical solution to the problem of state retention flip-flops for power gated circuits. To overcome the limitations of the previous approaches, the long wakeup delay and the degradation of reduction performance due to flip-flops with mux-feedback loop, we introduced a concept of 2-phase retention control scheme and retention pairs of flip-flops to practically reduce the required storage for state retention while the wakeup latency is constrained up to two clock cycles. With the proposed control scheme, we formulated the problem into an independent set based problem and developed an effective heuristic algorithm. Our experiment results showed that the proposed approach can reduce the state retention storage by 9.8% on average compared with the state-of-the-art MBRFF allocation for practical designs containing self-loop FFs while the wakeup delay is limited up to two clock cycles. In addition, we proposed a new design of a multi-bit retention flip-flop and its multi-bit flip-flop extension to address the overhead problem of the internal inverter chain of the conventional multi-bit retention flip-flop design. With the proposed multi-bit design, the sleep power consumption and the area are reduced by about 31.2% and 12.5%, respectively, compared with those of the single-bit retention flip-flop approach.

Bibliography

- [1] V. Oklobdzija, V. Stojanovic, D. Markovic, and N. Nedovic, *Digital System Clocking: High-Performance and Low-Power Aspects*, Wiley-IEEE Press, 2003.
- [2] J. S. M. Rao, J. Srinivas, P. Vishwanath, U. H, and J. Rao, “Clock Gating for Power Optimization in ASIC Design Cycle Theory & Practice,” In *Proceedings of ACM/IEEE International Symposium on Low Power Electronics & Design*, pp. 307–308, August 2008.
- [3] S. Wimer and I. Koren, “Design flow for flip-flop grouping in data-driven clock gating,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 4, pp. 771–778, April 2014.
- [4] G. Palumbo, F. Pappalardo, and S. Sannella, “Evaluation on power reduction applying gated clock approaches,” In *IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, vol. 4, pp. IV–IV, May 2002.
- [5] L. Benini, A. Bogliolo, and G. De Micheli, “A survey of design techniques for system-level dynamic power management,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, June 2000.
- [6] Q. Wang and S. Roy, “Power minimization by clock root gating,” In *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference*, pp. 249–254, January 2003.

- [7] L. Benini, P. Siegel, and G. De Micheli, "Saving power by synthesizing gated clocks for sequential circuits," *IEEE Design Test of Computers*, vol. 11, no. 4, pp. 32–41, 1994.
- [8] G. Yang and T. Kim, "Design and algorithm for clock gating and flip-flop co-optimization," In *IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–6, November 2018.
- [9] I. Han, J. Kim, J. Yi, and Y. Shin, "Register grouping for synthesis of clock gating logic," In *International Conference on IC Design and Technology*, pp. 1–4, June 2016.
- [10] R. Fraer, G. Kamhi, and M. K. Mhameed, "A new paradigm for synthesis and propagation of clock gating conditions," In *ACM/IEEE Design Automation Conference*, pp. 658–663, June 2008.
- [11] Y. Shin, J. Seomun, K. Choi, and T. Sakurai, "Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 15, no. 4, pp. 28:1–28:37, October 2010.
- [12] E. Choi, C. Shin, T. Kim, and Y. Shin, "Power-gating-aware high-level synthesis," In *Proceedings of ACM/IEEE International Symposium on Low Power Electronics & Design*, pp. 39–44, 2008.
- [13] K. Michael, F. David, A. Rob, G. Alan, and S. Kaijian, *Low Power Methodology Manual: For System-on-Chip Design*, Springer Publishing Company, Incorporated, 2007.
- [14] Y. Chen, Y. Shi, K. Lai, G. Hui, and S. Chang, "Efficient multiple-bit retention register assignment for power gated design: Concept and algorithms," In *IEEE/ACM International Conference on Computer-Aided Design*, pp. 309–316, November 2012.

- [15] A. Darbari, B. M. A. Hashimi, D. Flynn, and J. Biggs, “Selective state retention design using symbolic simulation,” In *Design, Automation Test in Europe Conference Exhibition*, pp. 1644–1649, April 2009.
- [16] 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, March 2015.
- [17] Y. Chen, H. Geng, K. Lai, Y. Shi, and S. Chang, “Multibit retention registers for power gated designs: Concept, design, and deployment,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 4, pp. 507–518, April 2014.
- [18] G. Fan and M. P. Lin, “State retention for power gated design with non-uniform multi-bit retention latches,” In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 607–614, November 2017.
- [19] G. Hyun and T. Kim, “Flip-flop State Driven Clock Gating: Concept, Design, and Methodology,” In *IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–6, November 2019.
- [20] G. Hyun and T. Kim, “Allocation of State Retention Registers Boosting Practical Applicability to Power Gated Circuits,” In *IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–6, November 2019.
- [21] M. Müller, S. Simon, H. Gryska, A. Wortmann, and S. Buch, “Low power synthesizable register files for processor and IP cores,” *Integration*, vol. 39, no. 2, pp. 131–155, March 2006.
- [22] C. Albrecht, “IWLS 2005 benchmarks,” <http://iwls.org/iwls2005/benchmarks.html>, 2005.

- [23] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, V. Oklobdzija, V. Stojanovic, D. Markovic, and N. Nedovic, *Introduction to Algorithms, Third Edition*, Wiley-IEEE Press, 2009.
- [24] H. Moon and T. Kim, “Design and allocation of loosely coupled multi-bit flip-flops for power reduction in post-placement optimization,” In *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, pp. 268–273, 2016.
- [25] S. Lin and M. P. Lin, “More effective power-gated circuit optimization with multi-bit retention registers,” In *IEEE/ACM International Conference on Computer-Aided Design*, pp. 213–217, November 2014.
- [26] G. Csardi and T. Nepusz, “The igraph software package for complex network research,” *InterJournal*, vol. Complex Systems, p. 1695, 2006.
- [27] LLC. Gurobi Optimization, “Gurobi optimizer reference manual,” <http://www.gurobi.com>, 2018.

초 록

저전력 설계는 최신 시스템-온-칩 (SoCs) 설계에서 매우 중요한 요소 중의 하나이다. 본 논문에서는 동적 및 정적 전력 소비를 감소시키기 위한 저전력 설계 방법론에 대해 논한다. 구체적으로 비용 효율적인 저전력 설계를 위하여 두 가지 새로운 기술을 제안한다.

우선 본 논문에서는 동적 전력 소비를 줄일 수 있는 새로운 클럭 게이팅 방법을 제안한다. 기존 플립-플롭 입력 데이터 토글 기반 클럭 게이팅은 가장 널리 사용되는 클럭 게이팅 기법 중의 하나이다. 하지만 이 방법은 더 많은 플립-플롭에 대해 적용할수록 클럭 게이팅에 필요한 부가 회로가 급격히 증가한다는 근본적인 한계를 지니고 있다. 이러한 한계를 극복하기 위하여 본 논문에서는 다음과 같이 새로운 클럭 게이팅 방법을 제안한다. 첫 번째로 기존 입력 데이터 토글 기반 클럭 게이팅 방법에 필요한 회로 자원을 분석하여 해당 방법의 비효율성을 보이고, 기존 방법에서 사용되는 입력 데이터 토글 검출에 필수적이지만 고비용의 XOR 게이트를 완벽히 제거한 ‘플립-플롭 상태 기반 클럭 게이팅’이라는 새로운 클럭 게이팅 방법을 제안한다. 두 번째로 제안된 XOR 게이트가 필요 없는 클럭 게이팅 방법을 위한 부가 회로를 제시하며, 다양한 타이밍 분석을 통하여 해당 회로가 안정적으로 적용될 수 있음을 보인다. 세 번째로 회로의 플립-플롭 상태 프로파일에 기반하여, 제안된 클럭 게이팅 기법을 기존 클럭 게이팅 기법과 완벽하게 통합할 수 있는 클럭 게이팅 방법론을 제안한다. 여러 벤치마크 회로에 대한 실험 결과는 기존 입력 데이터 토글 기반 클럭 게이팅 방법이 전력 소비 절감 기회를 놓치는 반면 본 논문에서 제안된 방법은 모든 타이밍 제약 조건을 만족하면서 전력 소비 감소에 매우 효과적임을 보여준다.

다음으로 정적 전력 소비를 줄이기 위한 방안으로, 본 논문에서는 기존 파워 게이트 회로의 상태 보존용 저장 공간 할당 방법들이 지니고 있는 두 가지 중요한 한계들을 해결할 수 있는 방법을 제안한다. 중요한 한계들이란 첫 번째로 다중-비트 상태 보존 플립-플롭의 무분별한 사용으로 인한 긴 웨이크업 지연 시간이며, 두 번째로 멀티플렉서 되먹임 루프가 있는 상태 보존 플립-플롭의 최적화 불가능성이다. 기존 방법들에서는 상태 보존을 위한 저장 공간을 최소화하기 위해 긴 웨이크업 지연 시간이 필수적이었다. 그리고 되먹임 루프가 있는 플립-플롭은 최적화할 수 없는 대상으로 다루어졌다. 그러나 일반적으로 하드웨어 기술 언어(HDL)로부터 생성되는 되먹임 루프를 지닌 플립-플롭은 무시할 수 있을 정도로 적은 양이 아니다. 첫 번째 한계를 해결하기 위한 방법으로 본 논문에서는 최대 2 비트의 다중-비트 상태 보존 플립-플롭을 사용하여 웨이크업 지연 시간을 두 클럭 사이클로 제한하면서도 상태 보존을 위한 저장 공간을 효율적으로 절약할 수 있음을 보인다. 그리고 두 번째 한계를 극복하기 위해서 되먹임 루프를 지닌 플립-플롭이 포함된 두 플립-플롭 쌍의 상태를 복원할 수 있는 2단 상태 보존 제어 방안을 제안한다. 또한 주어진 회로에서 충돌없이 동시에 존재할 수 있는 플립-플롭 쌍을 최대로 추출하기 위해 독립 집합 문제(independent set problem)기반의 연산법도 제안한다. 벤치마크 회로에 대한 실험 결과는 본 논문에서 제안된 방법이 웨이크업 지연 시간을 두 클럭 사이클로 제한하면서도 상태 보존에 필요한 저장 공간과 파워를 감소시키는데 매우 효과적임을 보여준다.

주요어: 저전력, 클럭 게이팅, 파워 게이팅, 상태 보존, 상태 보존 플립-플롭

학번: 2016-30222