INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

REDUCING CLOCK POWER IN SYNCHRONOUS CIRCUIT DESIGN

by

Yi Zhang

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering,
Department of Electronics,
Carleton University,
Ottawa, Ontario, Canada

August 10, 1998

© Yi Zhang 1998

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-36904-8

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

“Reducing Clock Power In Synchronous Circuit Design”

submitted by Yi Zhang, B.Eng.
in partial fulfillment of the requirements for
the degree of Master of Engineering

Thesis Supervisor

Chair, Department of Electronics

for my wife Yu Xu and my parents

Abstract

In this thesis, a novel approach of reducing the activities of active synchronous circuits is proposed. The proposed double-level-sensitive design methodology utilizes both levels of the clock signal and makes small flip-flop like circuits with multiple outputs (mop-flops). By making the mop-flops into standard cells which can be used like flip-flops, larger circuits can be built. It allows the designer to construct synchronous circuits of the same speed with only half the clock speed plus reduced clock transitions inside the mop-flop. The half clock rate reduces the power consumption not only for the synchronous circuits, but also for the clock distribution network. The timing, race, hazards, synchronization and potential false clock edge issues when making the mop-flops are also discussed.

The double-level-sensitive method can also work with conventional flip-flops. As an example, an N-bit up/down Gray code counter is designed which shows both simplicity and power efficiency advantages over existing circuits.

Table Of Contents

CHAPTER 1 INTRODUCTION	1
CHAPTER 2 BACKGROUND	4
2.0 Introduction	4
2.1 The Basis Of CMOS	5
2.1.1 Dynamic Power Consumption	6
2.1.1.1 Load Capacitance Transient Dissipation	6
2.1.1.2 Current Spiking During Switching	7
2.1.2 Leakage Power Consumption	8
2.2 Power Saving from Device-Level Concepts	9
2.2.1 Run The Circuit At Minimum Supply Voltage	9
2.2.2 Reduce Capacitance Load	10
2.2.2.1 Reduce Capacitance Load By Scaling Down The Devices	10
2.2.2.2 Reduce Capacitance Load By Rearranging The Order Of Input Signals	11
2.2.3 Increase Threshold Voltage	12
2.3 Power Saving from System-Level Concepts	13
2.3.1 Clock-Gating Logic	14
2.4 Power Saving From Logic-Level Concepts	15
2.4.1 Power Saving By Optimum State Encoding	15
2.4.2 Power Saving By Optimizing The Flip-flops	16
2.4.3 Power Saving By Using Latches	18
2.4.3.1 Example of Latch Based Circuit	21
2.5 Different Architectures Of FSMs	22
2.5.1 The Proposed FSM Architectures	23
2.5.1.1 The DLS Mop-Flop Architecture	23
2.5.1.2 The DLS FSM With Flip-Flop Architecture	24
CHAPTER 3 MAKING BETTER FLIP-FLOPS	25
3.0 Introduction	25
3.1 The Conventional DFF	27
3.1.1 The DFF Is An FSM	27
3.1.2 The Clock Induced Power Dissipation In DFF	28
3.1.3 The Clock Induced Power Dissipation In The DFF With Enable	31
3.2 The Clock Gating Enabled Flip-Flop Method	31
3.3 The Intrinsic Enabled Flip-Flop Method	33
3.3.1 The Intrinsic Enabled DFF	33
3.3.2 The Intrinsic Enabled T Flip-Flop (ETFF)	34
3.3.2.1 The Schematic Of The Intrinsic ETFF	35
3.3.2.1.1 Synchronous vs. Asynchronous	36
3.3.2.1.2 Races And Hazards In An Asynchronous Circuit	38
3.3.2.2 The Performance Of The Intrinsic ETFF	40
3.3.2.2.1 The Power Dissipation	40
3.3.2.2.2 The Speed	41

3.4 When To Use Which Flip-flop	41
3.5 Example	42
3.6 Summary	43
CHAPTER 4 DOUBLE-LEVEL-SENSITIVE MOP-FLOPS	44
4.0 Introduction	44
4.0.1 Brief Description Of Mop-flops	45
4.1 The Nature Of SHE	46
4.1.1 Example Circuits With Varies SHEs	48
4.2 Information Contained In Clock	49
4.3 Design DLS FSMs	50
4.3.1 Rules Of Synchronous Transition	50
4.3.2 State Diagrams For The DLS FSMs	52
4.3.3 State Tables For The DLS FSMs	54
4.3.3.1 State Table Representation	54
4.3.3.2 Changing State	54
4.4 The DLS Mop-flops	55
4.5 Making Bigger Mop-flops	56
4.5.1 Interfacing The Counter With Conventional Circuits	61
4.6 Synthesizing DLS Mop-flop Method	61
4.7 The Performance Comparison	63
4.7.1 The Size Comparison	63
4.7.2 The Speed Comparison	64
4.7.3 The Power Comparison	65
4.8 The Use Of Mop-Flop Type Gray-Code Counters	66
4.9 DLS Controllers	67
4.9.1 The Traffic Light Example	68
4.9.2 The Game Controller Example	71
4.9.3 Concerns Of The DLS State Diagram Generating Methods	75
4.10 Mop-flops Based Circuits	77
4.10.1 Making Longer Counters	77
4.10.2 A Pure Mop-flop Gray-Code Up/Down Counter Module	79
4.10.3 General FSMs With Up/Down Mop-flop Gray-code Modules	84
4.11 Summary	86
CHAPTER 5 DOUBLE-EDGE-TRIGGERED FSMs	88
5.0 Introduction	88
5.1 Count_to_8 Circuit With Two DFFs	89
5.2 The Synchronous N-Bit Gray Code Counter Module	91
5.3 The Synchronous N-Bit Gray Code Up/Down Counter Module	95
5.4 The Performance Of The Circuit	96
5.5 Applications Of Gray Code	96
5.6 Summary	98
CHAPTER 6 SUMMARY AND CONCLUSION	99
6.1 Thesis Summary	99

6.2 Thesis Contributions	101
6.2.1 The Enabled Flip-flop Method	101
6.2.2 The DLS Method	102
6.2.2.1 The Mop-flop	102
6.2.2.2 The Mop-Flop Module Based Circuits	104
6.2.2.3 DET Circuit	104
6.3 Future Work	105
References	106
Appendix A	108
Appendix B	114
Appendix C	117

List Of Tables

Table 1 Power comparison of the two DFFs shown in Figure 2.6	18
Table 2 Comparison of latch based and DFF based circuits	22
Table 3 Power dissipation estimation for circuits in Figure 3.1 and Figure 3.5(a)	32
Table 4 Power dissipation of the two proposed enabled flip-flops	40
Table 5 Size comparison of DLS and conventional Gray-code counters	64
Table 6 Speed comparison of DLS mop-flop and conventional Gray-code counters	64
Table 7 Power comparison of DLS mop-flop and conventional Gray-code counters	65
Table 8 Comparisons of the traffic light controllers	71
Table 9 Comparisons of game controllers	75
Table 10 Comparisons of seven-bit counters	79
Table 11 Maximum saving achieved by mop-flop designs over conventional designs	86
Table 12 Comparison of various N-bit counter circuits	87
Table 13 Size and power comparisons of proposed 5-bit Gray code counter module vs. patented modules	96

List Of Figures

Figure 2.1	Basic CMOS model for dynamic power dissipation	5
Figure 2.2	Better put the mostly on gate at the bottom	12
Figure 2.3	A charge pump to bias the back gate (a) pump up (b) pump down	13
Figure 2.4	System-level clock gating structure	14
Figure 2.5	Basic FSM structure	16
Figure 2.6	DFFs (a) with transmission gates (b) 6-state DFF	17
Figure 2.7	Structure of D latch and its symbol	19
Figure 2.8	General structure of latch based FSM	19
Figure 2.9	State diagram of the excess-3 code FSM	21
Figure 2.10	Latch based excess-3 code FSM	22
Figure 2.11	The proposed FSM architectures	24
Figure 3.1	DFF with enable	26
Figure 3.2	The 6-state DFF	28
Figure 3.3	Internal nodes switching of a DFF ($D=Q=1$) under clock rising edge	29
Figure 3.4	Internal nodes switching of a DFF ($D=Q=0$) under clock rising edge	30
Figure 3.5	A gated flip-flop with (a) an AND gate (b) an OR gate	31
Figure 3.6	Timing diagram for the gated DFF with an AND gate	32
Figure 3.7	State diagram of the enabled DFF	34
Figure 3.8	State diagram of ETFF	35
Figure 3.9	Circuit schematic of ETFF	36
Figure 3.10	Static-1 hazard in the D latch	39
Figure 4.0	Feedback loops in (a) D master-slave flip-flop (b) ETFF	48
Figure 4.1	Timing in the form of (a) δ function (b) conventional clock signal	50
Figure 4.2	State diagram of a two-bit Gray-code counter	52
Figure 4.3	State diagram of a two-bit Gray-code counter as the DLS FSM	53
Figure 4.4	Schematic of the two-bit Gray-code counter	56
Figure 4.5	A three-bit Gray-code counter (a) state diagram (b) Symbol	57
Figure 4.6	(a) The schematic of the three-bit Gray-code counter	57
Figure 4.6	(b) Mux based three-bit Gray code counter	58
Figure 4.7	The schematic of a four-bit Gray-code counter	60
Figure 4.8	The use of dummy clock signal	62
Figure 4.9	The 2-bit DLS Gray-code counter generated by the SYNOPSIS	63
Figure 4.10	Count-to-10 Gray-code counter	67
Figure 4.11	State diagram for the traffic light controller[Hay93]	68
Figure 4.12	Modified state diagram for the traffic light controller	69
Figure 4.13	DLS traffic light controller	70
Figure 4.14	State diagram of the game controller	72
Figure 4.15	State diagram for DLS game controller	73
Figure 4.16	The DLS mop-flop game controller circuit	75
Figure 4.17	General structure of the ripple counter	77
Figure 4.18	Ripple mop-flop counter	78
Figure 4.19	State diagram of the four-bit up/down Gray-code counting mop-flop module	80
Figure 4.20	Four-bit Gray-code up/down counting mop-flop module	82
Figure 4.21	The direction control signal generation circuit	82

Figure 4.22	An eight-bit pure Gray-code up/down counter module	83
Figure 4.23	Mop-flops based FSM	84
Figure 4.24	Random FSM example	85
Figure 5.1	State diagram of count_to_8 circuit with only two flip-flops	90
Figure 5.2	Count_to_8 or 2-bit Gray code counter module circuit with only 2 flip-flops	91
Figure 5.3	State diagram of the 3-bit Gray code counter module	92
Figure 5.4	The schematic of the (a) 3-bit (b) 4-bit Gray code counter module	93
Figure 5.5	State diagram of 4-bit Gray code counter module	94
Figure 5.6	The interface between asynchronous world and synchronous world	97

DEFINITIONS

DET: Double-edge-triggered. Triggering on both rising and falling edges of the clock.

DLS: Double-level-sensitive. Triggering on both levels of the clock.

DFF: D flip-flop.

FSM: Finite-State Machine.

SHE: State-holding elements. Flip-flop, latches or feedback loops where the state variable is stored.

mop-flop: Multiple output flip-flop. A flip-flop with multiple independent synchronous outputs.

synchronous signal: A signal that never changes near the clock edge.

asynchronous state-table: A state table for either a clockless sequential circuit or a clocked circuit in which the clock is treated by the state table in exactly the same way as any other inputs.

synchronous transition: A state transition that only responds directly or indirectly to clock change.

asynchronous transition: A state transition that may respond to a non-clock change.

synchronous state: If state X of an FSM can only be reached after a clock edge¹ or in direct response to a *synchronous transition*, this state is defined as *synchronous state*.

asynchronous state: If state Y of an FSM can also be reached after non-clock signal's edge or in direct response to an *asynchronous transition*, this state is defined as an *asynchronous state*.

asynchronous circuit: If an FSM's output is taken from *asynchronous states*, this FSM is an *asynchronous circuit*.

synchronous circuit: If an FSM's output is taken from *synchronous states* and/or inputs which are synchronous, this FSM is an *synchronous circuit*.

externally-synchronous circuit: A flip-flop which externally behaves like a synchronous circuit. Refer to section 3.3.2.1.1.

1. Transients caused by X/clock edge, which can logically interact with the clock/X to affect storage feedback loops, must have died out before the clock edge/X changes. This determine the setup and hold times.

CHAPTER 1

INTRODUCTION

The increasing demand of portable applications and computation-intensive applications pushes the modern VLSI chip's size and speed to its extreme. The rapid evolution of integrated circuit technologies allows people to put more and more devices on a single chip and run it at higher and higher speed. As a result, the chip gets hotter and hotter. The heat caused by excessive power dissipation severely limits further improvement of the chip performance. Also in portable applications, excessive power dissipation is a critical threat to the battery life.

Enormous amount of efforts have been spent on reducing the power consumption of existing circuits and developing new low-power VLSI design techniques. A common approach is to trade speed for low power by a choice of supply and threshold voltage [Liu93][Cha92]. By scaling down both the device size and power-supply voltage, one may achieve lower power consumption with minimal reduction of overall operation speed. Power saving can also be achieved through system-level concepts and logic-level concepts. Automatically turning off inactive modules in a system [Ben94] and re-encoding sequential circuits to reduce the average number of bit changes per state transition [Hac94], are examples of system- and logic-level concepts respectively.

In this thesis, power saving techniques for sequential machines are investigated. A special enabled flip-flop is introduced. The double-level-sensitive (DLS) and the double-edge-triggered

(DET) design methods are extended. These are based on the reduction of the average number of transitions per state in a Finite-State Machine (FSM) and the reduction of average internal activity when an FSM is not changing states. Before the introduction of these new methodologies, some conventional logic-level power reducing methods are discussed in Chapter 2 as background.

In Chapter 3, the proposed enabled flip-flop method is discussed. It is developed based on the clock-gating method but without the potential for hazard or false-clock-edge generation. The focus of this chapter, as well as this thesis, is on static type logic rather than dynamic logic configurations. There are a few reasons for doing so:

- (1) Static logic is the more common logic design.
- (2) It allows gating the clock to a whole unit for even more power saving.
- (3) It does not have a minimum clock speed requirement.

By proper design, an enabled flip-flop can use reduced power when disabled without gating the clock. It can be utilized on individual flip-flops and single clock cycles. In conventional design enabling usually does not save power. The extended enabled flip-flop is a novel power saving circuit which can be applied to low-activity FSMs without any extra efforts. Secondly, the analysis of making a simple externally-synchronous FSM, the enabled flip-flop, provides the readers with the insight on designing externally-synchronous FSMs, which serves as the introduction to the next chapter.

In Chapter 4, the DLS methodology is proposed. With the knowledge gained in the previous chapter, the FSMs are designed without using flip-flops. The resulting FSM itself can be treated as a “giant” DLS multiple outputs flip-flop (mop-flop). As the name implies, it triggers at both levels of the clock and has multiple independent outputs. With proper design, the resulting FSM has a reduced device count and power consumption over the corresponding conventional FSM. A few examples are shown at the end of Chapter 4. A saving/improvement of up to 50%-

60% on size, speed, and power can be obtained on these circuits.

It is worth pointing out that the enabled flip-flop design method and the DLS mop-flop design method described in Chapter 3 and Chapter 4 can be done using standard synthesizer tools, such as Synopsis, except for the hazard check mentioned in section 4.0.1. Also, the mop-flops constructed in Chapter 4 can be used as standard cells and bigger circuits can be built with them. Some examples can be found at the end of Chapter 4.

In Chapter 5, the proposed DLS design methodology is combined with the proposed enabled flip-flops to form one special type of large scale FSMs, counters. The resulting counter circuit triggers on both edges of the clock and shows both simplicity and power efficiency advantages over existing counter designs, the most efficient of which are found in U.S. patents. [refer to Chapter 5]

Finally, this thesis is summarized in Chapter 6.

All the devices used in this thesis are CMOS devices. The power consumption and speed simulations are carried out by HSpice, using the Canadian Microelectronics Corporation's (CMC) 0.8 micron BiCMOS HSPICE library Version 1.1. The HSpice files of the circuits and modules used in this thesis can be found in Appendix A.

The power-supply voltage used in this thesis is 5 volts. This seems high as today's industry standard is 3.3 volts. However, the focus of this thesis is to propose a new algorithm rather than circuit on-chip realization. The proposed algorithms should offer the same percentage power saving on 5V or 3.3V.

The devices used were 0.8/3 micron NMOS and 0.8/6 micron PMOS running at 10 MHz. The percentage power saving from using these design methods should be orthogonal to speed or feature size.

CHAPTER 2

BACKGROUND

2.0 Introduction

Power saving methods for a VLSI chip can be divided into different groups based on their focus on different level concepts:

1. Device-level concepts: such as lower V_{DD} .
2. System-level concepts: such as clock-gating.
3. Logic-level concepts: such as optimum state encoding.

For CMOS, the device-level power savings are usually based on the formula:

$$Power = CV^2f \quad (2.1)$$

This shows that the amount of power dissipated in a CMOS circuitry is proportional to the circuitry's equivalent parasitic capacitance, effective operating frequency and the square of the supply voltage. Several power saving methods based on the physics of the CMOS devices are reviewed here. These methods focus on reducing the supply voltage and parasitic capacitance of CMOS.

The system-level concepts focus on reducing circuitry's operation frequency. They also save power by turning off the inactive modules of a system. An interesting implementation methodology is discussed in this chapter.

The logic-level concepts achieve power saving by optimizing the circuit structure, either by reducing the number of devices in a circuit and/or by reducing the circuit activity for performing the same function. They focus on reducing both circuitry's equivalent capacitance load and the effective operating frequency. The logic-level abstraction is the approach mostly used in this thesis and is the focus of most of this chapter.

2.1 The Basis Of CMOS

If there is one single characteristic that can alone justify the existence of CMOS devices, it is their low power dissipation at low frequency. Unfortunately, as the speed of the circuit increases, the power consumption in CMOS grows linearly.

As the acronym implies, every CMOS circuit consists of two complementary parts acting as pull-up and pull-down circuitry, and made of p-MOS and n-MOS respectively. For any stable inputs, either the pull-up circuitry or the pull-down circuitry will be conductive and connect either V_{DD} or V_{SS} to the output. They never conduct at the same time, except during the fast transition time. The basic CMOS model for dynamic power dissipation is shown in Figure 2.1:

Figure 2.1 Basic CMOS model for dynamic power dissipation

In Figure 2.1, C_{in} is the equivalent input capacitance which includes the input line capacitance, n-MOS and p-MOS gate capacitance. C_{out} is the equivalent output capacitance which includes the output line capacitance, n-MOS and p-MOS sources to drains capacitance, and the output load capacitance. There is also capacitance between the gates and drains of p-MOS and n-MOS devices.

The power dissipated on CMOS can be roughly divided into two parts: the dynamic power consumption and the leakage power consumption.

2.1.1 Dynamic Power Consumption

The dynamic power consumption of CMOS contains two parts: the output (load) capacitance transient dissipation and current spiking dissipation during switching.

2.1.1.1 Load Capacitance Transient Dissipation

As shown in Figure 2.1, whenever the CMOS circuit switches, the output is either charging the output capacitance up to V_{DD} or discharging it down to V_{SS} . For the low-to-high output transition, the amount of power E_s transferred from V_{DD} through p-MOS pull-up device to C_{out} can be calculated as:

$$E_s = \int_{t_0}^{t_1} V i_c(t) dt \quad (2.2)$$

Where $V = V_{DD} - V_{SS}$ and $i_c(t)$ is given by

$$i_c(t) = C_{out} \frac{d}{dt} v_c(t) \quad (2.3)$$

Substituting equation (2.3) into equation (2.2) and assuming the initial voltage across the

capacitor is zero and the capacitor is fully charged at the end of the charging cycle, E_s can be calculated as:

$$E_s = C_{out} V \int_{t_0}^{t_1} \frac{d}{dt} v_c(t) dt = C_{out} V \int_0^V dv_c = C_{out} V^2 \quad (2.4)$$

Half of the E_s drawn from the power source is stored in the C_{out} because $E_{stored} = \frac{1}{2} C_{out} V^2$. The other half is dissipated as heat.

When the high-to-low output transition occurs, the power E_c stored in the C_{out} will be fully discharged into the ground. Given the frequency f of output transitions, the power dissipated is as shown in equation (2.1). From this equation one can see, at a data frequency which allows C to be fully charged and discharged, the power consumption of the CMOS device is directly proportional to its output switching frequency.

2.1.1.2 Current Spiking During Switching

The power is proportional to switching frequency only if the rise/fall time is short. Otherwise, the current spiking effect has to be taken into consideration for power consumption.

Current spiking refers to the phenomenon that when a CMOS gate is undergoing logic level switching, both n-devices and p-devices are turned partially on for a short time, when the input voltage is around $\frac{1}{2}(V_{DD} - V_{SS})$. The resulting low impedance path from V_{DD} to V_{SS} causes a short surge of current flow. This current is a function of the supply voltage ($V_{DD} - V_{SS}$).

The faster the circuit switches between levels the smaller the current spiking power used. However the speed of signal change is restricted by the RC time constant of the circuit. Since fast signal transition will cause problems such as ground bounce, very fast signal transitions are also

not desired.

The current spiking power of a CMOS inverter can be estimated by the formula given in [John97]:

$$Power = \frac{1}{12} \beta f t_{rf} (V_{DD} - 2|V_{tn}|)^3 \quad (2.5)^1$$

Since the current spiking power dissipation is proportional to V_{DD}^3 , it is much less important at low power-supply voltages. As long as the t_{rf} is not excessively long, this power is of second order importance for most CMOS circuits.

2.1.2 Leakage Power Consumption²

The leakage power consumption is very small and often ignored in static logic. MOS transistors are often modeled as switches. When the switch is off, there is no current flow between its drain and source. This is not completely accurate. There is sub-threshold leakage current present in MOS transistors which can be estimated by [John97]:

$$I_{DS} = I_0 \exp\left(q \frac{V_{GS}}{nkT} - 1\right) \quad (2.6)$$

Where I_0 , q , k and T are constants and constant n is normally between 1 and 2. This leakage current is important for low power when the circuit runs from batteries and is idle for long time. One example in [John97] points out that for a typical 0.5 micron process has a subthreshold leakage current of less than 5pA per micron, the total subthreshold current, of a 10 million transis-

-
1. Where $\beta = \left(\frac{W}{L}\right) \mu C_{ox}$, and $|V_{tn}|$ is the magnitude of the threshold voltage and is assumed the same for both p- and n-channel transistors. The t_{rf} is the rise and fall time of the input signal.
 2. Section 2.1.1.3 and 2.1.2 are included for the purpose of completeness of CMOS power analysis. The current spiking and leakage power are ignored in the future analysis.

tors chip and with each transistor 10 micron wide, can be 0.1mA.

It also becomes more important as supply voltage decreases. For low supply voltage, the threshold voltage must be reduced to maintain reasonable switching speed, but dropping the threshold voltage will increase the leakage current. One solution is mentioned in section 2.2.3.

2.2 Power Saving from Device-Level Concepts

From the previous discussion of where the power is dissipated in a CMOS device, one can summarize the power saving methods for the physical(device)-level concepts. Such methods include:

- Run the circuit at minimum supply voltage
- Reduce capacitance load
- Increase threshold voltage

2.2.1 Run The Circuit At Minimum Supply Voltage

From the power consumption equations in section 2.1.1.1 one can see, the power dissipation of a CMOS device is proportional to the square of the supply voltage. If one can lower the supply voltage by half, the power dissipation will be cut by three quarters. A further power saving is possible due to less current-spiking power as the result of lower supply voltage. Power reduction is indeed one of the major reasons that push the industry to replace the standard 5V power-supply with the new standard of 3.3V and 2.5V power-supply. Research on using 1.8V and 1.1V power-supply is currently ongoing.

The drawback of reducing the power-supply voltage is reduced speed. In order to compensate for the speed reduction, one has to increase the computational parallelism to maintain the same computing throughput. Since the speed reduction is not quadratic, doubling the logic to increase speed is reasonable if the power is reduced by 3/4 as a result of supply voltage being halved. This results in more complicated algorithms and more control overhead. As more devices

are required to perform the same function, power dissipation tends to rise again.

If the process can be changed and the gate oxide thickness can be reduced, lowering voltage can increase speed. More will be said in section 2.2.2.1.

2.2.2 Reduce Capacitance Load

This can be done by scaling down the devices and/or rearranging the signals.

2.2.2.1 Reduce Capacitance Load By Scaling Down The Devices

Reducing supply voltage reduces power but also reduces speed. However, reduced supply voltage allows one to do scaling. If one scales without also scaling t_{ox} ¹, the g_m ² of the transistor goes down. This reduces speed. In order to scale t_{ox} , one must reduce supply voltage. Otherwise, the reduced thickness of t_{ox} will increase the stress on gate oxide and cause oxide to breakdown.

The scaled smaller devices have shorter gate length which allows carriers to pass through it faster. Smaller devices also have less parasitic capacitance associated with them, therefore the capacitive load of a smaller device is also smaller. This not only makes smaller devices faster (less RC time), but also reduces their power consumption.

Assume the scaling factor is α , the speed of the original process can be estimated by:

$$\Delta t \approx \frac{4}{3} \frac{L^2}{\mu_n (V_{DD} - V_{th})} \quad (2.7)$$

Since both L and V_{DD} are scaled down by $\alpha < 1$, the speed of the scaled process is:

-
1. Gate oxide thickness.
 2. Gain of the transistor.

$$\Delta t_{new} \approx \frac{4}{3} \frac{\alpha^2 L^2}{\mu_n \alpha (V_{DD} - V_{th})} = \alpha \Delta t \quad (2.8)$$

Overall, if one can control the process, reducing the supply voltage is necessary to increase speed by the means of scaling. Detailed analysis on the effects of scaling down the device can be found in [Sne96].

2.2.2.2 Reduce Capacitance Load By Rearranging The Order Of Input Signals

Sometime one can reduce the effective capacitive load by simply rearranging the order of input signals fed into a CMOS gate. As shown in Figure 2.2, there are two signals entering the NAND gate, A and B. Assume B is the “mostly on signal”, i.e. gate 2 is off and gate 4 is on for most of the time. Input A is switching and causing gate 1 and 3 to turn on and off.

If one feeds the switching signal A into gate 3. As shown in Figure 2.2 (a), both capacitors are discharged through n-MOS gates 3 and 4 to the ground. Then input A goes to “0” as shown in Figure 2.2 (b). Since the internal capacitance $C_{internal}$ is now blocked from the V_{DD} by gate 3, only C_{load} is charged. Next time the input A goes “1”, only one capacitor is being discharged. If one switches the order of inputs A and B, both capacitors will be charged and discharged.

Figure 2.2 Better put the mostly on gate at the bottom

Experiments have been done on the circuits used in this thesis, by changing the order of the signals, one may get some extra power saving. Since the amount of power saved is insignificant comparing with the power saved by using the proposed design methods, the effect of switching signals order is not considered in the later chapters.

2.2.3 Increase Threshold Voltage

By scaling down the device size and lowering its power-supply voltage as in the last two sections, not only will the dynamic power consumption of the CMOS devices be reduced, its leakage power consumption will also be reduced.

Further reducing of subthreshold power consumption can be achieved by increasing the threshold voltage of a MOS device. This can be done by using the “backgating” method. For example, when the substrate of a NMOS is connected to a voltage which is about two volts below

the V_{SS} , the back gate (substrate) will try to turn the NMOS gate “off”. As a result, the subthreshold leakage current will be reduced exponentially. A charge pump as shown in Figure 2.3 can be used to generate the required voltage.

Since the charge pump circuit consumes power itself, from power saving point of view, this method is only useful when one such pump can save more power from other devices’ subthreshold leakage power than it consumes.

Figure 2.3 A charge pump to bias the back gate (a) pump up (b) pump down

2.3 Power Saving from System-Level Concepts

The power saving methods discussed in last section focus on reducing the power-supply voltage and the equivalent parasitic capacitance of a CMOS circuitry. There is another factor in the power dissipation equation (2.1). It is the effective operating frequency f of a circuitry.

A system designed to operate at a maximum frequency f does not mean that every part of the circuitry always runs that fast. For example, suppose there is an FSM inside that system oper-

ating at a clock of frequency f . When this FSM is waiting for a certain event to happen, thousands of clock cycles may pass. In this case, the effective operating frequency of the FSM module is 3 orders smaller than f . Power can be saved by turning off the FSM clock during that waiting period.

The reason to turn off the clock signal is that, even when the FSM is inactive, the clock signal alone will cause some internal gate capacitors of the FSM's "state-holding elements" (SHE) to charge or discharge. By turning off the clock signal fed into the inactive FSM, some power dissipated on the clock lines can be saved.

2.3.1 Clock-Gating Logic

The control signal, used to block the clock signal from being fed into the FSM, has to be generated by circuitry too. This can be seen in Figure 2.4. The control logic takes the inputs to the FSM and the current state of the FSM to decide when to pass the system clock to the FSM. By doing so, the activity of the FSM is reduced to its minimum, i.e., the effective f is minimized.

Figure 2.4 System-level clock-gating structure

Generating the minimum effective f may force the control logic to be quite complex. It not only takes more devices to build it, but also may consume more power than it can save. A compro-

mise has to be made between the complexity of the control logic and its coverage. There have been some algorithms published which can optimize the overall power consumption. One of them can be found in [Ben94].

In this thesis, two clock-gating approaches are used. The first one is to use an enabled flip-flop which gates the clock on each flip-flop, rather than all the flip-flops together. More details are discussed in Chapter 3. The second one goes one step further to manipulate the clock inside the flip-flops. More details are discussed in Chapter 4.

2.4 Power Saving From Logic-Level Concepts

Three approaches are discussed here:

- Optimum state encoding
- Optimizing the flip-flops
- Using latches

2.4.1 Power Saving By Optimum State Encoding

An FSM consists of a combinational circuit and several SHEs¹, say, flip-flops. The flip-flops memorize the current state of the FSM. When the FSM moves to next state, new values will be stored in those flip-flops. As shown in Figure 2.5, when the stored values change, their feedback as inputs to the combinational logic will cause gates switching. Therefore, the more state variables change per state transition, the higher the effective frequency f of the flip-flops and the gates will be, and the more power will be consumed.

1. The output generation circuit is ignored in this thesis for simplicity consideration. One may also treat the output generation circuit as part of the combinational logic of the next block of FSM.

Figure 2.5 Basic FSM structure

However, this is at best a general principle. In fact reflected-Gray-code counters only change one bit at a time but require so many extra gates over a binary counter that the power saving from the single bit changes between states is completely overshadowed by the power in the additional gates. Also, it is not feasible to apply Gray code encoding to certain FSMs.

Some encoding algorithms for minimizing power dissipation in sequential circuits have been published, one can be found in [Hac94]. Gray encoding is also used in later chapters for building novel circuits.

2.4.2 Power Saving By Optimizing The Flip-flops

Reducing the power consumption of an FSM can also be achieved by selecting a power-efficient flip-flop. Only D flip-flops (DFF) are discussed in this section for simplicity.

Figure 2.6 shows two commonly used DFF configurations. The transmission gates DFF shown in Figure 2.6 (a) contains 4 transmission gates and 6 inverters¹ [Hir98], a total of 20 transistors. The 6-state DFF, with its state diagram in Appendix C, shown in Figure 2.6 (b) consists

1. The clock buffer 2 is used to reduce the direct clock loading from 6 transistors to 2. It is important when the number of flip-flops is large.

five 2-input NAND gates and one 3-input NAND gate, a total of 26 transistors.

Figure 2.6 DFFs (a) with transmission gates (b) 6-state DFF

Normally, the more devices in a circuit, the more power the device takes. This is not true here. Although the DFF shown in Figure 2.6 (a) contains 6 fewer transistors, it has the clock signal fed into 4 transmission gates directly and has two inverters which are always switching no matter whether the DFF is active or not.

In comparison, the DFF shown in Figure 2.6 (b) contains more devices but has the clock signal directly fed into only 2 gates. It is “quieter” especially when the DFF is active. Therefore, the 6-state DFF saves power both when it is active and inactive. An HSpice simulation was done

to confirms this, as shown in Table 1.

Table 1: Power comparison of the two DFFs shown in Figure 2.6

	transmission gate DFF	6-state DFF	power saving of 6-state DFF
size	20 transistors	26 transistors	N/A
idle power	0.30 mW	0.24 mW	20%
toggle power	0.58 mW	0.46 mW	21%

In Table 1, the idle power means the power dissipated in the flip-flop when the input stays constant; the toggle power means the power dissipated in the flip-flop when the flip-flop changes its output every clock cycle. It is clear that by choosing a good flip-flop configuration, one can save power.

As mentioned at the end of Chapter 1, the size of the transistors and the speed of circuit are orthogonal to the percentage amount of power saved.

So far, the 6-state DFF is the most power-efficient static DFF among the well-known configurations. Further improvement is attempted in the next chapter.

2.4.3 Power Saving By Using Latches

The latches can also be used in an FSM. The structure of a D latch is simpler than a DFF. A standard D latch is shown in Figure 2.7.

From Figure 2.7 one can see, the D latch can hold one bit of a state variable just like the DFF while its size is only about half of a DFF. Indeed, the master-slave DFF is built by cascading

two such D latches, each with opposite clock input.

Figure 2.7 Structure of D latch and its symbol

The power consumption in an FSM built with latches will be always better than, or at worst equal to, the FSM built with master-slave DFFs. This will now be proven. The general structure of a latch based FSM is shown in Figure 2.8.

Figure 2.8 General structure of latch based FSM

Assume there are Y states in an FSM. In a DFF based structure, one needs y flip-flops, where:

$$y = \lceil \log_2 Y \rceil, \quad \lceil \cdot \rceil \text{ is the ceiling} \quad (2.9)$$

and a combinational circuit which can handle both the inputs and the y feedbacks. In a latch based structure, one can divide the Y states into two groups of states. A group of M states which latch on phase one ($\Phi 1$) and a group of N states which latch on phase two ($\Phi 2$).

At any given time, the machine must remember m bits during $\Phi 1$ and n bits during $\Phi 2$.

$$\text{where} \quad m = \lceil \log_2 M \rceil \quad (2.10)$$

$$n = \lceil \log_2 N \rceil \quad (2.11)$$

Thus one uses $(m+n)$ latches.

$$Y \geq \text{MAX}(M, N) \quad (2.12) \quad \text{which implies}$$

$$y \geq \text{MAX}(m, n) \quad (2.13)$$

Therefore,

$$m + n \leq 2y \quad (2.14)$$

But $2y$ is the number of latches contained in y master-slave DFFs. Therefore, devices may be saved. Furthermore, since the inputs and $(m+n)$ feedbacks are partitioned into two combinational logic circuits as shown in Figure 2.8, it turns out that the total number of devices in two partitioned combinational logic circuits may be less than the number of devices found in a non-partitioned combinational circuit which has to handle all inputs and Y feedbacks together.

A more straightforward worst-case-proof can be done by taking the original circuit with y DFFs, making each DFF master-slave and breaking it into two latches. Then the two combinational circuits in Fig 2.8 reduces to wires and one has a latches based circuit with:

$$M = N = Y \quad (2.15)$$

$$m = n = y \quad (2.16)$$

2.4.3.1 Example of Latch Based Circuit

One example is shown here to illustrate the potential advantage of using latches to build FSM. The function of this FSM is to locate serial words which are valid excess-3 code and can be described by the following state diagram.¹

Figure 2.9 State diagram of the excess-3 code FSM

By putting the $\Phi 1$ states and $\Phi 2$ states into separate state tables and separate K-maps, a set of equations can be derived. Refer to Figure 2.8: the input of this circuit is X, which is one bit of the 4-bit excess-3 code. The equations derived from $\Phi 1$ states form the combinational circuit 1, its outputs, p and q, are fed into $\Phi 1$ latches. The equations derived from $\Phi 2$ states form the combinational circuit 2, its outputs, a, b and e, are fed into $\Phi 2$ latches. The final latch based FSM is shown in Figure 2.10.

1. Excess-3 code is calculated as BCD-code + 3, valid excess-3 codes are 4-bit code 01XX, 10XX, 0011 and 1100, where XX stands for any "0" and "1" combinations. The advantages of excess-3 codes is that a carry out is automatically generated for sums larger than 10 and inverting each bit gives the compliment.

Figure 2.10 Latch based excess-3 code FSM

The comparison between this latch based circuit and the best known conventional edge-triggered master-slave DFF based circuit is listed in Table 2.

Table 2: Comparison of latch based and DFF based^a circuits

	2-input gates	3-input gates	latches	gate inputs	literals
DFF circuit	7	6	8	35	28
Latch circuit	6	3	5	20	16

a. An alternate circuit which stores the x input in a shift register can be made. With a Mealy output it would use 6 latches.

The advantage of using latches in this FSM is apparent.

2.5 Different Architectures Of FSMs

Two conventional ways of making FSMs have been discussed.

1) Single phased DFF based circuits

2) Latch based circuits

2.5.1 The Proposed FSM Architectures

Two architectures will be presented in this thesis:

1. Double-level-sensitive (DLS) Mop-flop Architecture. (Chapter 4)
2. DLS FSM with flip-flop Architecture. (Chapter 5)

2.5.1.1 The DLS Mop-Flop Architecture

In this architecture, the storage elements in an FSM are expanded into a DLS mop-flop like structure.

Each mop-flop, like a flip-flop, is internally an asynchronous machine. However, externally its outputs are synchronous. That is by making external inputs obey setup and hold time restrictions with respect to the clock. The outputs can be guaranteed to be spaced a minimum distance from a clock edge.

All of the DLS mop-flop developed for this thesis could be partitioned into latch based circuits. The clock to these latches might be internally gated.

These machines are fully synchronous and trigger on both levels of the clock.

The main advantages are:

1. Saving power by using both clock levels/edges.

2. Incorporate clock-gating within standard cells. It will be shown that this makes clock skew much more controllable and it avoids false clock edges. Both of these are serious problems with clock-gating of conventional flip-flops.

This architecture, namely DLS mop-flop architecture, is outlined in Figure 2.11 (a). It allows the designer to manipulate the clock signals, and/or further reduce the circuit size, to achieve power saving. It can also be easily synthesized by existing CAD tools. More details can be found in Chapter 4.

2.5.1.2 The DLS FSM With Flip-Flop Architecture

A variation of the mop-flop architecture can be achieved by replacing the latches with toggle flip-flops. The resulting FSM has a structure similar to the conventional flip-flop based FSM and triggers on both edges of the clock. This architecture can be also called the double-edge-triggered (DET) architecture. It is outlined in Figure 2.11 (b) and more details can be found in Chapter 5.

Figure 2.11 The proposed FSM architectures (a) Typical form of a DLS mop-flop (b) Typical form of a DET FSM

CHAPTER 3

MAKING BETTER FLIP-FLOPS

3.0 Introduction

From the discussion of the D flip-flops (DFFs) in Section 2.4.2 one can see that DFFs take a considerable amount of power even when their outputs are not switching. It would be useful to have an enable signal attached to the flip-flops to tell them to “turn off” when they are not doing useful work. Other than the power saving, the use of an enable signal can also make the design of an FSM more flexible. When the enable signal is turned off, the flip-flop’s output will be irrelevant to the input; when the enable signal is turned on, the flip-flop will just behave as a normal flip-flop. By doing so, the complexity of the combinational circuit of an FSM will at least appear to be reduced.

The conventional way of making a DFF with enable signal is to simply multiplex the D input with previous Q output of the flip-flop, using the enable as the selection signal, as shown in Figure 3.1.

Figure 3.1 DFF with enable

The amount of power consumed in above circuitry mostly comes from the power dissipated in the DFF. The MUX consumes much less power than the DFF and dissipates virtually no power when its input signals do not change. Unfortunately, as long as the clock goes, the DFF takes power even when the MUX output stays the same.

In this chapter, a power dissipation analysis will first be performed on the conventional DFF. It will expose the reason why the flip-flop takes considerable power even when its output is not changing. Then, two better flip-flops configurations, the clock-gating enabled flip-flop (gated flip-flop) and a new intrinsic enabled flip-flop, which can reduce such power wasting will be considered.

Of the two better flip-flops, the gated flip-flop has a smaller circuit size and saves more power. Unfortunately, since it is made by manually inserting an extra clock-blocking gate into an existing flip-flop, it may suffer from the false clocking edge problem. The intrinsic enabled flip-flop method eliminates the false clocking edge problem by integrating the enable signal into the design of the flip-flop. Compared with the conventional enabled DFF, shown in Figure 3.1 it has a smaller size and can save considerable power when the flip-flop is not too active. HSpice simulations are performed on all three circuits. Several rules to help decide which type to choose, are given at the end of the chapter.

Finally, note that understanding the flip-flop design method in this chapter is quite impor-

tant. It serves as a bridge to the next chapter where the multiple outputs flip-flops (mop-flop), i.e., FSMs made with feedback loops as “State-Holding Elements” (SHE), are developed.

3.1 The Conventional DFF

There are several configurations of static DFF. To analyse their performance and find ways of making improvements, one has to treat them not as an elementary gate, but as an asynchronous FSM¹.

3.1.1 The DFF Is An FSM

The flip-flop is one of the most commonly used SHE in a *synchronous circuit*. It is a circuit with at least 2 feedback loops to hold the current state. For a commonly used DFF, it has at least two inputs, D and clock, and one output Q. Ideally, its function is to update the output value Q to equal the input value D on the clock rising edge² and the rest of the time keep Q as it is no matter how D changes.

The DFF is an FSM. It has combinational circuitry and SHEs, the feedback loops, inside. By treating the flip-flop as an FSM, the clock signal is not a special signal any more. This is illustrated by Figure 3.2, redrawn from Figure 2.6 (b).

-
1. It is an asynchronous FSM behaves like a externally-synchronous FSM.
 2. A rising clock edge-triggered flip-flop is assumed.

Figure 3.2 The 6-state DFF

3.1.2 The Clock Induced Power Dissipation In DFF

Every DFF, constructed as in Figure 3.2, contains 6 gates. For an FSM with three such DFFs, the DFFs alone contain 18 gates. This is often bigger than the size of the combinational circuit in the same FSM. When the FSM is changing states, both the gates of the combinational circuit and the flip-flops are switching and consuming power. When the FSM is not changing the states, the gates of combinational circuit consume power only from input changes. However, some of the gates of the DFFs follow the clock signal even though the D inputs are constant. This can be seen from Figure 3.3:

Figure 3.3 Internal nodes switching of a DFF ($D=Q=1$) under clock rising edge

In Figure 3.3, assume the $D = "1"$ and $Q = "1"$,

=> output of gate G1, G2 and G3 = "1", output of G6 = "0";

As the clock changes "0" -> "1",

=> output of gate G2 goes "1" -> "0";

Therefore, there is power dissipated in gate G2's output capacitance and gate G1, G2, G3 and G5's input capacitors are charged/discharged.

After all internal nodes reach stable states, the clock goes "1" -> "0". This causes output of G2 to go "0" -> "1" and the same number of capacitors are charged/discharged again.

In Figure 3.4, assume the $D = "0"$ and $Q = "0"$,

=> output of gate G2, G3 and G6 = "1", output of G1 = "0";

As the clock changes "0" -> "1",

=> output of gate G3 goes "1" -> "0";

Since G3 is a three-input gate, its output capacitance is also bigger. As the bigger capacitance is being charged and discharged, it takes just as much power (about 6 capacitors charging/discharging) as in Figure 3.3.

Figure 3.4 Internal nodes switching of a DFF ($D=Q=0$) under clock rising edge

The amount of power consumed in Figure 3.3 and Figure 3.4 is significant, especially for the FSMs which do not change every state variable on every clock edges. For example, an 8-bit binary counter consists of 8 DFFs. When the counter is not counting, for every clock rising edge¹, there are 8 gates (G2 or G3) inside the DFFs switching twice and 6×8 capacitors being charged and discharged. Even when the counter is counting at every clock rising edge, on average, only 2 flip-flops are really changing their outputs². The rest of the flip-flops just flip the gates and charge/discharge capacitors for nothing. It is desirable to find a way to reduce the power wasted this way.

1. Rising clock edge-triggered DFFs are assumed.

2. The probable of number of flip-flops switching is: $1 + 1/2 + 1/4 + 1/8 + \dots$ approaches 2

3.1.3 The Clock Induced Power Dissipation In The DFF With Enable

From the analysis of clock induced power dissipation in the DFFs one can see that the enable signal in Figure 3.1 can only stop the input D, but not the clock signal, from being fed into the DFF. It can not stop the power waste discussed in last section. One must find other ways to make improvements.

3.2 The Clock-Gating Enabled Flip-Flop Method

Instead of enabling a DFF by selecting which signal to be used as the D input, one can also enable a DFF by controlling the clock input. This method is called clock-gating and can be seen in Figure 3.5.

Figure 3.5 A gated flip-flop with (a) an AND gate (b) an OR gate. The shaded latch can be added to either circuit to remove false clock edges. Here it is only shown added to (b). The extra latch adds enough to the power dissipation so there is no saving unless one “en” signal can control several flip-flops.

In Figure 3.5 (a), when the enable signal is at logic “0”, the output of the gate G will be held at logic “0”. There will be no gate flipping and thus no capacitor charging/discharging inside the DFF. By gating the flip-flop, there is only one capacitor at the input of the gate G to be charged and discharged when the enable signal goes low. Less power will be wasted when the flip-flop is supposed to be inactive. The AND gate used in Figure 3.5 (a) can also be replaced by an OR gate as shown in Figure 3.5 (b), the circuitry will work the same way except now the clock is clamped at logic “1” when the gated flip-flop is disabled.

Compared with the circuitry shown in Figure 3.1, the gated flip-flop method saves dynamic power when it is not enabled. A power dissipation comparison can be estimated using Table 3.

Table 3: Power dissipation estimation for circuits in Figure 3.1 and Figure 3.5(a)

Circuit	Figure 3.1	Figure 3.5(a)
Enabled	6 capacitors	8 capacitors
Disabled	6 capacitors	1 capacitor

Calculation¹ shows that for flip-flop activity less than 71%, the gated flip-flop shown in Figure 3.5(a) saves power over the DFF shown in Figure 3.1.

However, there is a limitation associates with the gated flip-flop method. Since the clock signal is gated, false clock rising/falling edges can be generated by the gate G and fed into the DFF. One such problem is shown in Figure 3.6:

Figure 3.6 Timing diagram for the gated DFF with an AND gate

To eliminate this problem, one has to make sure that the enable signal will never rise/fall

1. From: $8 \text{ capacitors} * ON + 1 \text{ capacitor} * (1-ON) = 6 \text{ capacitors} * ON + 6 \text{ capacitors} * (1-ON)$
 $\Rightarrow ON=71\%$. Refer to section 3.4 for details.

when clock signal is high/low for the circuitry configuration of Figure 3.5 (a)/(b), respectively. The enable signal must be restricted to only change in the first/second half clock cycle for OR/AND gating.

To do so, a D latch may be added to restrict the changing of the enable signal to half of the clock cycle, as shown by the shaded latch in Figure 3.5 (b). However, the added latch will make the circuit 50% larger and cause 50% more transitions¹. This is not desirable. A dynamic latch may be used which will reduce the device count and power dissipation. However, even with the use of dynamic logic the power dissipation of this gated flip-flop is still larger than the one that only uses one AND/OR gate. It is only feasible when one such gate is used to control several flip-flops.

3.3 The Intrinsic Enabled Flip-Flop Method

Instead of making an enabled flip-flop by adding an extra gate to the DFF, as was done in the gated flip-flop, one can also design an enabled DFF with the enable signal built in. The resulting flip-flop will have a smaller size and waste less power, when it is not doing useful work, than the circuit of Figure 3.1. Also the limitation on the enable signal of Figure 3.6 will be removed.

3.3.1 The Intrinsic Enabled DFF

A state diagram of the enabled flip-flop can be expressed by the FSM state transition diagram as shown in Figure 3.7.

1. Master-slave type DFF is assumed.

Figure 3.7 State diagram of the enabled DFF

3.3.2 The Intrinsic Enabled T Flip-Flop (ETFF)

If one removes all control from the D input and enables the flip-flop only when the output is going to toggle, one gets a toggle or enabled-toggle flip-flop. Therefore, to perform the same function as the enabled DFF shown in Figure 3.7, one only needs the enable signal and the current state to decide the next state. The knowledge of the input D is redundant. A simpler state transition diagram can be drawn as shown in Figure 3.8.

Figure 3.8 State diagram of ETFF

3.3.2.1 The Schematic Of The Intrinsic ETFF

The circuit derived from the state diagram of Figure 3.8 is shown in Figure 3.9. As in the Figure 3.7, the “e” stands for the enable signal, the “c” stands for the clock signal and the output of the flip-flop is q1. By using the complex gates to replace the gates G1 to G8, the circuit can be further simplified. The final circuit size is 36 transistors, which is slightly (4 transistors) smaller than the conventional enabled DFF circuit shown in Figure 3.1.

Before this ETFF can be used in FSMs, it has to be checked to be synchronous and race and hazards free.

Figure 3.9 Circuit schematic of ETFF

3.3.2.1.1 Synchronous vs. Asynchronous

In a conventional circuit, if the flip-flops or latches are used to store the state variables, this circuit is called a *synchronous circuit*. The ETFF is a circuit which does not contain any flip-flops nor latches. In this section, the meaning of synchronous and asynchronous is discussed and the ETFF is proven to be an *externally-synchronous circuit*.

DEFINITIONS

Normally a *synchronous circuit* has the following properties:

- 1) Has global clock.
- 2) The clock controls all switches in the storage elements. No other variables can initiate changes in what is stored.
 - (a) Change their stored values and outputs¹ only in response to the clock.

1. For Mealy machines, output may change in response to a non-clock but synchronous input signal, so the output transition indirectly responds to a clock change and is a *synchronous transition*, refer to the rule 3 in section 4.3.1.

(b) Have no internal timing problems caused by races or hazards.

3) Races are avoided by:

Making the delays between storage loops such that there is no interactions between storage elements during a state change. Setup and hold time specifications are such that during the state change no signals have time to travel between flip-flops hence no feedback loops can influence any others. In *asynchronous circuit* jargon, the clock causes races because many state variables change at a time. But the inter flip-flop delays make the final state independent of which flip-flop is faster.

4) All storage is in individual storage elements, normally latches or flip-flops.

In this thesis, when discussing flip-flops or flip-flop circuits, more definitions have to be used.

- (1) If state X of an FSM can only be reached after a clock edge or in direct response to a *synchronous transition*¹, this state is defined as a *synchronous state*.
- (2) If state Y of an FSM can also be reached after non-clock signal's edges or in direct response to an *asynchronous transition*, this state is defined as an *asynchronous state*.
- (3) To guarantee an FSM's output will change synchronously one can take its output only from synchronous state variables and/or inputs which are synchronous.²
- (4) If an FSM's output is only taken from *synchronous states*, this FSM is a *synchronous circuit*³; If an FSM's output is taken from *asynchronous states*, it is an *asynchronous circuit*; Or -

1. Refer to section 4.3.1.

2. This is for Mealy machine.

3. This definition is for Moore machine. All FSMs discussed in this thesis are restricted to Moore machines.

ETFF IS AN EXTERNALLY-SYNCHRONOUS CIRCUIT

The state diagram shown in Figure 3.8 is fundamentally different from a synchronous state diagram. In synchronous design, one can only change from one state to the next state right after the clock's active edge. The flip-flops inside the FSM will guarantee that all states are *synchronous states*. In Figure 3.8, there are no flip-flops inside. The state changes between state B, C and D, A will occur once all the conditions labelled on the arrows are satisfied. If for any transitions the clock condition is NOT the last to be satisfied, the machine will NOT change state on the clock edges, but on the edges of other non-clock input signals.

For example, if the current state is B and $c, e = "0", "0"$. The machine will sit on state B. Then input e changes $"0" \rightarrow "1"$ when clock signal is still low. As a result, the machine will immediately change from state B to C. This transition does not occur on clock rising edge. Thus the state C is an *asynchronous state*.

Fortunately, state C and A are not the place where the output is taken. The output of the flip-flop is taken from state B and D. The only condition for the machine to move to state B or D is the clock signal going high, thus state B and D are *synchronous states* and this ETFF circuit is an *externally-synchronous circuit*. It is a synchronous flip-flop¹.

From gate level implementation point of view, using the clock signal with other inputs as the conditions for state changes is equivalent to clock-gating. The result is the generation of the false clock rising edges. From behavioural point of view, this causes the machine to move into an *asynchronous state*. Designers using the flip-flops will not even be aware.

3.3.2.1.2 Races And Hazards In An Asynchronous Circuit

RACES

-
1. The ones-catching JK flip-flop is an example of an "asynchronous flip-flop".

According to the state diagram shown in Figure 3.8, this machine will always travel to the next state with only one state variable change. Therefore, it is race free.

HAZARDS

Gate G4, G5 and G6 form a three-gate D latch [TI85]. When the state machine is in state 00, there is a potential static-0 hazard in the Q output when C changes from “1” $\rightarrow$ “0”. Since the AND gate of G4 is composed of an NAND gate and an inverter, the data paths of C and complement C are both one NAND/NOR gate plus one inverter. Therefore, the delays are well balanced and the glitch never shows up in simulations.

The hazard inherents with the D latch is shown in Figure 3.10.

Figure 3.10 Static-1 hazard in the D latch

In a practical circuit, there is a delay mismatch due to circuit layout, if the mismatch is small, the fast glitch will be absorbed by the G6 gate. To make the design even safer from the potential hazard, one can try to put the extra delay into the data path of C to balance it with the path $\bar{C}$.

To completely cure this potential hazard one can replace the D latch with the transmission D latch as shown in Figure 2.6 (a). The transmission latch will not pull the output low when there is a delay mismatch in the control signal (clk). By doing so, the resulting circuit will be even small-

er. Since the clock signal can be turned off by the enable signal, the excess power dissipation problem associates with the transmission type DFF is also solved.

Most circuits displayed in later chapters also *asynchronous circuits* which must externally behave like *synchronous circuits*. They were all checked to be race and hazards free. Detailed procedure can be found in [Rab96].

3.3.2.2 The Performance Of The Intrinsic ETFF

The power and speed of the ETFF are analysed here.

3.3.2.2.1 The Power Dissipation

In Figure 3.9, when the ETFF is not enabled (i.e. $e = "0"$), both gate A and B will turn off. Compared with the circuit shown in Figure 3.3 and 3.4, the intrinsic ETFF has fewer capacitors being charged/discharged and one less gate flipping.

HSpice simulations were performed on the 6-state DFF of Figure 3.3/3.4 (6DFF)¹, the clock-gated flip-flop of Figure 3.5 (a) (GFF) and the ETFF of Figure 3.9. With a clock signal running at 10 MHz, the measured power dissipation is listed in Table 4.

Table 4: Power dissipation of the two proposed enabled flip-flops

(unit: mw)	6DFF ^a	ETFF	GFF
enable=0	0.24	0.10	0.05
enable=1	0.46	0.66	0.54

a. The value shown in this column is the average value of Figure 3.3 and 3.4

1. The conventional enabled DFF shown in Figure 3.1 takes more power than the 6-state DFF.

From Table 4 one can see, both the ETFF and the GFF have smaller power dissipation when they are not enabled. This feature is desirable when one wants to minimize the power spent on inactive flip-flops. The ETFF consumes slightly more power than the GFF when it is disabled, but it is free of the false clock edge restriction, thus the ETFF can be used more easily without restrictive timing considerations. If the flip-flops are active most of the time, there is no need to use the two proposed enabled flip-flops of the ETFF and the GFF, the conventional enabled DFF, as shown in Figure 3.1 should be used instead.

3.3.2.2 The Speed

The data-path delay of the 6-states DFF is about two gates. The data-path delay of the GFF is also two gates, but the output is delayed because of the delayed clock. The GFF circuit can run at the same speed of the DFF but its output will be one gate further delayed. The result of clock-gating is to reduce setup time and increase hold time and also increase t_{CHQV} (Clock High to Q Valid time). The maximum clock speed is unchanged. Clock-gating will not cause a problem if all the flip-flops are gated but will cause clock skew if some flip-flops are gated and some are not. The speed of the ETFF is about the same as 6-state DFF since there are also about two gates in the data-path, the gate B and the complex gate of G4, G5, and G6. Since there is no extra clock-gating gate, the output of the ETFF does not cause clock skew.

Compared with the MUX enabled DFF shown in Figure 3.1, which has 4 gates in the data-path, both the GFF and the ETFF are faster.

3.4 When To Use Which Flip-flop

Since both the ETFF and the GFF save power when they are disabled and consume more power when they are enabled than the 6-state DFF, there is an activation threshold¹ below which

1. The activity of the flip-flop.

one should consider using the ETFF and the GFF to save power. A simple calculation can determine the value of the threshold. Assume during normal circumstance, the flip-flop is active for $X\%$ of the time¹, then one has:

$$(P_{\text{Active ETFF}} - P_{\text{Active DFF}})X = (P_{\text{Idle ETFF}} - P_{\text{Idle DFF}})(1-X) \quad (3.1)$$

Substituting the values in Table 4 into equation 3.1 one has,

$$(0.66 - 0.46)X = (0.24 - 0.10)(1 - X) \quad (3.2)$$

Solve equation (3.2) for X and the result is X equals 41%. Which means, if the flip-flop is active in less than 41% of the time, power saving can be achieved by simply replacing the 6-state DFF by the ETFF. This value for the GFF is 68%. The less the activity of the flip-flop, the more the power saving can be achieved by using the ETFF and the GFF.

As long as the statistics of a flip-flop's activity is below the threshold, the ETFF and GFF will save power statistically and automatically. However, the global clock-gating may save more power if one can identify global criteria for shutting down a large number of flip-flops.

3.5 Example

The power saving from using the enabled flip-flops depends on the activity of the flip-flops of an FSM in which they are used. This activity is not the same as the activity of the FSM. Even if the FSM is active for every clock rising edge, only a small percentage of the flip-flops in the FSM are actually switching. For example, an 8-bit binary counter is active for every clock rising edge. But the flip-flop represents the most significant bit only changes every 128 clock cycles. Even counting at every clock cycle, the average activity of the eight flip-flops inside the counter is only 25%.

1. Active on X cycles out of 100 clock cycles.

From section 3.4, if the average activity of the flip-flops in an IC chip is 5%, a 0.12 mW^1 power saving per flip-flop can be achieved by using ETFF. Considering there are thousands of flip-flops, the total power saving by using the ETFFs instead of the DFFs will be significant.

3.6 Summary

The system level power saving method discussed at Chapter 2 tries to turn off inactive modules to save power. In this chapter, enabled flip-flops were described, which goes one step further by turning off not only the inactive flip-flops of an inactive circuit module, but also the inactive flip-flops in an active circuit module. In summary,

- Local clock-gating does not require shutting down the whole sub-system.
- ETFF does not add skew problems or false clock edge problems.
- GFF does add false clock edge problems and skew problems.
- The local gating methods do require a knowledge of flip-flop activity, which may be hard to estimate.

1. For DFF, power dissipation with 5% activity is: $0.05 \times 0.46 + 0.95 \times 0.24 = 0.251 \text{ mW}$;
For ETFF this value is: $0.05 \times 0.66 + 0.95 \times 0.10 = 0.128 \text{ mW}$.
The saving is $0.251 - 0.128 = 0.12 \text{ mW}$, or 48%.

CHAPTER 4

DOUBLE-LEVEL-SENSITIVE MOP-FLOPS

4.0 Introduction

The analysis of the flip-flops in Chapter 3 shows that the idling power consumption of a flip-flop is a function of the number of gates that the clock signal directly feeds. In *synchronous circuits*, the clock signal only feeds the flip-flops. By using the enabled flip-flops described in Chapter 3, one can reduce the number of gates directly affected by the clock signal, and achieve power saving when the flip-flop is idling. However, a more aggressive approach would be to reduce the number of flip-flops used in an FSM. By doing so, not only is the idling power consumption reduced, but also active power may be saved.

The number of conventional flip-flops required in an FSM is determined by the number of states of that FSM. For an N states FSM, one needs $\lceil \log_2 N \rceil$ conventional flip-flops; each holds one bit of the state variable. The number of conventional flip-flops in an FSM can not be arbitrarily reduced. However, this is only true for conventional flip-flops which have only one effective output per flip-flop. If one can find a way to merge several flip-flops in an FSM into one or more bigger “state-holding blocks,” i.e., special flip-flops with multiple outputs. One may reduce the number of gates used in “state-holding element” (SHE) and as a result, power saving may be

achieved.

Conventional flip-flops have two complementary outputs. However the second output is just the inverse of the first output, so it can not store additional information.

In this chapter, the possibility of creating some special flip-flops which have more than one independent outputs is investigated. The analysis of the nature of the SHE in an FSM, leads naturally into a novel variation of the double-level-sensitive (DLS) FSM design method. Using this method, one can effectively design an FSM using DLS multiple-output flip-flops, or “mop-flops” for short.

4.0.1 Brief Description Of Mop-flops

Mop-flops utilize several state-holding feedback loops. These loops interact with each other in a complex manner more like those in the 6-state DFF of Figure 3.2 than of the latch based FSM of Figure 2.8. The resulting circuits are often smaller and less power hungry than the conventional flip-flops they replaced.

These mop-flops have the following characteristics:

1. They have multiple independent outputs.
2. Output only change in response to a clock change.
3. Timing restrictions can be defined with respect to the clock in particular:
 - Setup-time
 - Hold-time
 - Clock-to-output time

This allows them to be assembled as *synchronous circuit* elements.

4. Internally they are *asynchronous circuits*, they must obey Huffman’s rules [Ung69]:

- No critical races

- No hazards
- No essential hazards

The above rules can be violated if certain delays can be guaranteed. Also, if delays are critical, the mop-flops must be made into standard cells with enough analog simulation of the design to ensure delay bounds.

Since there are often fewer gates that are directly affected by the clock signal, the mop-flops often have less power consumption when they are idling.

The enabled flip-flop method in Chapter 3 achieves power saving when the flip-flops are idling. The proposed DLS mop-flop also saves power when flip-flops are active by reducing the overall active gates in the circuits. They are easy to use in counter related circuits. Also, with increased design work, general controllers can also be made. The detailed analysis on why the proposed method saves switching power and some working examples are given in this chapter.

It is important to point out that the proposed DLS mop-flops can be synthesized by the existing commercial software, such as the SYNOPSIS, which makes the implementation of this novel technique much easier. However, SYNOPSIS can provide race free but not hazard free synthesis. Hazards must be checked manually. The hazard inherent when a mux is switched can be minimized with transmission gates.

4.1 The Nature Of SHE

To remember the states of an FSM, one can use flip-flops or latches. The flip-flops and latches remember their states by independent internal feedback loops. The combinational logic is used to make different state variables to interact with each other and the inputs.

The feedback loops are the most basic SHEs where the information is stored. In general, there are 4 types of feedback loops depend on how the information is stored.

1. Full-time feedback loops:

These full-time feedback loops are always *sensitized*¹. If there is a change in X, it will cause a change in Y which will confirm the change in X. The information stored in the full time feedback loops cannot be changed. The full-time feedback loops are mentioned here only to emphasize part-time loops.

2. Part-time feedback loops:

These part-time feedback loops store some of the time but are not sensitized all the time. In 2a, when $D=0$, it will store; When $D=1$, it will set. This loop once set cannot be cleared. This is overcome in 2b. When $S=R=1$, it will set; when $S=R=0$, it will clear; when $S=1, R=0$, it will store.

3. False feedback loops:

The false feedback loops are never completely sensitized.

4. Dependent feedback loops:

The dependent feedback loops are loops which are made of part of other loops. They do not need to be included in state analysis. Breaking the feedback at x removes both loops, so they are dependent.

1. The definition of "Sensitized" is: if x changes, y changes, then x is sensitized.

4.1.1 Example Circuits With Varies SHEs

A D master-slave flip-flop made with two D latches, as shown in Figure 2.6 (a), with opposite clock phase, is redrawn in Figure 4.0 (a). The ETFF of Figure 3.9 is redrawn in Figure 4.0 (b):

Figure 4.0 Feedback loops in (a) D master-slave flip-flop (b) ETFF

In this case, there are two part-time feedback loops in the DFF. Each loop memorizes the state for one phase of the clock. When one loop remembers the previous input, the second loop holds the output steady after a new output is computed.

The ETFF also has two part-time feedback loops. It also has one false feedback loop to feed the inverting output to the input. This false feedback loop does not store any information.

A DFF normally needs at least two part-time feedback loops to memorize one state bit, each part-time feedback loop memorizes half of the time. Sometime more feedback loops are needed. Recall the 6-state DFF shown in Figure 2.6 (b). It remembers its current state by utilizing two dependent feedback loops while using the third dependent feedback loop for output-encoding.

In a well designed latch based FSM, as shown in section 2.4.3.1, one part-time feedback

loop can memorize more than a half bit, but still less than one full bit. Since normally the more feedback loops there are, the more gates are required to form them, one should design an FSM with as few feedback loops as possible. Therefore, the ultimate goal of designing an FSM will be to use one feedback loop to hold one bit of the state variable.

4.2 Information Contained In Clock

The clock signal is the regulator of the *synchronous circuits*. It regulates the behavior of every modules in a synchronous system by telling them when to change their outputs. The use of a globally synchronous clock signal to control all state changes differentiates the *synchronous circuits* from the *asynchronous circuits*.

An FSM made with rising-edge-triggered DFFs only utilizes the timing information contained on the clock rising edges. The timing information contained on the clock falling edges is wasted. Thus the clock signal sends out twice as much timing information (the clock edges) as the FSM really requires, the clock has to run twice as fast, and the extra transition wastes power in the flip-flops.

Theoretically, edge-triggered flip-flops can be clocked by δ functions, as shown in Figure 4.1 (a), which has only one timing signal per cycle. The square wave clock signal, shown in Figure 4.1(b), not only contains timing edges but also two different voltage levels. These two voltage levels can be used to make a different output for even and odd cycles (half cycles).

Figure 4.1 Timing in the form of (a) δ function (b) conventional clock signal

This thought introduces the DLS design method, in which both levels/edges of the clock signal are utilized to trigger FSM state changes. The mop-flop circuits in this chapter will try to utilize the information stored in both levels of the clock signal by the means of feedback loops; while the circuits in Chapter 5 will try to utilize the information stored in both edges of the clock signal by the means of flip-flops.

4.3 Design DLS FSMs

4.3.1 Rules Of Synchronous Transition

A *synchronous transition* can be defined as the one which only changes state in response to a clock change. A *synchronous transition* is clearly distinguishable if the clock is the only input signal to cause the state transition (Rule 1).

However, when other inputs are used along with the clock signal as the conditions for state transition, it is harder to tell if a state transition is synchronous or asynchronous. One example is shown here, where a state transition may occur in response to a non-clock signal x .

To determine whether or not a machine changes states on clock edges, one must look at all the transitions leaving a particular state. In this example, the FSM sits in state A when clock (c) is low. When clock is high, the FSM will move to state F or G depends on the non-clock input x. Since the moment clock goes high the value of x is already determined, a *synchronous transition* is guaranteed.

In general, a transition is synchronous if all arrows leaving a state require condition c (are anded with c) and all self-loops require condition $\bar{c}$ or vice versa (Rule 2).

For this discussion one restriction, that c and x can not change at the same time, is needed. However, this is a normal restriction on all flip-flops.

It is also possible that a *synchronous signal*, one which is restricted to never change near the clock edge, is used for clocking (Rule 3), the resulting transitions will be synchronous provided:

- The total delay does not cause a setup-time violation on the next clock cycle.
- No other input will:
 - cause a race with the clock.
 - cause an error by arriving before the next clock when the logic design intended it to come after. (Hold time violation)

For a mop-flop one would specify a maximum clock-to-input for these inputs and maximum input-to-output propagation delay. Inside a mop-flop, the transitions do not need to be synchronous as long as they do not change on a clock edge.

If a state can only be reached through *synchronous transition*, the state is a *synchronous state*. If a state can be reached through an *asynchronous transition*, the state is an *asynchronous state*.

4.3.2 State Diagrams For The DLS FSMs

The two edges of the clock signal arrive one after another alternatively. In conventional rising-edge-triggered FSMs, the machine only changes states whenever there is a clock rising edge. This can be seen in Figure 4.2, which is a state diagram of a conventional edge-triggered two-bit Gray-code counter. The condition for the states to change are all the same, the clock rising edges. Therefore, the default way of drawing the state diagram will be not to show these clock rising edges on the state diagram, and leaves the arrows unlabeled. Since the clock falling edges are ignored, they will not be labeled on the state diagram either. Only if other conditions have to be satisfied before the state changes can happen, will these other conditions be labeled on the arrows.

Figure 4.2 *State diagram of a two-bit Gray-code counter*

In DLS FSM, the two edges of the clock signal have to be treated differently and both have to be labeled on state diagrams. This is shown in Figure 4.3.

Figure 4.3 State diagram of a two-bit Gray-code counter as the DLS FSM

Here the circuit of Figure 4.2 is implemented with as a DLS FSM. This machine can only change from the state A to the state B or from the state C to the state D when the clock is low. To change from the state D to the state A or from the state B to the state C, the machine has to wait for the clock's high levels.

In one way, the idea of the DLS FSMs is similar to the idea of the latch based FSMs. All the states have to be divided into the clock high phase states and the clock low phase states, the machine moves from the state of one phase to the state of the other phase alternatively. It should never move between the states of the same phase.

However, the DLS FSMs differ from the latch based FSMs in two ways: Firstly, in the latch based FSM, the SHEs are centralized feedback loops inside the latches, which only hold a half bit per loop. In the DLS FSMs, the SHEs are distributed feedback loops, which can utilize the information stored in clock signal levels (refer to section 4.2). Secondly, since the distributed feedback loops may interact with each other, Gray code state encoding is required. For latch based circuits, this is not required.

Since the only input of this circuit is the clock signal, all states in Figure 4.3 are *synchro-*

nous states, so are the Gray-code counters discussed in the following sections.

4.3.3 State Tables For The DLS FSMs

Since the clock is treated as one input, it has to be included in the state table.

4.3.3.1 State Table Representation

The latched loops and the inputs must uniquely define the next state. Thus in an *asynchronous state table*¹, if one underlines and/or shades the state variables which are latched at the present time, they and the inputs must uniquely define all next states.

One simple example is used here to illustrate this point:

The state table is:

state q1 q2		next c=0	state c=1
0	0	<u>0</u> 1	<u>0</u> <u>0</u>
0	1	<u>0</u> <u>1</u>	1 <u>1</u>
1	1	<u>1</u> 0	<u>1</u> <u>1</u>
1	0	<u>1</u> <u>0</u>	0 <u>0</u>

latched
latched & stable
stable state

For machines which have dead end states, all loops can be latched for all inputs. On entering other stable states, at least one state variable must become unlatched as a result of an input change and change value. At the same time, other state variables latch to the values they had as unlatched variables in the old states. This is described in more detail below.

4.3.3.2 Changing State

Changing state is the result of an input changes. To change state, some latched state variables q_k must become unlatched and change values. To maintain the new state one may:

1. Generally speaking, all DLS designs take clock as one input, thus the state tables are *asynchronous state table*.

1) Do nothing, in which case the unlatched new state is maintained by the input.

2) Latch the original state variable after a short delay. This gives a race. This type of circuit needs a tightly controlled pulse on some control input c .

3) Latch a new variable which was previously unlatched. This variable will not change value; it will just go from unlatched to latched, i.e. q_2 maintains its value but gets latched as c goes high.

The maximum number of stable states can be calculated by counting the number of latched loops for each input combination and adding them up as $\sum_{AllInputCombination} 2^{\text{Number of latched loops}}$. For example, the state table shown in section 4.3.3.1 has one latched loop for input $c=0$ and one for $c=1$. So the maximum number of states is $(2_{c=0}^1 + 2_{c=1}^1 = 4)$.

4.4 The DLS Mop-flops

The state diagram shown in Figure 4.3 is very similar to the one shown in Figure 3.7, and it is even simpler. As predicted in section 4.1, the resulting circuit should also be similar. Figure 4.4 is the derived schematic of the two-bit Gray-code counter designed as DLS mop-flop. Compare with Figure 3.8, it is the same as the ETFF described in Figure 4.0 with $e="1"$. It has two outputs and two feedback loops, each holding one bit of state variable. This machine:

- a) changes states only on clock edges
- b) has two independent outputs

It is a *synchronous circuit*, since all the states are *synchronous states*. It can also be consid-

ered a flip-flop with two independent outputs. Its outputs are the same as its state variables.

Figure 4.4 Schematic of the two-bit Gray-code counter

The two-bit Gray-code counter circuit shows the general way of making the mop-flops by employing the DLS method. There are four things one should remember when this design method is used:

- First: one has to label the clock edges on the arrows of the DLS state diagram. This is crucial for getting feedback loops to act as the SHEs.
- Secondly: One must assign states so that only one state variable changes at a time.
- Thirdly: Hazards must be avoided.
- Finally: The outputs of the circuit may be
 - the synchronous state variables.
 - signals logically derived from the synchronous state variables.

The rest of the design work can be carried out just like one does in the design of the conventional FSMs with flip-flops.

4.5 Making Bigger Mop-flops

The mop-flop is an FSM by its own and the more states it handles, the more state variables it has to have. Using the previous approach, a three-bit Gray-code counter built as DLS mop-flop

is derived. The state diagram is shown in Figure 4.5.

Figure 4.5 A three-bit Gray-code counter (a) state diagram (b) Symbol

The resulting circuit is shown in Figure 4.6 (a).

Figure 4.6 (a) The schematic of the three-bit Gray-code counter

The circuit in Figure 4.6 (a) is a synchronous FSM and it is also a flip-flop with three part-time feedback loops. Its structure can be shown more clearly by regrouping it into a mux/latch

based circuit as shown in Figure 4.6 (b):

Figure 4.6 (b) Mux based three-bit Gray code counter

Here, an unshaded state inside the circle is an unlatched state maintained by the input. On the next clock event, its value is latched. An unshaded state outside the circle is the unlatched state whose output changes following the input. Since there are two latched loops for either clock inputs, the maximum number of states is 8.

From Figure 4.6 (b) one can see, at any time there are two latched loops and the next state is uniquely define by these two latched states and the input.

- When clock is low, q3 can change while q1 and q2 are latching their previous outputs;
- When clock is high, either q1 or q2 (depends on q3's value) can change while other two outputs are latching.

This agrees with the Gray code state assignment where only one bit can change at a time.

The AND gates (q_3c and $\overline{q_3}c$) to gate the clock signal to M1 and M2 can be moved into M1 and M2, which makes the mux based mop-flop look more like the conventional master-slave DFF, i.e. latches plus non-gated clock. All the mop-flop circuits shown in this chapter have a similar structure to that of Figure 4.6 (b). Since the clock is the only input signal to the circuits, all the states are *synchronous states* and the circuits are *synchronous circuits*.

In this circuit there are three independent outputs. This requires three independent *sources*¹ to generate the outputs. There are a total of 3 part-time feedback loops where q_1 remembers 3/4 of the time; q_2 remembers 3/4 of the time; and q_3 remembers 1/2 of the time. The machine can remember 2 full bits at any given time. The third independent source is the clock input.

There are 6 latches, with 6 part-time feedback loops, in a conventional binary counter with master-slave DFF. At any time, 3 of the part-time feedback loops are latched and another 3 are unlatched. The maximum number of states it can remember is 8. It would be 16 if the clock was fed directly to an output.

Bigger mop-flops can be designed. One more example is the four-bit Gray-code counter. The state diagram is omitted, but the schematic is shown in Figure 4.7. This is a four-output flip-flop (mop-flop).

1. The *source* is defined as the state variable which is latched or an input includes clock.

Figure 4.7 *The schematic of a four-bit Gray-code counter*

One can make further bigger mop-flops in this way. No matter how big the circuit turns out, it always uses one feedback-loop per state-variable bit. However, as the number of states grows, the mop-flop becomes more and more complicated and the number of levels in the circuit also goes higher and thus increasing the delay. At a certain point, these big mop-flop just do not work anymore. One such example to show how the circuit complexity explodes is the eight-bit Gray-code counter made with the DLS method. Its schematic is shown in Appendix B.1. This monster contains hundreds of gates and is a dozen logic levels deep. It is slow and too complicated to be of any use.

One reason the circuit complexity explodes is that the number of states in an N-bit-state-variable flip-flop is 2^N . Just like in normal FSM design, when the number of states is large, people tend to do state partitioning on the giant FSM and use several smaller FSMs to do the equivalent work of that giant FSM, or, new design approaches have to be used. Based on experimental

results, the number of state variables a mop-flop can handle before it gets too complicated is about 4 or 5.

4.5.1 Interfacing The Counter With Conventional Circuits

Since the outputs of the above counters are identical to the outputs of the conventional Gray-code counters, there is no interface needed between the above counters running on half-speed clock to the surrounding conventional circuits running on full speed clock, except that a second half-frequency clock may be required. These counters can also be easily preloaded by adding a MUX in each feedback loop.

4.6 Synthesizing DLS Mop-flop Method

As mentioned before, one advantage of the DLS mop-flop is that it can be synthesized by the existing commercial softwares. However, for these conventional logic synthesizers to accept the designs based on the DLS method, the synthesizers must be tricked.

As discussed in section 4.3, the conventional FSM design method assumes that there is a rising clock edge for every state transition, so this timing information is not labelled on the arrows. The existing logic synthesizers will automatically add the default timing information back into the state diagram which is fed into them. In the DLS design method, it is not necessary to have a rising clock edge for every state transition. Instead, both the clock rising edges and the falling edges may drive the machine to the next state. If one feeds the state diagram with clock edges already labelled on the arrows to the existing logic synthesizers, they will blindly add one more clock rising edge for each state transition. To solve this problem, and make the state table format input of the DLS design legal to the synthesizers, a dummy clock signal has to be added to the already completed design to mimic the synthesizer's behavior. This is illustrated in Figure 4.8.

Figure 4.8 *The use of dummy clock signal*

In Figure 4.8, a complete state diagram of the DLS mop-flop two-bit Gray-code counter is shown. The dummy clock rising edges are shown by the dark arrows. The original FSM is modified into an FSM with the dummy clock as its clock signal and the real clock signal, the “c” labelled on dashed arrows, as its input. To move from one state to another, the input condition has to be satisfied, then the dummy clock rising edge will move the state to the next state. When one feeds the state diagram of the modified FSM into the logic synthesizers, the dummy clock signal is ignored and not fed into the synchronizers. The synthesizers will automatically add a clock signal, which equivalents to the dummy clock in the original design, back and treat the design as a conventional FSM. Therefore, after these small modifications performed in one’s mind, the state diagram of the modified DLS mop-flop design, as shown in Figure 4.8, is now legal to be fed into the synthesizers.

The cheated synthesizer, i.e. SYNOPSIS, will now generate the schematic based on the modified state diagram. It treats the dummy clock signal as the real clock signal and puts flip-flops accordingly. The resulting circuit is shown in Figure 4.9. The dashed arrows in Figure 4.8 correspond to the changes of q_1 and q_2 due to the changes of clock and outputs of two DFFs. The dark arrows correspond to the changes of outputs of two DFFs due to the changes of clock and inputs

of two DFFs.

Figure 4.9 The 2-bit DLS Gray-code counter generated by the SYNOPSIS

In Figure 4.9, those two FD1 flip-flops and the clk input are the results of the introduction of the dummy clock, they do not exist in the original design, thus can be removed. The inputs of the two flip-flops then can be directly connected to the outputs of the flip-flops and forming two complete feedback loops. As shown in Figure 4.8, after removing the dummy clock, the resulting circuit will be exactly the one designed originally. The final schematic of this 2-bit Gray-code counter is the one shown in Figure 4.4.

The circuits shown in Figure 4.6 (a), 4.7 and Appendix B.1 were all made in this way.

4.7 The Performance Comparison

The size, speed and power analysis are performed.

4.7.1 The Size Comparison

Assume the 6-state DFF is used in conventional circuits because of its simplicity and power efficiency. For an N-bit Gray-code counter, the flip-flops alone will contain $26N$ transistors. The combinational circuits are also large. The circuits of the conventional two-bit, three-bit, and four-bit Gray-code counters are shown in Appendix B.2 and their sizes are listed in Table 5.

The DLS mop-flops do not have any combinational circuits, their sizes are just the sizes of the mop-flops and thus very small. The size comparison is tabulated in Table 5.

Table 5: Size comparison of DLS and conventional Gray-code counters

(unit: transistors)	DLS mop-flop	Conventional method	Saving
2-bit	22	52	58%
3-bit	72	108	33%
4-bit	94	186	50%

The saving in the number of transistors ranges from 33% to almost 60%, the result is quite satisfactory.

4.7.2 The Speed Comparison

The speed of the conventional two-bit Gray-code counter is the same as the 6-state DFF, since their data delay paths are all about 3 gates¹. For DLS mop-flop Gray-code counters, there is no extra delay from the combinational circuit, all the delays are within the mop-flops. The delays of mop-flops are about 2 gates, 3 to 4 gates, and 4 to 6 gates (can be further improved), respectively. Therefore, the counters based on the DLS mop-flops are faster. The speed comparison can be tabulated as shown in Table 6.

Table 6: Speed comparison of DLS mop-flop and conventional Gray-code counters

	DLS mop-flop	Conventional method	Improvement
2 bits	2 gates delay	3 gates delay	33%
3 bits	4 gates delay	6 gates delay	33%
4 bits	6 gates delay	8 gates delay	25%

Since the speed of the circuit is decided by the worst case delay, the longest delays are

1. For speed estimation, the unit is gate, which means either AND/OR gate or NAND/NOR gate.

used in the above comparison. The above circuits also have fan out problem, which will slow them down about the same amount, i.e. when comparing the 4-bit mop-flop and conventional 4-bit Gray-code counters.

4.7.3 The Power Comparison

The power comparison is carried out with a clock speed of 10 MHz for conventional counters. Since the mop-flop Gray-code counters count at every clock edges, to get the same counting speed at the output, a half-speed clock signal is used. The power comparison is tabulated in Table 7. The HSpice files of the mop-flop counters can be found in Appendix A.2.

Table 7: Power comparison of DLS mop-flop and conventional Gray-code counters

	DLS mop-flop	Conventional method	Saving
2-bit	0.37 mW	0.69 mW	46%
3-bit	0.64 mW	0.96 mW	33%
4-bit	0.61 mW	1.43 mW	57%

Recall from the discussion in Chapter 3 that all the flip-flops take considerable power, whether they are active or not. So the power consumption of the circuit can be estimated by the power consumption of the active and inactive flip-flops in that circuit.

Since Gray-codes counter only changes one bit at a time, there is only one flip-flop active per clock cycle and it will take 0.46 mW power on average. The rest of flip-flops will take 0.24 mW power per flip-flop on average even though they are inactive. There is no combinational circuit in the two-bit Gray-code counter, so its total power consumption can be estimated by the addition of one active flip-flop and one inactive flip-flop, which is 0.70mW and agrees with the simulation result.

In the mop-flop counters, there is no power spent on idling flip-flops (since the mop-flop is always active). Therefore, its power consumption does not have to be proportional to its size.

Indeed, it is more or less proportional to the average number of active transistors. For example, there are 22 transistors in the two-bit counter and, on average of about 11 transistor switching per output bit change. The three-bit counter has 72 transistors, on average of about 24 transistors switching per output bit change, about twice as many as in the two-bit counter. Its power dissipation is also about twice as much as the two-bit counter. The simulation results agree with the analysis.

The most important power advantage of using DLS circuits may not come from anything discussed above, but from the saving in the clock distribution network, which may account for more than 40% of the total power consumption of the physical chip. A half-speed clock will at least half the power dissipated on the clock distribution network, not to mention the relief on layout tasks.

4.8 The Use Of Mop-Flop Type Gray-Code Counters

If the power saving is the main concern of the design, for counter type circuits, one should use mop-flop type Gray-code counters instead of the binary counters. Although there are only three mop-flop type Gray-code counters described so far, other length counters can be easily made. For example, a decade Gray-code counter is shown in Figure 4.10.

For counter initialization, a reset can be included by adding an AND gate in each output just before it is fed back.

Figure 4.10 *Count-to-10 Gray-code counter*

Another place where the mop-flop type Gray-code counters may be used is in high frequency dividers [For92]. From Table 6 one can see, since the delay of the novel two-bit counter is about 2/3 less than the conventional one, it can run at a faster maximum speed. At such high speed, the 46% power saving will be more significant.

As stated earlier, the length of the mop-flop type Gray-code counter is limited to about 5 bits.

4.9 DLS Controllers

The DLS method can also be used to build small controllers. Two such examples will be examined in this section. They also illustrate the various ways of making the DLS state diagrams. To simplify the exposition, only the combinational circuit and the SHE are analyzed, the output encoding part is omitted.

4.9.1 The Traffic Light Example

The traffic light example came from John Hayes' digital design book [Hay93]. This example is to design a controller for a traffic light which controls the intersection of two streets. Its task is to switch the lights on and off in a sequence that responds to the flow of the vehicles and pedestrians. If there are no pedestrians, the lights will automatically change to allow east-west (EW) traffic to follow and then north-south (NS) traffic to follow, and so on. If a pedestrian pushes the Walk button, all the lights will turn red and allow the pedestrian to cross the street. The simplest version of this controller can be described as shown in Figure 4.11, where the $x=0$ means no pedestrian and the $x=1$ means a pedestrian pushed the button.

Figure 4.11 State diagram for the traffic light controller[Hay93]

Since there are three states in the original state diagram and they form a triangle, this causes a race in the DLS controller, so the state diagram has to be modified. An extra state has to be added to break the triangle. Also, to use the DLS method, the machine has to be able to always travel from one phase of clock to another phase of clock.

This extra state can be added between state S_1 and S_2 . It will simply add some delays to the original design. A better approach is to utilize the extra state to do useful things, as shown in Figure 4.12.

Figure 4.12 Modified state diagram for the traffic light controller

In the new state diagram, the machine will loop between the state A and the state B when no pedestrian pushes the button, so the east-west traffic and the north-south traffic can get their green lights alternatively. If a pedestrian pushed the button, the lights will go red on both directions, then after the pedestrian passed, the side that previously had the green light will get the green light again. The modified state diagram preserves all the functions the original one had. By introducing an extra state C, one unfairness of the original design is also cured. Now after the lights go all red, the north-south traffic will not always get the green light first.

In Figure 4.12, the state C and D are *synchronous states* (refer to section 3.3.2.1.1). Consider state D. The only way to reach state D is from state A with $xc=10$. Since for $xc=00$ the machine will move from state A to state B, the transition to state D can only be *synchronous transition* according to the second rule of *synchronous transition* stated in section 4.3.1. The state D is therefore *synchronous state*. The same arguments apply to state C.

The state A and B are more complex. When there is no pedestrian, x stays “0”. In which case, the machine loops between state A and B on alternate clock edges. Since now the clock is

the only input, the transitions between state A and B are synchronous according to the first rule of *synchronous transition* stated in section 4.3.1. The state A and B are *synchronous states* under such circumstance.

When there is a pedestrian, x goes high and machine sits at state C or D. After allowing pedestrian enough time to cross the road, x is cleared. In general the transition to state A or B caused by x going low can be asynchronous. The only way to make sure states A and B are synchronous is to control x going low so that it only happens following a clock edge, i.e. make the x synchronous¹. Then all the transitions to state A and B are *synchronous transitions* and state A and B are *synchronous states* at all time.

When performing the state assignment to the state diagram shown in Figure 4.12, one has to make sure all the adjacent states differ in only one bit. This is required to ensure there is no race when the machine moves from any states to their next states. The resulting circuit is shown in Figure 4.13 and its HSpice file is listed in Appendix A.2.4. This is a DLS mop-flop with two independent outputs. Since in the modified circuit, both levels of the clock signal are used to trigger the state transition, only a half-speed clock signal is required.

Figure 4.13 DLS traffic light controller

-
1. Synchronizing x would require an extra DFF. However a properly designed flip-flop based machine would also require this flip-flop to avoid setup/hold time violation in the controller.

From Figure 4.13 one can see, the resulting circuit only has 46 transistors, even less than the number of transistors in two 6-state DFFs. Detail comparisons are tabulated in Table 8.

Table 8: Comparisons of the traffic light controllers

	DLS mop-flop	Conventional	Saving
Size	46 transistors	60 transistors	23%
Delay	4 gates	4 gates	0%
Power	0.56 mW	0.95 mW	41%

Of course, this particular circuit is never going to run under very high speed, so the power and speed are not important design parameters to be concerned with. But this example does show that a general controller can be made into DLS mop-flop and can achieve savings on both the size and the power consumption.

4.9.2 The Game Controller Example

This game controller example came from the undergraduate digital design lab at the author's department¹. The state diagram of the conventional controller can be seen in Figure 4.14. In this example, the same speed clock is used for both conventional circuit and DLS circuit. It illustrates that even run at the same speed clock, the DLS mop-flop circuit may still save power over the conventional design.

1. Refer to Appendix C for details.

Figure 4.14 State diagram of the game controller

At the start of each game, the machine will be reset to the state R. On the next clock rising edge, the machine will move to the state A and wait for a “ready” input, i.e., ROUT. Upon the arriving of the “ready” signal, the machine will move to the state B and wait for the players to push buttons. If one of the players pushed the button before the “ready” signal, the machine will still move to state B, it is up to the rest of the game circuits to decide which player pushed illegally. The push button signal will bring the machine from the state B to the state E, at the same time the PUSH signal is cleared by other circuits. Then the machine moves directly to state D. At the state D, machine will check if any player tries to steal a round by holding the push button and not release it from the beginning of the round, the machine will move to the state B and the player will be punished. Otherwise, the machine will go back to the state A and the next round starts. The schematic of this circuit, is shown in Appendix B.2.4.

As in the last example, the original state diagram has to be modified to make a DLS state diagram. A few extra states have to be added to break any triangles in the state diagram and make sure the resulting one still satisfies the original timing requirement of the controller. This is illustrated in Figure 4.15.

Figure 4.15 State diagram for DLS game controller

In Figure 4.15, machine starts from state R, when clock goes high, the machine moves to state A and waits for either the ready signal or an illegal push. If one of the conditions is satisfied, the machine will move to state B when clock is low. The longest time the machine may wait for clock to go low is the time the clock is high, i.e. a half-clock-cycle response time. In the original state diagram, this time was one clock cycle. So the DLS game controller can response faster than the conventional controller.

If the machine moves to state B due to ready signal, it will sit on state B and wait for a push signal to move to state E. The same analysis applies as above. The longest time the machine takes to response to the input change is a half-clock cycle. This is faster than the conventional controller.

State E and D are two waiting states. In DLS state diagram they are replaced by 4 states which gives the same total waiting time. Therefore, the modified machine will response to the inputs change faster and keep the waiting time the same.

From the analysis of *synchronous* and *asynchronous states* performed in section 3.3.2.1.1 and 4.3.1, it is clear that transitions into state E', D, D' and A are synchronous. The *asynchronous state* E should not be a problem since one can take the output from E' if required. The only potential problem is the state B. When the machine moves from D to B, the state B is also synchronous. But when the machine moves from A to B, the transition may not be synchronous. However, since the ROUT signal is generated by a *synchronous circuit* and PUSH signal is also synchronized before fed into the controller, one can always regulate them to arrive when clock signal is high. So the transition from state A to B is synchronous according to the third rule of *synchronous transition* stated in section 4.3.1. In this case, the output taken from state B will also be synchronous.

It is also possible that the output of state B is used by some states which are latched during clock low period. In this case, it makes no difference to those next states on which part of the clock low period the transition to state B occur.

The state diagram shown in Figure 4.15 is used to derive the DLS mop-flop. As mentioned in traffic light controller example, when performing the state assignment to the new state diagram, care has to be taken to ensure there is only one state variable bit change when the machine moves from one current state to its next state. The derived circuit schematic is shown in Figure 4.16 and the HSpice file can be found in Appendix A.2.5.

In this simulation, a 10 MHz clock is used for the DLS mop-flop, the same as the clock used in the conventional controller. The gate counting is performed on the circuits shown in Figure 4.16 and Appendix A.2.5, which did not take the output decoding stage into consideration. The reason is to focus on the design of the FSM itself. The mop-flop controller has a little worse fan out problem than the conventional controller, i.e., 6 vs. 4, but this can be reduced by rearranging the gates. Without doing so, the delay estimation based on the gate counting of the longest delay path may not be accurate. More detailed analysis is required to estimate both circuit's speed, which was not performed since this thesis focuses on low power rather than high speed.

The comparisons of conventional game controller and mop-flop game controller are summarized in Table 9.

Table 9: Comparisons of game controllers

	DLS mop-flop	Conventional	Saving/ Reducing
Size	122 transistors	146 transistors	16%
Delay	6 gates	8 gates	25%
Power	0.6 mW	1.2 mW	50%

Figure 4.16 *The DLS mop-flop game controller circuit*

4.9.3 Concerns Of The DLS State Diagram Generating Methods

The state diagrams for the Gray-code counters are the most straight forward. To make

counter circuits' state diagrams, one needs only to write down the states following the counting sequence specification, then labels the clock signals on each arrow of the state transition. No extra states have to be added and all the states in the state diagram are synchronous. Since the DLS state diagram can not have any cycles¹ with an odd number of edges, in case of making odd number counting counter, one has to design a larger even number counting counter then reset all the bits to zero once it reaches the required counting. By doing so, one clock high or clock low signal might be wasted when one resets all the bits to zero.

General purpose controllers are harder to design. First of all, extra states may have to be added to break any cycles with odd number of edges. The reason behind is that one cannot make a race-free state assignment under such circumstance. After adding those extra states, the designer has to check if the added delays will affect other circuits' normal operation. If it does, adjustments have to be made. Sometime one may even have to adjust other circuits to make them work with the controller.

Then the designer has to make sure that all the outputs taken from the machine have to be taken from *synchronous states* or the inputs of the machine have to be synchronous, if this is required when interfacing this FSM with other circuits. As discussed in traffic light controller and game controller circuits, the *asynchronous states* can always be modified into *synchronous states* if the designer can make sure the inputs arrive at the right phase of clock.

All circuits have to be checked to be hazards free to ensure the proper operation. As discussed in Section 3.3.2, the hazards not only increases the power consumption of the circuits, but also may cause the malfunction of the FSMs. If the hazards is a problem, then the cures discussed in Section 3.3.2 have to be applied.

1. A cycle in a graph is a closed path through two or more edges.

4.10 Mop-flops Based Circuits

The DLS mop-flops can be also used as the building blocks to form more complex circuits. A few examples will be discussed in this section.

4.10.1 Making Longer Counters

The DLS N-bit Gray-code counting mop-flop becomes too complicated when the N is bigger than 5 or 6. That means the mop-flops made so far can not count above 2^6 . For circuits like DRAM refresh circuits or dual-slope A/D convertors, counter counts of 2^{10} or more are required. This problem can be solved by building bigger counters with the Gray-code counting mop-flops as the modules.

One type of counter which is considered the most power-efficient and most compact in size, is the ripple counter. They are widely used in applications where strict clock synchronization is not required¹, such as in DRAM refresh circuits. The general structure of the ripple counter is shown in Figure 4.17.

Figure 4.17 General structure of the ripple counter

1. Ripple counters can be thought of as a mop-flop where different output bits have different Clock High Q Valid (CHQV) time. As long as the longest delay time is shorter than clock cycle time, it can be considered a *synchronous circuit*.

By using the same idea, the two-bit, three-bit, and four-bit DLS Gray-code counting mop-flops shown in section 4.3 and 4.4 can be cascaded to form longer counters. For example, two 4-bit DLS Gray-code counting mop-flops are cascaded as shown in Figure 4.18.

Figure 4.18 Ripple mop-flop counter. The dashed line shows where the upper counter changes.

However, since the four-bit flip-flop module counts on the both clock edges and the Q4 is at the same frequency as Q3, the cascaded counter shown in Figure 4.18 only counts to 2^7 . In other words, four bits plus four bits equals seven bits. In this circuit, both mop-flops are counting in Gray-code but the overall 8 bits counting is not in Gray-code. There will be two bits changes whenever the upper mop-flop is counting. Think of the ripple counter, whenever there is a ripple carry, both flip-flops are changing outputs.

The performance of this seven-bit cascaded mop-flop is better than the corresponding seven-bit ripple counter. The worst data path delay in the seven-bit ripple counter is about 28 gates. While in the cascaded mop-flop counter this value is only 12 gates, about 2.3 times less. The HSpice simulation result is even more favorable towards the cascaded mop-flop counter. To count at the same speed, the cascaded counter runs at half clock speed, i.e., a 5 MHz clock while the ripple counter runs at 10 MHz clock. Since the cascaded counter wasted one bit counting abil-

ity, its size is a little bigger than the ripple counter. The Table 10 summaries all the comparisons.

Table 10: Comparisons of seven-bit counters

	Cascaded(mop-flop)	Ripple(Binary code)	Saving
Size	188 transistors	182 transistors	-3%
Speed	2.5 ns	7 ns	64%
Power	0.79 mW	1.0 mW	21%

Even longer counters can be made by simply cascading more modules to the existing one.

4.10.2 A Pure Mop-flop Gray-Code Up/Down Counter Module

By cascading the Gray-code counting mop-flops, one not only loses one bit of counting ability, but also loses the property of counting in pure Gray-code. The cascading method is only good for arbitrary number counting, not good enough for making longer pure Gray-code counters.

In some practical circuitry, the longer Gray-code counter should not only count in pure Gray-code, but also have the ability of counting both upward and downward, as the control signal indicates. For this purpose, the two-bit, three-bit, and the four-bit Gray-code counting mop-flops are too simple to be used as the building blocks. A more complex building module has to be created before the up/down pure Gray-code counter can be assembled.

First an enhanced four-bit up/down Gray-code counting mop-flop module was designed. The state graph is illustrated in Figure 4.19. If the direction signal is kept constant, all the states in Figure 4.19 are *synchronous states*. When the direction signal is changing, care has to be taken to ensure that it will not result in an *asynchronous transition* or at least no damage will be caused. This can be easily achieved with a synchronous “e” signal.

Figure 4.19 State diagram of the four-bit up/down Gray-code counting mop-flop module

The major difference between this mop-flop module and the one shown in Figure 4.5 (a) is that an extra direction control signal “e”, is added. When the “e” is high, the counter will count in one direction; when the “e” is low, the counter will count in another direction. For example, assume the “e” is low and the counter module is counting “1101”->“1111”->“1110”, and the counter is on the state “1110” and the “c” is low. If one toggles the “e” signal at this moment, the next state of “1110” will be “1111”, and the machine will start counting backwards.

There are two special states in the 16 states of Figure 4.19, they are the state “0000” and the state “1000”. No matter whether the count is upward or downward, starting from those two states, the next states are the same. Therefore, in the state diagram of Figure 4.19, there is only one next state for the state “0000” and “1000”.

One may also notice that when the machine is changing counting direction, the direction signal change will move it backward immediately on the same clock signal phase, i.e. from state “1110” to state “1111” then back to state “1110”. It seems the machine does one extra count. This

may not be true. If the direction control signal's changing is caused by one negative counting event, then the machine should move one state backward, i.e., subtract one from the current counts. Therefore, designer has to make sure that the direction signal only changes when the negative counting starts. The generation of the direction signal has to also allow the counter to operate properly.

The state diagram, after the additions described in section 4.6, can be fed into a synthesizer. The derived circuit schematic is shown in Figure 4.20.

The circuitry shown in Figure 4.20 is only the mop-flop part of the complete circuitry. Now the direction control signal has to be generated. There are two events which will both affect the direction control signal. The first event is the external up/down control signal, which depends on the external counting events. The second event is the internal direction switching, which is automatically generated when the four-bit flip-flop reaches state "0000" and "1000". In other words, if the external up/down signal remains the same, when the machine reaches state "0000" or "1000", it should automatically change its counting direction. The overall direction control signal "e" can thus be generated as shown in Figure 4.21.

Figure 4.20 Four-bit Gray-code up/down counting mop-flop module

Figure 4.21 The direction control signal generation circuit

By referring to the Figure 4.19 one can see, the machine will only reach the state “0000” or “1000” when the clock signal is low. So the NAND gate will open and its output is pulled up by the clock signal. Therefore, no false clock falling edge will be generated by the clock-gating NAND gate. The signal E of the D flip-flop’s output will only change once every time the machine crosses the state “0000” or “1000”.

To control the timing of the direction signal, the timing of up/down signal has to be controlled, so the machine will not do extra/less counting when it changes counting direction.

A complete four-bit pure Gray-code up/down counter mop-flop module can be assembled by combining both circuits in Figure 4.20 and Figure 4.21. Its HSpice file is listed in Appendix A.2.6. Now the longer counters can be directly made by cascading the newly derived modules. For example, the eight-bit pure Gray-code up/down counter mop-flop module can be made as shown in Figure 4.22.

Figure 4.22 An eight-bit pure Gray-code up/down counter module

The circuit structure shown in Figure 4.22 is very similar to the one shown in Figure 4.18. The only difference is that the new clock signal for the second module is not the output Q4, but the E signal shown in Figure 4.21. Unlike the signal Q4 in Figure 4.18, which will change twice when the machine moves from state “0000” to “1000”, the E signal only changes once every time the machine crosses the state “0000” or “1000”. Therefore, the cascading of the two four-bit mop-flop modules results an eight-bit pure Gray-code up/down counter mop-flop module.

Longer counter can be derived by simply adding more four-bit up/down counting mop-flop modules to it. However, it is not a complete Gray-code counter since the most significant bit of the mop-flop module is different from the one of the Gray-code counter. Therefore, the most significant bit of an N-bit Gray-code counter should be designed differently. The simplest structure one can use is a single Toggle flip-flop with the clock input driven by the E signal from previ-

ous module. To get arbitrary length counters, one may also have to design a three-bit and a two-bit cascaded Gray-code up/down mop-flop modules.

The conventional eight-bit Gray-code up/down counter contains eight conventional D flip-flops and the combinational circuitry is very complex. The eight flip-flops alone will take 2.14 mW power¹ while the whole cascaded counter shown in Figure 4.22 only takes 1.09 mW power. The power saving is significant.

4.10.3 General FSMs With Up/Down Mop-flop Gray-code Modules

General FSMs can be made with up/down mop-flop Gray-code modules. For example, an FSM with upto 16 states can be made by using two mop-flop modules as shown in Figure 4.23.

Figure 4.23 Mop-flops based FSM

By doing so, the clock frequency can be halved, thus saves power. It can also remove the clock-gating task which further saves power. The circuit will also be more compact by making mop-flops into standard cells.

The general 2-output mop-flop M1 and M2 can have two inputs: u/d and e. The output

1. At anytime, 7 idling flip-flops and 1 active flip-flop take $7 \times 0.24 + 1 \times 0.46 = 2.14$ mW

sequence of q1q2 is:

- u/d=1, e=1: 00 -> 01 -> 11 -> 10 -> 00
- u/d=0, e=1: 00 -> 10 -> 11 -> 01 -> 00
- u/d=x, e=0: 00 -> 00, 01 -> 01, 11 -> 11, 10 -> 10

The M1 mop-flop in the FSM shown in Figure 4.23 can stay in any one of the 4 states while M2 mop-flop moves up or down, or vice versa. Or both M1 and M2 can move either up or down at the same. If the circuit is disabled, both mop-flops will stay in their current states. Since each mop-flop can change one bit at a time, the FSM can change up to 2 bits at a time.

A Random FSM example is shown in Figure 4.24.

State	Next State					State		Next State		Inputs		Inputs generated by feedback loops
	u1 00	u2 01	u1 00	u2 01	e1e2	AB	CD	AB	CD	u1,e1	u2,e2	
1	-	-	2	-	11	00	00	01	01	1,1	1,1	$u1 = (B + \overline{A})\overline{C}$ $u2 = \overline{B} + C$ $e1 = A + \overline{C}$ $e2 = D + \overline{A}$
2	-	-	3	3	10	01	01	11	01	1,1	-,0	
3	-	3	3	4	11	11	01	10	00	1,1	0,1	
4	-	5	5	-	01	10	00	10	01	-,0	1,1	
5	-	6	-	-	11	10	01	11	11	0,1	1,1	
6	-	7	-	-	11	11	11	01	10	0,1	1,1	
7	-	1	-	-	11	01	10	00	00	0,1	1,1	

u1=u/d1, u2=u/d2, -=don't care

Figure 4.24 Random FSM example using the general model of Figure 4.23 (a)

The DLS FSMs made this way have less limitations than those shown in Section 4.9, while enjoying the power saving from the half-speed clock. Since more than one bit can change at any time, odd number of edges in a cycle is possible. By adding more modules, larger circuits can be built. Since the hazards and races are dealt with inside of each module, when making larger circuits, one does not have to worry about them any more. The design can be carried out more like in the conventional DFF based designs with each mop-flop module treated as one functional block.

4.11 Summary

In this chapter, the design methodology of the mop-flops, based on the DLS method, was discussed. The FSMs based on this method eliminate the power and logic wasted on the inactive half clock cycles. The resulting circuits combine the combinational circuits and the SHEs together, and form various mop-flops. Comparing with the conventional FSMs, these new FSMs are smaller in size, faster in speed and economic on power. The maximum saving/improving achieved is listed in Table 11.

Table 11: Maximum saving achieved by mop-flop designs^a over conventional designs

	Power	Size	Speed
Saving/Improving	57%	58%	64%

a. Limited to fairly small circuit modules.

Some counter examples and controller examples were shown in this chapter to illustrate the ways of applying this design method. For controller designs, the state diagram of the conventional design method can usually be modified to suit the purpose of the DLS design method. However, care has to be taken to avoid races in state assignment and taking outputs from *asynchronous states*.

For large circuits, the proper smaller mop-flops can be designed first. Then these smaller mop-flops can be made into standard cells and form the bigger circuits. By doing so, circuits design can be more flexible since triangle state assignments are possible. The timing, race and hazards issues can also be dealt with inside the standard cells. As always, care has to be given to the synchronization and potential false clock edge issues inside the standard cells. Further research on designing mop-flop modules based circuits is required.

The example of the N-bit Gray-code up/down counter shown in this chapter dissipates much less power than the conventional circuits. But its speed is slower than conventional DFF based circuit, its size is larger than the ripple counter, and its maximum output delay increases as

counter gets larger. A comparison of different counter circuits is listed in Table 12.

Table 12: Comparison of various N-bit counter circuits

	Speed	Power	Area
mop-flop (Gray code)	slow	lowest	small
DFF (Gray code)	fast	high	large
ripple (Binary code)	slowest	low	small

By using the DLS method with conventional flip-flops, one can achieve smaller, faster and more power-efficient designs, as will be shown in next chapter.

CHAPTER 5

DOUBLE-EDGE-TRIGGERED FSMs

5.0 Introduction

According to the discussion in Chapter 4, one may think that the DLS method can be only used to make mop-flops. Indeed, the DLS method can also be used to make FSMs with the conventional D flip-flops as the SHEs. The FSMs made this way can be called double-edge-triggered (DET) FSMs and they trigger on both edges of the clock signal. One easy way of thinking this is illustrated in Figure 2.11 and redrawn here.

The N-bit Gray code up/down counter shown in last chapter was very power-efficient, its power dissipation is even smaller than a conventional three-bit Binary counter. Its only disadvantage is that it is a ripple type counter and its maximum output delay time increases as the counter gets longer. In this chapter, this problem was cured by employing the DLS method with the use of the conventional DFFs. The resulting circuits not only get rid of the above disadvantage, but also have a smaller size and lower power dissipation. Their overall performance out-perform previous Gray up/down counters found in the literature. The best of which can be found in U.S patents,

which are listed in References at the end of the thesis.

5.1 Count_to_8 Circuit With Two DFFs

As mentioned before, the clock is the control signal of a synchronous FSM. It signals the FSM when to move from the current state to the next state. The conventional flip-flop based FSMs only utilize one edge of the clock signal. The mop-flop circuits in Chapter 4 utilizes the information contained on both levels of the clock signals. It is also possible to design FSM which can utilize the information contained on both edges of the clock signal.

Latches contain single part-time feedback loop can utilize either clock high level or clock low level. Fundamentally, flip-flops contain at least two part-time feedback loops and can utilize both levels of the clock signal internally. Externally, the flip-flops behave like edge-triggered devices. To build a FSM which can utilize both rising and falling edges of the clock signal, the FSM has to contain both rising-edge-triggered flip-flop and falling-edge-triggered flip-flop.

In conventional edge-triggered flip-flop circuits no distinction is made between the first and last half of the clock cycle. However using the clock as data can effectively break each state into two states. A state under clock high can have a different meaning than it under clock low. In one sense the clock is replacing one storage element.

This concept can be better illustrated by an example. In conventional circuit design, say, for a conventional counter which can count to 8, one needs three flip-flops. Here, the same circuit can also be made with only two DFFs, with the help of the alternating clock rising or falling edges. The behavior level description is shown in Figure 5.1.

Figure 5.1 State diagram of count_to_8 circuit with only two flip-flops

From Figure 5.1 one can see, the output of this circuit will circle from “00”, “00”, “01”, “11”, “10”, “10”, “11”, “01” and back to “00”, and so on. From the first “00” to the next “00” one round later, there are 8 states the machine has to travel. Therefore, although there are only 4 states in this FSM, it can be used to count 8 events. Only two flip-flops are needed to build this circuit.

The reason that this machine can count to 8 with only two flip-flop is because it treats the same state with a different clock phase as two different states. In other words, both of the clock edges are utilized. For example, when this machine is in the state “11” and the clock signal is logic “0”, it will move the machine to the state “01”, or “010” where the third bit “0” is the clock acts as a Pseudo-state bit. Since the machine is edge-triggered, the clock low level will not move the machine from state “01” back to state “11”. So the FSM made this way is called double-edge-triggered (DET) FSM instead of double-level-sensitive (DLS) FSM.

If the machine moved to state “01” from state “00” as the result of clock rising edge, the third bit will be “1”. This third bit is used to indicate which next state the machine is going to move to and can also be used to generate output.

The schematic of the circuit is shown in Figure 5.2. The corresponding HSpice file is

listed in Appendix A.3.1.

Figure 5.2 *Count_to_8 or 2-bit Gray code counter module circuit with only 2 flip-flops*

The circuit in Figure 5.2 contains two conventional DFFs, but these two flip-flops are triggered on different clock edges. It was derived from the state diagram in Figure 5.1 by hand. Similar to those DLS circuits shown in Chapter 4, this circuit also works twice faster than the single-phase circuit under the same clock speed.

5.2 The Synchronous N-Bit Gray Code Counter Module

The circuit shown in Figure 5.2 counts the same sequence as a two-bit Gray code counter module. Therefore, it is also the 2-bit Gray code module. The task now is to find a way to cascade more bits to the existing module.

The state diagram of the Figure 5.1 would be a good start point. A 3-bit Gray code counter module can be easily derived from the existing 2-bit Gray code counter module. The behavior level description is shown in Figure 5.3.

Figure 5.3 State diagram of the 3-bit Gray code counter module

Comparing Figure 5.3 with Figure 5.1 one can see, other than the most significant bit, the state diagram of Figure 5.3 is made up of two 2-bit Gray code counter modules shown in Figure 5.1. When the machine travels in the shaded area, the most significant bit is “1”; when the machine travels in the unshaded area, the most significant bit is “0”. The transition occurs at the state “010” and “110”. Therefore, the circuit realization of the 3-bit Gray code counter module can be easily achieved by adding one enabled DFF to the circuit of the 2-bit Gray code counter module circuit. Its schematic is shown in Figure 5.4 (a) and its HSpice file is listed in Appendix A.3.

The enabled DFF shown in Figure 5.4 is indeed the GFF discussed in Chapter 3. To perform the same function, the EDFF and ETFF can also be used here. The reason to use GFF here is to save two gates given that there is no false clock edge problem in this circuit.

Figure 5.4 The schematic of the (a) 3-bit (b) 4-bit Gray code counter module

Treated as a three-bit Gray code counter module, the above circuit can be further cascaded. Similar to the derivation of the 3-bit Gray code counter module circuit from the 2-bit Gray code counter module circuit, the behavior level description of the 4-bit Gray code counter module circuit is also constructed by two copies of the state diagram of Figure 5.3, as shown in Figure 5.5.

The circuit level realization of the state diagram shown in Figure 5.5 is also simply an extra enabled DFF on top of the circuit shown in Figure 5.4 (a). In this case, this DFF will be enabled when $q_1q_2q_3$ states are “100”, as shown in Figure 5.4 (b).

Figure 5.5 State diagram of 4-bit Gray code counter module

An N-bit Gray code counter module can thus be built except for the most significant bit. As discussed in Chapter 4, the most significant bit is different from all other bits and needs special attention. For example, assume N equals 4, then the most significant bit q_4 has to be derived from a slightly different state diagram of Figure 5.5. The q_4 bit will not only toggle when $q_1q_2q_3$ states are “100”, but also when $q_1q_2q_3$ states are “000”. In other words, the enabled DFF of the q_4 bit is now enabled twice as often and the circuit turns out to be even simpler.

When the N is large, the activity of the flip-flops of the most significant bits decreases. As calculated in Chapter 3, when the flip-flop’s activity below 68%, the GFF configuration can shut down the inactive flip-flops and thus save some power. For the module circuit shown above, the GFF can be used for any extra cascading stages.

5.3 The Synchronous N-Bit Gray Code Up/Down Counter Module

The N bits Gray code counter module discussed in section 5.2 can be easily modified into the N bits up/down Gray code counter module.

If the counting direction changes can be restricted to only right after clock rising edges, the modification will be the easiest. By looking at the circuit shown in Figure 5.4 one can see, that the only DFF actives on the clock falling edge is the q1 flip-flop. To make the counter count in the other direction, one can generate a “direction control signal”¹ which always arrives right after the clock rising edge and lasts for one clock period long. The function of this “direction control signal” is to mask the clock falling edge fed into the q1 flip-flop by pulling the clock signal high for one clock period², or simply disable the q1 flip-flop for one clock cycle. For example, assume the machine is on the state “0111” of Figure 5.5, it counts in the direction of “0110” -> “0111” -> “0101”. The next clock arriving edge is the falling edge, but it is blocked by the “direction control signal”. Since the q1 flip-flop can not switch on this falling edge clock signal and all other flip-flops in the circuit are rising-edge-triggered flip-flops, therefore, the machine will stay in the “0111” for this clock signal. The next time the clock edge arrives, it is the clock rising edge. Starting from the state “0111” with a clock rising edge, the machine will move back to the state “0110”, the counting direction is thus changed.

In case the actual counting direction changes may also occur right after clock falling edges, the “direction control signal” has to be applied to rest of the flip-flops to enable and disable them in a complementary fashion of q1 flip-flop.

1. Or “enable” signal.

2. i.e., clock-gating.

5.4 The Performance Of The Circuit

The proposed N-bit Gray code up/down counter is very compact, it is almost made up of enabled flip-flops only with no extra gates. Therefore, a few of the best United States Patents¹ were used in this section for the purpose of comparison.

Use the 5-bit Gray code counter module as an example. The module made by the proposed method used only 5 DFFs and 4 extra gates. The module based on US patent 5,355,396 contains 6 DFFs plus 13 extra gates. The module based on US patent 5,428,654 contains 5 DFFs plus 25 extra gates. As for the power, simulation shows that the proposed module only takes 1.16mW power. For US patent 5,355,396, the power dissipated on 6 DFFs alone will be $5 \times 0.26 + 1 \times 0.46 = 1.76\text{mW}$. For US patent 5,428,654, the power dissipated on 5 DFFs alone will still be $4 \times 0.26 + 1 \times 0.46 = 1.50\text{mW}$. By taking the power dissipated in 13 and 25 extra gates into consideration, their power consumption will be even higher. This is summarized in Table 13.

Table 13: Size and power comparisons of proposed 5-bit Gray code counter module vs. patented modules

	proposed scheme	US patent 5,355,396	saving	US patent 5,428,654	saving
size	34 gates	49 gates	31%	55 gates	38%
power	1.16mW	> 1.76mW	> 34%	> 1.50mW	> 23%

The speed of the proposed scheme and those two patented schemes are comparable.

5.5 Applications Of Gray Code

The counters are widely used in almost every digital computer, programmable controller, dual-slope A-to-D converter, all digital PLL and other digital systems. Binary code counters are most commonly used since binary code is a weighted code and suitable for arithmetic uses.

1. The two best circuits are: US patent 5,355,396 & US patent 5,428,654. Others are US patent 5,448,606; US patent 4,780,894; US patent 5,191,425; US patent 4,528,665; etc.

Gray code has the property that they only change one bit at a time and thus provide a glitch-free counting sequence. This unique characteristic makes Gray code counters essential in some low power, low noise and race-free applications.

For example, Gray code is widely used where asynchronous world has to interface with the synchronous world, as shown in Figure 5.6:

Figure 5.6 The interface between asynchronous world and synchronous world

In above figure, the asynchronous input bit stream is counted by the recovered clock. Then the counting is sampled into the *synchronous circuit* by the system clock. Since the recovered clock is asynchronous to the system clock, the n-bit counting output may change at any time even on the system clock's rising edge. If binary code counter is used, error may occur if only part of the n-bit counting changes are captured by the system clock. For example, assume $n=4$ and the outputs are changing from 0111 to 1000. The comparator at the synchronous world may capture anything from 0000 to 1111. Thus the wrong control signal may be generated. If Gray code counter is used instead, since there is only one bit change at a time, the worst thing can happen is that the comparator did not catch the change and the error is only one LSB.

This is the major reason for using Gray code and the applications most cited in. A few dozens of patents relate to such application of Gray code counter were filed. More details can be found in the Reference section of the thesis.

Gray code also has the advantage of minimizing the coupling noise and power consumption. For example, if Gray code is applied to bus addressing, on average, 50% of bus switching activities can be eliminated and resulting in a significant power saving. Some research had been done by implementing Gray code addressing in a RISC-like pipelined processor VLSI-BAM (VLSI-Berkeley Abstract Machine)[Su94]. Hitachi Ltd. of Japan also reported the operating speed improvement and coupling noise reduction by applying Gray code in their semiconductor memory device [Sat94].

5.6 Summary

The proposed DET N-bit Gray code counter achieved impressive performance improvement over conventional best known circuits. It saves power by:

- Gating the clock or use ETFF
- Using half-speed clock

Unlike the mop-flop circuits shown in Chapter 4, since the DFFs are used to build the circuits in this chapter, these circuits do not have to be checked for races and hazards. They are also guaranteed to be synchronous.

For small Gray code counters, i.e. less than 5 bits, mop-flop counters should be used. For larger Gray code counters, the proposed N-bit counter module with flip-flops has size, speed and power advantages over the mop-flop counters.

Unfortunately, this design method is currently not applicable to non-counter type circuits.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 Thesis Summary

Three different types of circuits were discussed in this thesis.

Circuit 1 Clock disabling toggle flip-flop

Properties Clock disabling to
save flip-flop idling
power

Advantages Saves power for low
activity flip-flops

No false clock rising/

falling edge problem

Output will not change on clock edges

Limitations Uses more power under high activity

Larger flip-flop size than conventional DFF

Usage Any low activity flip-flops

Circuit 2 **Mop-flop**

Properties Multiple outputs flip-flop
 Both clock levels triggered
 Synchronous outputs
 Can be made into standard
 cells

Advantages Smaller circuit size
 Power saving
 Use half-speed clock
 Can make controllers

Internal Limitations

Needs to be hazards free
 Has to use race-free state assignment
 Harder to make large single mop-flop
 Needs odd, even states

External Limitations

Used in synchronous circuit has setup,
 hold and clock-to-output limitations

Usage

Small Gray-code counters
 Small controllers
 Large controllers with mop-flop modules

Circuit 3 **Double edge-triggered circuit**

<u>Properties</u>	DFFs trigger on either edges and circuit triggers on both edges
<u>Advantages</u>	Can use half-speed clock No need to use race-free state assignment Smallest circuit size and power dissipation
<u>Limitations</u>	Not yet generalized to anything but Gray code counters Needs odd, even states $t_{\text{cycle}} = 1/2 t_{\text{clock cycle}}$	
<u>Usage</u>	Gray code counters Should be able to make more general type circuits	

6.2 Thesis Contributions

- 1) DLS mop-flops and circuits use mop-flop modules.
- 2) DET counters.
- 3) Power saving with enabled toggle flip-flop.

The major contribution of this thesis working was the proposed DLS mop-flop design methodology.

6.2.1 The Enabled Flip-flop Method

The enabled flip-flop method, a power saving method based on the reduction of the internal activity of a FSM when it is not changing states, was discussed in Chapter 3. The conventional approach to this problem will be manually insert circuitry to stop the clock feeds into the inactive modules of a system, as was done in [Ben94]. However, the coverage and the complexity of those clock-stopping circuitry discourage the usage of this method.

In Chapter 3 of this thesis, the clock disabling toggle flip-flop was reviewed. It reduces the power consumption by gating the clock feeds into the individual flip-flops. Circuits designed with this clock-gating enabled flip-flop inherently save power without the requirement of external circuitry. It can save the power for not only inactive FSM, but also the inactive state variables of an active FSM. Since the clock-gating method has the potential of false clock edge generation and increasing clock skew, in Chapter 3 an intrinsic enabled-toggle-flip-flop (ETFF) was also described. It inherits the advantages of simplicity and efficiency of the clock-gating method and overcomes the problem of false clock edges without adding clock skew. A power saving of one third was observed.

6.2.2 The DLS Method

6.2.2.1 The Mop-flop

The DLS design method is a power saving method which greatly reduces the average number of transitions in an FSM. Since the CMOS circuits only dissipate measurable power during switching, the average switching activity of a CMOS circuit is a good measure of its average power dissipation. A sequential FSM is most active when it changes state. The average number of transitions per state depends on the state encoding. The state encoding which minimizes the transitions per state is the Gray code. By assigning every adjacent states with Gray code state coding, it is guaranteed that there are no races.

The Chapter 4 starts by examining conventional FSM designs. The structure of a *synchronous circuit* is such that every closed path contains a SHE (latch or flip-flop) controlled by the clock. Thus the circuit consists of combinational circuits alternating with SHEs. The method developed in Chapter 3 only focuses on optimizing the SHE to realize power saving when the FSM is not changing states. The state encoding method, which achieves the power saving from reducing the average number of transitions per state change, only takes active power consumption into consideration. By combining these two methodologies, one may achieve power saving from

both the combinational circuits and the SHEs, and achieve power saving from both the reduction of the number of transitions per state change and the reducing of the number of transitions when the FSM is not changing states. Based on above observations, a novel FSM design methodology, the DLS mop-flop design method, which can achieve both the above objectives, was developed and illustrated in Chapter 4.

In essence, all digital circuits can be viewed as being *asynchronous circuits* [Brz94] in which each component reacts to changes on its inputs as these changes arrive, and produces changes on its outputs when it concludes its computation. The *synchronous circuit* is merely a special case of *asynchronous circuit* in which one input signal is designated as a “clock”. Because of the using of special design rules, the *synchronous circuit*’s outputs may only change immediately after a certain clock event. In conventional *synchronous circuits*, the clock event is one of the rising or falling edges. Since these edges follow each other, in a conventional circuit half of the clock edges are wasted. Furthermore, the half wasted clock edges will excite the SHEs of that *synchronous circuit* and cause internal nodes switching, as a result, more power is wasted. The DLS mop-flop proposed in Chapter 4 avoids this problem by utilizing both levels of the clock signal as a “triggering” signal.

The DLS mop-flop utilizes the simplest SHE – a part-time feedback loop. By doing so, it requires the Gray code encoding of the state variables. Since there are no flip-flops in the circuit, the size of the circuit can be minimized. With proper design, the whole circuit size can be even smaller than the corresponding circuit’s flip-flops size. Some of the examples can also be found in the end of Chapter 4. The resulting circuits can be called multiple outputs flip-flops, and a saving/improvement of up to 50%-60% on size, speed, and power was found on some circuits. The drawback of not using flip-flops is that one has to make sure there are no races or hazards inside the mop-flop. Also the largest practical mop-flop circuit is about 32 states. Some more concerns of building DLS mop-flops are discussed in section 4.9.3.

6.2.2.2 The Mop-Flop Module Based Circuits

For large general FSM circuits, the proper smaller mop-flops can be designed first. Then these smaller mop-flops can be made into standard cells and form the bigger circuits. By doing so, circuits design can be more flexible since odd number of edges in a cycle is possible. The timing, race and hazards issues can also be dealt with inside the standard cells. As always, care has to be given to the synchronization and potential false clock edge issues inside the standard cells. Further research on designing mop-flop modules based circuits is required.

6.2.2.3 DET Circuit

The DLS technique can also work with flip-flops as its SHEs and resulting in DET circuits. One can think of this as replacing those Muxes in Figure 4.6 (b) by enabled-flip-flops, which were developed in Chapter 3 of this thesis, considerable circuit complexity reduction and power saving may be achieved. An N-bit cascadable Gray-code up/down counter circuit was designed, using the combined techniques, and illustrated in Chapter 5. The resulting circuit shows both the simplicity and power efficiency advantages over the best designs found by a comprehensive patent and literature search.

The major advantage of applying the proposed DET design methodology on designing *synchronous circuits* is power saving. However, throughout this thesis, the power dissipation discussed was only the power dissipated in the FSMs. In the real synchronous chip, there is another major source of power consumption: the power dissipated in the clock distribution network, which may account for 40% or more of the total power consumption of the chip [Dob92]. Since the power dissipated in clock distribution network is directly proportional to clock frequency, by using a DLS or DET design methodology, half of this power can be saved immediately.

There are some other advantages of applying the proposed DLS design methodology on designing synchronous circuits. According to the examples shown in Chapter 4 and 5, the circuits constructed by the proposed DLS design methodology often have a reduced device count and

shorter delay chains. This means the circuits are smaller and faster. The outputs of those circuits are changing in a Gray code manner, which results only one bit change at a time. This not only saves power on data buses, it also minimizes the on-chip noise caused by the output swings. As discussed in Chapter 5, the usage of Gray code outputs are also especially critical in the asynchronous to synchronous interfaces.

It is worth pointing out that the enabled flip-flop design method and the DLS mop-flop design method described in Chapter 3 and Chapter 4 can be directly used under standard synthesis tools, such as SYNOPSIS, which makes the design based on these new techniques extremely easy.

6.3 Future Work

There is still work that need to be done on the proposed DLS design methodology. When designing a mop-flop with consecutive multi-way branches, the number of states that must be added may be large.

Circuits using mop-flop modules can overcome the above problem, but races and hazards issues have to be carefully dealt with inside the mop-flop modules. As always, care has to be given to the synchronization issues. Further research on designing mop-flop modules based circuits is required.

More works also need to be done on applying the DET method on general applications as well as analyzing the performance of the large mop-flop type circuits.

References

- [Ben94] Luca Benini, Polly Siegel, and Giovanni De Micheli, "Saving Power by Synthesizing Gated Clocks for Sequential Circuits," IEEE Design & Test of Computers, Winter 1994, pp. 32-41.
- [Brz94] Janusz A. Brzozowski, Carl-Johan H. Seger, "Asynchronous circuits," Springer-Verlag, New York, 1994, TK7868A79B79.
- [Cha92] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen, "Low-Power CMOS Digital Design," IEEE Journal of Solid-State Circuits, Vol. 27, NO. 4, April 1992, pp. 473-484.
- [Dob92] D. Dobberpuhl, et al. "A 200-MHz 64-b Dual-Issue CMOS Microprocessor," IEEE Journal of Solid-State Circuits, Vol. 27, no. 11, pp.1555-1566, Nov. 1992.
- [For92] Navid Foroudi, "A High-Speed CMOS Dual-Modulus Frequency Divider For Mobile Radio Frequency Synthesizers," M.Eng. Thesis, Carleton University, 1992.
- [Hac94] Gary D. Hachtel, Mariano Hermida, Abelardo Pardo, Massimo Poncino, and Fabio Somenzi, "Re-Encoding Sequential Circuits to Reduce Power Dissipation," ICCAD'94, San Jose, CA.
- [Hay93] John P. Hayes, "Introduction To Digital Logic Design," Addison Wesley, 15461, 1993, pp. 519-528.
- [Hir98] Hiroshi Kawaguchi and Takayasu Sakurai, "A Reduced Clock-Swing Flip-Flop (RCSFF) for 63% Power Reduction," IEEE Journal of Solid-State Circuits, VOL. 33, NO. 5, May 1998, pp. 807-811.
- [Joh97] Michael John Sebastian Smith, "Application-Specific Integrated Circuits," Addison-Wesley, 1997.
- [Liu93] Dake Liu and Christer Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltage," IEEE Journal of Solid-State Circuits, Vol. 28, NO. 1, January 1993, pp. 10-17.
- [Rab96] Jan M. Rabaey, "Digital Integrated Circuits," Prentice Hall, 1996.
- [Sat94] Katsuyuki Sato, Akishima, Miki Matsumoto et al., US patent 5,313,423. May 17, 1994.
- [Su94] Su, Ching-Long et al., "Saving Power in the Control Path of Embedded Processors," IEEE

Design & Test of Computers, vol. 11, No. 4, Winter 1994.

[Sne96] Martin Snelgrove, "VLSI circuits," unpublished, Carleton University, 1996.

[TI85] "The TTL Data Book", Volumn 2, type SN5475, SN5474, SN54L75, SN54L77, SN54LS75, SN54LS77, SN7475, SN74LS75.

[Ung69] S.H. Unger, "Asynchronous Sequential Switching Circuits," New York, Wiley-Interscience, 1969.

US patent 4,528,665: Burns, Gary D., Mackenthun, Donald W., Schaber, Scott D., "Gray Code Counter With Error Detector In A Memory System," July 1985.

US patent 4,654,851: Busby, W. Ray, "Multiple Data Path Simulator," Mar. 1987.

US patent 4,780,894: Watkins, Daniel, Wong, Jimmy, "N-bit Gray Code Counter," Apr. 1987.

US patent 4,949,341: Lopez, David E., Colunga, Tomas, "Built-in Self Test Method For Application Specific Integrated Circuit Libraries," Aug. 1990.

US patent 5,191,425: Hachiyama Hiroki, Iwasawa, Takahiro, Ohmae, Masanori, Yoshida, "Image Pickup Apparatus Having A Gray Counter," Mar. 1993.

US patent 5,313,423: Sato, Katsuyuki, Matsumoto, Miki, et al. "Semiconductor Memory Device," May 1994.

US patent 5,355,396: Tai, "Circuitry And Method For Modularized Single Transition Counting," Oct. 1994.

US patent 5,426,756: Shyi, Jonathan, Shen, Kenny, "Memory Controller And Method Determining Empty/Full Status Of A FIFO Memory Using Gray Code Counters," June 1995.

US patent 5,428,654: Imran Baqai, "Up/Down Counter Apparatus," Jun. 1995.

US patent 5,448,606: Snelgrove, Andrew H., "Gray Code Counter," Jun. 1993.

US patent 5,555,524: Castellano, Andrew J., "Semi-Synchronous Dual Port FIFO," Sep. 1996.

Appendix A

A.0 The HSpice files for some complex gates

A.0.1 AOA

```
*AOA: F=not(ab+cd)
*complex gate
.SUBCKT AOA a b c d out
MP1 out c 1 VCC MPCH-0P8 L=0.8U W=6U
MP2 out d 1 VCC MPCH-0P8 L=0.8U W=6U
MP3 1 a VCC VCC MPCH-0P8 L=0.8U W=6U
MP4 1 b VCC VCC MPCH-0P8 L=0.8U W=6U
MN5 out a 2 0 MNCH-0P8 L=0.8U W=3U
MN6 2 b 0 0 MNCH-0P8 L=0.8U W=3U
MN7 out c 3 0 MNCH-0P8 L=0.8U W=3U
MN8 3 d 0 0 MNCH-0P8 L=0.8U W=3U
.ENDS
```

A.0.1 OA

```
*OA: F=not((a+b)c)
*complex gate
.SUBCKT OA a b c out
MN1-A1 out a 2 0 MNCH-0P8 L=0.8U W=3U
MN1-A2 out b 2 0 MNCH-0P8 L=0.8U W=3U
MN1-A3 2 c 0 0 MNCH-0P8 L=0.8U W=3U
MN1-A4 1 a VCC VCC MPCH-0P8 L=0.8U W=6U
MN1-A5 out c VCC VCC MPCH-0P8 L=0.8U W=6U
MN1-A6 out b 1 VCC MPCH-0P8 L=0.8U W=6U
.ENDS
```

A.1 The HSpice files for the circuits in Chapter 3

A.1.1 The DFF circuit

```
*
*6 states D Flip-flop
*
.SUBCKT DFF CLK D Q NOTQ
XDFF4-A1 1 3 4 NAND
XDFF4-A2 4 CLK 3 NAND
XDFF4-A3 3 CLK 1 2 NAND3
XDFF4-A4 2 D 1 NAND
XDFF4-A5 3 NOTQ Q NAND
XDFF4-A6 Q 2 NOTQ NAND
.ENDS
```

A.1.2 The GFF circuit

```
*
```

Appendix

*The gated 6 states D flip-flop

*

```
.SUBCKT GFF CLK T Q NOTQ
XDFF4-A0 CLK T GCLK OR
*XDFF4-A0 CLK T GCLK AND
XDFF4-A1 1 3 4 NAND
XDFF4-A2 4 GCLK 3 NAND
XDFF4-A3 3 GCLK 1 2 NAND3
XDFF4-A4 2 NOTQ 1 NAND
XDFF4-A5 3 NOTQ Q NAND
XDFF4-A6 Q 2 NOTQ NAND
.ENDS
```

A.1.3 The ETFF circuit

*

*The enabled T flip-flop circuit

*

```
.SUBCKT ETFF c e q1
X1 e notc inv
X2 q1 notq1 inv
X3 q2 notq2 inv
X4 c notc 4 NOR
X5 c e 5 NAND
X6 notc q1 4 notq1 6 AOA
X7 5 not5 inv
X8 notq1 5 notq2 not5 q1 AOA
X10 5 notq2 6 q2 OA
.ENDS
```

A.2 The HSpice files for the circuits in Chapter 4

A2.1 The double-level-sensitive two-bit Gray code counter

*

*The two-bit Gray code counter

*

```
.SUBCKT gray2 clk q1 q2
XGRAY2-1 q2 clk 1 OR
XGRAY2-3 clk notc INV
XGRAY2-2 q2 notc 2 AND
XGRAY2-4 q1 1 3 AND
XGRAY2-5 3 2 q1 OR
XGRAY2-6 q1 notq1 INV
XGRAY2-7 1 notq1 4 AND
XGRAY2-8 4 2 q2 OR
.ENDS
*
*gray2_cpx: gray2 counter made with complex gates
*
.SUBCKT gray2_cpx c q1 q2
X1 c notc inv
```

Appendix

```
X2 q1 notq1 inv
X3 q1 c q2 notc notq2 AOA
X4 q2 notc notq1 c q1 AOA
X5 notq2 q2 inv
.ENDS
```

A.2.2 The double-level-sensitive three-bit Gray code counter

```
*
*Three-bit Gray code counter
*
.SUBCKT gray3 c q1 q2 q3
X1 c notc INV
X2 q3 notq3 INV
X3 q2 notq2 INV
X4 q3 notc 1 NOR
X5 notq3 notc 2 NOR
X6 q1 q2 3 AND
X7 q1 q2 4 NOR
X8 3 4 5 NOR
X9 notq2 1 6 AND
X10 q1 1 7 NOR
X11 q1 2 8 AND
X12 q2 2 9 NOR
X13 notq3 c 10 AND
X14 5 notc 11 AND
X15 10 11 q3 NOR
X16 8 9 q2 NOR
X17 6 7 q1 NOR
.ENDS
```

A.2.3 The double-level-sensitive four-bit Gray code counter

```
*
*Four-bit Gray code counter
*
.SUBCKT gray4 c q1 q2 q3 q4
X1 q4 notq4 INV
X2 c notc INV
X3 q3 notq3 INV
X4 q2 notq2 INV
X5 c notq4 31 NAND
X6 q1 q2 32 AND
X7 q1 q2 33 NOR
X8 32 33 34 NOR
X9 notc notq4 39 NOR
X10 notq3 31 38 NOR
X11 34 q3 35 AND
X12 34 q3 36 NOR
X13 35 36 37 NOR
X14 q3 31 48 NOR
X15 34 39 40 AND
X16 q3 39 41 NOR
```

Appendix

```
X17 40 41 q3 NOR
X18 q1 38 42 AND
X19 q2 38 43 NOR
X20 42 43 q2 NOR
X24 notq2 48 46 AND
X25 q1 48 47 NOR
X26 46 47 q1 NOR
.ENDS
```

A.2.4 The double-level-sensitive traffic light controller circuit

```
*
*schematic description of the traffic light controller
*
X1 q1 x 1 NOR
X2 c q2 2 AND
X3 c q2 3 NOR
X4 2 3 4 NOR
X5 c 1 5 AND
X6 q2 1 6 NOR
X7 1 4 7 OR
X8 x q1 8 NAND
X9 5 6 q2 NOR
X10 7 8 q1 NAND
```

A.2.5 The double-level-sensitive game controller circuit

```
*
*schematic description of the game controller
*
X1 c notc inv
X2 q2 notq2 inv
X3 q3 notq3 inv
X4 p notp inv
X5 q1 q2 5 NOR
X6 5 notc 28 6 AND3
X7 5 notc 7 NOR
X8 6 7 8 NOR
X9 notq2 notq3 q2 q3 11 AOA
X12 q1 q3 12 NOR
X13 q2 notp 13 AND
X14 13 8 14 NOR
X15 q1 q2 15 NAND
X16 c 11 not28 16 OR3
X17 16 q1 17 NAND
X18 p q2 notq3 notc 18 NAND4
X19 p q1 notc notq3 19 NOR4
X20 R p 20 OR
X21 20 c 12 21 NAND3
X22 14 notq3 notc 15 q3 OAO
X25 17 18 q1 NAND
X26 19 notq2 21 q2 OA
*X28 is added to gate 8 just to mask it when q1q2q3=000, since
```

Appendix

*next state only q2 will change, gate 8 only affect q3

X28 q1 q2 q3 28 OR3

X29 28 not28 inv

A.2.6 A 4-bit Gray code up/down counter module

*

*The schmatic description of a 4-bit Gray code up/down counter module

*

.SUBCKT async_gray_4 c updown q q1 q2 q3 q4

*inputs are c(clk) and updown signals

*outputs are 4 gray code bits and q is used as clock signal for next block

X1 q1 notq1 inv

X2 e note inv

X3 q2 notq2 inv

X4 13 not13 inv

X5 c notc inv

X6 17 not17 inv

X7 q3 notq3 inv

X8 q4 notq4 inv

X9 15 not15 inv

X10 e notq1 10 NAND

X11 q1 e notq1 note 11 AOA

X12 notq2 notq1 q2 q1 12 AOA

X13 10 notq2 11 q2 13 AOA

X14 q1 notq2 14 NAND

X15 notq2 q3 15 NOR

X16 12 q3 14 notq3 16 AOA

X17 c notq4 17 NAND

X18 15 notq1 18 AND

X19 18 16 19 NOR

X20 11 notq3 20 NOR

X21 15 note 21 NAND

X22 q3 17 22 OR

X23 22 q1 23 NAND

X24 14 e 24 OR

X25 24 not13 25 NAND

X26 notq4 notc 26 NOR

X27 note 19 27 AND

X28 note 19 28 NOR

X29 27 28 29 NOR

X30 20 not17 q2 17 30 AOA

X31 17 21 31 OR

X32 31 14 23 q1 NAND3

X33 25 26 33 AND

X34 q3 26 34 NOR

X35 33 34 q3 NOR

X37 29 notc 37 AND

X38 notq4 c 38 AND

X39 37 38 q4 NOR

X36 30 not15 q2 NAND

*the circuit to generate the e signal

*when q2q3q4 are 000, enable flipflop

Xe1 q2 q3 q4 enable NOR3

Appendix

```
Xe2 enable q d XOR
Xe3 c d q notq type4
Xe4 q updown e XOR
.ENDS
```

A.3 The HSpice files for the circuits in Chapter 5

A.3.1 The 2_bit module

```
*
* The schematic description of the 2_bit_count_to_8 circuit
*
X1 q1 q2 db XOR
X2 c notc inv
X3 NOTPR NOTCLR c NOTQ2 Q2 NOTQ2 yidff
X4 NOTPR NOTCLR notc db Q1 NOTQ1 yidff
.SUBCKT YIDFF NOTPR NOTCLR CLK D Q NOTQ
*NOTPR AND NOTCLR ARE RESET SIGNALS, SET TO "1" FOR NORMAL OPERATION
*CLK AND D ARE INPUT SIGNAL
*Q AND NOTQ ARE OUTPUT SIGNAL
XDFF-A1 NOTPR 1 3 4 NAND3
XDFF-A2 4 NOTCLR CLK 3 NAND3
XDFF-A3 3 CLK 1 2 NAND3
XDFF-A4 2 NOTCLR D 1 NAND3
XDFF-A5 NOTPR 3 NOTQ Q NAND3
XDFF-A6 Q 2 NOTCLR NOTQ NAND3
.ENDS
```

A.3.2 The 3_bit module

```
*
* The schematic description of the 3_bit_count_to_16 circuit
*
X1 q1 q2 db XOR
X2 c notc inv
X3 NOTPR NOTCLR c NOTQ2 Q2 NOTQ2 yidff
X4 NOTPR NOTCLR notc db Q1 NOTQ1 yidff
*extra Toggle flip-flop
X10 q2 notq1 c 8 NOR3
X11 NOTPR NOTCLR 8 notq3 q3 NOTQ3 yidff
```

Appendix B

B.1 The double-level-sensitive method based eight-bit Gray code counter

B.2 Conventional Gray code counters

(note: the outputs should be taken directly from the outputs of the flip-flops)

B.2.1 The 4-bit Gray code counter

B.2.2 The 3-bit Gray code counter

B.2.3 The 2-bit Gray code counter

B.2.4 The conventional master controller circuit

Appendix C

C.1 The State Graph Of The 6-state D Flip-flop

All conditions labelled on arrows are inputs.
sr are state variables of the first stage and
also the outputs to the second stage

First Stage

Second Stage

C.2 The Tug-of-War Game

The players of this game see a row of 5 LEDs represented by the numbers -2 through 2:

After pushing reset, all the LEDs should come on for a second and then go off. This is the get ready signal. After a random time the LED comes on again. Then each player will try to push his button before the other player does. The position of the lit LED will move toward the fastest button pusher.

Thus if player 2 is fastest, LED 0 will go out and LED 1 will come on. This is the end of **round** one. After a second the LEDs will all go out. This is the get ready signal for round two. After a random time LED 1 will come on (assuming player 2 won the first round). Again each player will try to push his button first, and again the light will shift toward the fastest button pusher. The game is won when one player makes the position of the lit LED move off the end of the display.

If a player jumps the light, i.e. he pushes his button while the LED is off, then the light will come on immediately and it will be shifted one position away from him.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved