Low Power Design Techniques for Digital Logic Circuits

By

Yinshui Xia

BSc, MSc

© Copyright by Yinshui Xia 2003

A thesis presented in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy

Napier University

School of Engineering

March 2003

Declaration

I declare that no portion of the work referred in this thesis has been submitted in support of an application of another degree, qualification or other academic awards of this or any other university or institution of learning.

Yinshui Xia

Acknowledgements

The research presented in this thesis was carried out under the supervision of Professor A. E. A. Almaini, School of Engineering, Napier University. I sincerely thank him for offering me an opportunity to study at Napier University, guiding my research to the logic synthesis world, and encouraging me during this research. I am very grateful to Professor Almaini for his regular comments and invaluable suggestions. I do appreciate his patience to teach the scientific writing in English and correct my research papers and this thesis.

Thanks are due to Professor J. Hajto, my second supervisor, for his essential encouragement of this research.

I would like to acknowledge the financial support of the School of Engineering, Napier University, Edinburgh. Without this support, this research work would not have been possible.

I would like to thank Dr Lingli Wang of Altera, a former member of the digital techniques group, for his kindly support to install Linux operating system and great help to learn Linux operating system and GNU C language programming environment when I started this research. I also would like to thank Dr Alex Bystrov, also a former member of the group, at Newcastle University for the valuable discussion of my research.

Thanks are due to other members of the digital techniques group, especially Mr Belgasem Ali, Mr Khalid Faraj and Mr Meng Yang, for their enjoyable working environment and various discussions. Thanks are also due to my former supervisors and research partners in the different period in my life. They are Prof. X. Wu of Ningbo University, Prof. T. G. Clarkson of King's College London, Dr Nan Zhuang of Synopsys.

The time I have spent in the School of Engineering of the Napier University has been the most challenging of my life. I would like to thank all staffs and PhD students in the School for their kind helps. Finally, the special thanks are due to my wife Qiufen and my daughter Kankan for their understanding and support during my research study.

Abstract

With the rapid increase in the density and the size of chips and systems, area and power dissipation become critical concern in Very Large Scale Integrated (VLSI) circuit design. Low power design techniques are essential for today's VLSI industry. The history of symbolic logic and some typical techniques for finite state machine (FSM) logic synthesis are reviewed.

The state assignment is used to optimize area and power dissipation for FSMs. Two cost functions, targeting area and power, are presented. The Genetic Algorithm (GA) is used to search for a good state assignment to minimize the cost functions. The algorithm has been implemented in C. The program can produce better results than NOVA, which is integrated into SIS by UC Berkeley, and other publications both in area and power tested by MCNC benchmarks.

Flip-flops are the core components of FSMs. The reduction of power dissipation from flip-flops can save power for digital systems significantly. Three new kinds of flip-flops, called differential CMOS single edge-triggered flip-flop with clock gating, double edge-triggered and multiple valued flip-flops employing multiple valued clocks, are proposed. All circuits are simulated using PSpice.

Most researchers have focused on developing low-power techniques in AND/OR or NAND & NOR based circuits. The low power techniques for AND/XOR based circuits are still in their early stage of development. To implement a complex function involving many inputs, a form of decomposition into smaller subfunctions is required such that the subfunctions fit into the primitive elements to be used in the implementation. Best polarity based XOR gate decomposition technique has been developed, which targets low power using Huffman algorithm. Compared to the published results, the proposed method shows considerable improvement in power dissipation. Further, Boolean functions can be expressed by Fixed Polarity Reed-Muller (FPRM) forms. Based on polarity transformation, an algorithm is developed and implemented in C language which can find the best polarity for power and area optimization. Benchmark examples of up to 21 inputs run on a personal computer are given.

Contents

List of Abbreviations

ADD Algebraic Decision Diagram

AND AND operation

BAL-CS Basic And Logic Combining Section

BCP Binary Clock Pulse

BDD Binary Decision Diagram

CAD Computer-Aided Design

CCB Clock Chain Based

CG Clock Gating

CK Clock Signal

CMOS Complementary Metal-Oxide-Semiconductor

C^2MOS Clocked CMOS

CP Clock Pulse

DC Direct Current

DET Double Edge-Triggered

DETFF Double Edge-Triggered Flip-Flop

EPT Even Polarity Transformation

FF Flip-Flop

FSM Finite State Machine

FPGA Filed Programmable Gate Array

FPRM Fixed Polarity Reed-Muller

GA Genetic Algorithm

HDL Hardware Description Language

IC Integrated Circuit

ITV Input Threshold Voltage

LCS Logic Combining Section

LID Location InDicator

LSB Less Significant Bit

MCNC Microelectronics Center North Carolina

MSB Most Significant Bit

MV Multiple Valued

MVCP Multiple Valued Clock Pulse

MVFF Multiple Valued Flip-Flop

NLS Negative-Level-Sampling

NMOS N-type Metal-Oxide-Semiconductor

NP Narrow Pulse

OBDD Ordered Boolean Decision Diagram

OPT Odd Polarity Transformation

PDA Personal Digital Assistant

PDP Power Delay Product

PLA Programmable Logic Array

PLS Positive-Level-Sampling

PMOS P-type Metal-Oxide-Semiconductor

PPRM Positive Polarity Reed-Muller

PTB Pass Transistor Based

RTL Register Transfer Level

SA Switching Activity

SC Sampling Circuit

SET-FF Single Edge-Triggered Flip-Flop

SOP Sum Of Products

SPICE Simulation Program with Integrated Circuit Emphasis

STG State Transition Graph

STT State Transition Table

TSPC True Single Phase Clocking

VHDL Very large scale Hardware Description Language

VLSI Very Large Scale Integration

XOR EXclusive OR operation

List of Figures

List of Tables

Chapter 1

Introduction

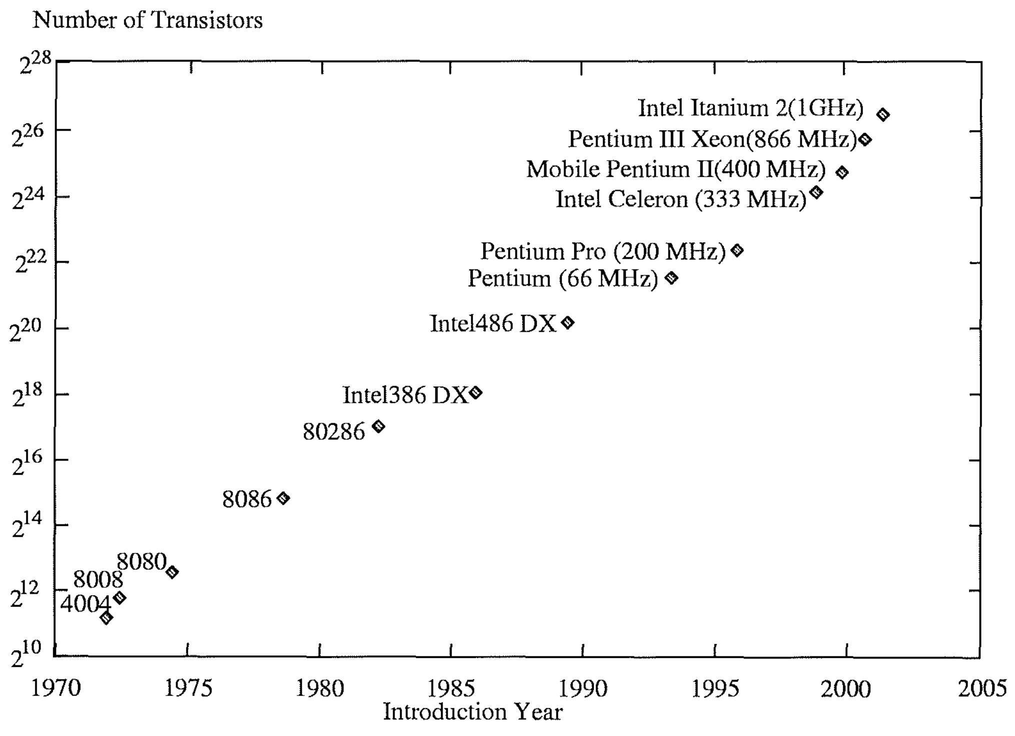
1.1 Motivations of low power design

The genesis of low power microelectronics can be traced to the invention of the transistor in 1947. It was a breakthrough of virtually unparalleled importance in electronics to eliminate the crushing needs for several watts of heater power and several hundred volts of anode voltage in vacuum tubes in exchange for transistor operation in the tens of milliwatts range. The capability to fully utilize the low power assets of the transistor was provided by the invention of the integrated circuit in 1958. Since the first integrated circuit (IC) was developed by Jack Kilby in Texas Instruments and then became the first commercial IC by Fairchild Instruments in 1961, the IC technology has progressed greatly. The size and density of IC chips and systems are increasing as Gordon Moore, co-founder of Intel, predicted in 1960s that the number of transistors on an IC chip could be doubled every 12 to 18 months. This is well confirmed by the development trace of Intel microprocessors as shown in Fig. 1.1.

The continuing increase in chip density and operating frequency have made power consumption a major concern in very large scale integrated (VLSI) circuit design. For example, the PC chip from Motorola consumes 8.5W, the Pentium chip from Intel consumes 16W, and DEC's 21164 (300MHz on a die area of $3cm^2$) consumes 50W! It is extrapolated that $10cm^2$ microprocessor, clocked at 500MHz would consume 315 W in near future [83]. Unless power consumption is dramatically reduced the resulting heat will limit the feasible packing and performance of VLSI circuits and systems.

Perhaps the primary driving factor for designing low power systems has been the remarkable success and growth of the class of portable personal computing devices and wireless communication systems which demand complex functionality and high speed communication. Probably no segment of the electronics industry has a growth potential as explosive as that of the personal digital assistant (PDA) which has been characterized as a combined pocket cellular phone, pager, e-mail terminal, fax, computer, calendar, address directory, notebook, etc. [96][69][26].

The market of portable applications is growing very rapidly. Fig. 1.2 shows the various PC percentages of the PC market in 1992 and 1998 [60]. In portable applications, average power consumption is a critical design constraint since it is related to the battery life time, size and weight [55][66]. The reason for this is illustrated with the simple example of a multi-media terminal. The projected power for such a terminal, when implemented using off-the-shelf components not designed for low-power operation [95], is about 40W. With advanced Nickel-Metal-Hydride battery technologies yielding around 65 watt-hours/kilogram [56], this terminal would require an unacceptable 6 kilograms of batteries for 10 hours of operation between recharges. Even with new



* These data are originally from ***"Intel Microprocessor Quick Reference Guide"***

Figure 1.1: Moore Law

battery technologies, such as rechargeable Lithium Ion or Lithium Polymer cells, it is anticipated that the expected battery lifetime will increase to about 90-110 watt-hours/kilogram, which still leads to an unacceptable 3.6 to 4.4 kilograms of battery cells. Fig. 1.3 shows the battery capacity increasing over the last 30 years [56]. From Fig. 1.3, it can be observed that battery capacity has only improved with a factor 2 to 4 over the last 30 years while the computation power of digital IC's has increased by more than 4 orders of magnitude [51]. The gap is increasing with respect to power demand. In the absence of low power design techniques, the current and future portable devices will suffer from either very short battery life or unreasonably heavy battery pack.

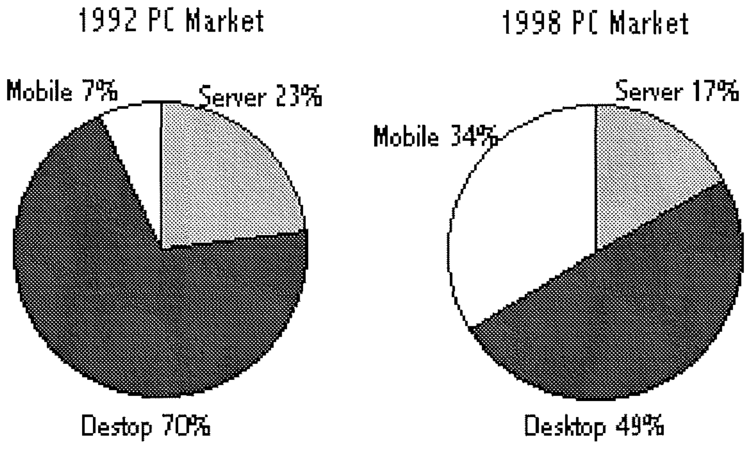


Figure 1.2: Market of portable applications is growing very rapidly

These problems make it necessary to develop power aware VLSI design tools that help achieve low power in these systems. Indeed, the Semiconductor Industry Association has identified low-power design techniques as a necessary technological need [93].

1.2 VLSI chip design approaches for low power

1.2.1 Sources of power dissipation

Since CMOS technology is predominant in the realization of today's IC and CMOS devices are intrinsically low power consuming, all circuits for the rest of this thesis refer to CMOS circuits. It is judicious to briefly discuss the mechanisms for power consumption in CMOS circuits. Take the inverter in Fig. 1.4 for example. The power dissipation for the inverter consists of three major sources which are expressed in the following equation:

$$P_{total} = P_{leakage} + P_{short\_current} + P_{switching} \tag{1.1}$$

The first term, $P_{leakage}$, can arise from substrate injection and subthreshold effects and is primarily determined by fabrication technology considerations [24]. The second term, $P_{short\_current}$, is due to the direct-path short circuit current, which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground [118]. The third term, $P_{switching}$, represents the switching power dissipation or dynamic power dissipation. This is the result of capacitance charging and discharging in the circuit. The situation is modeled in Fig.1.4 where the parasitic capacitances are lumped at the output in the capacitor C_L. Consider the behavior of the circuit over one full cycle of operation with the input voltage going from V_{dd} to ground and back to V_{dd}. When the input changes from V_{dd} to ground, the

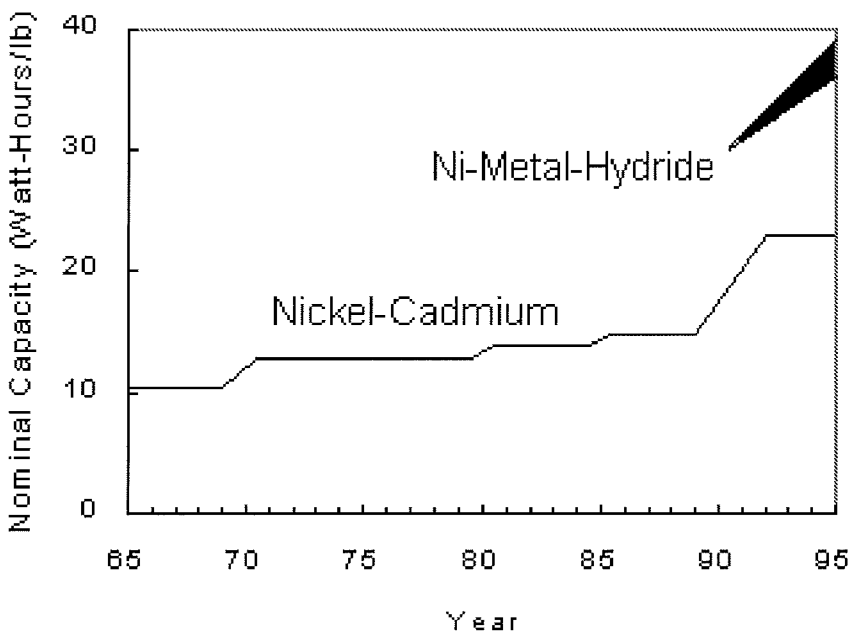


Figure 1.3: Battery capacity during the past 30 years

capacitor C_L is charged. This charging process draws an energy equal to $C_L V_{dd}^2$ from the power supply and $\frac{1}{2} C_L V_{dd}^2$ is stored in the capacitor C_L because at the end of the transition the output capacitor C_L is charged to V_{dd}. Hence, $\frac{1}{2} C_L V_{dd}^2$ is dissipated in the PMOS network. When the input changes from ground to V_{dd}, the capacitor C_L is discharged. And $\frac{1}{2} C_L V_{dd}^2$ is dissipated in the NMOS network. Hence, for a full cycle, $C_L V_{dd}^2$ from the power supply is consumed. This leads to the conclusion that CMOS power consumption depends on the switching activity of the signals involved. If α represents the signal activity, the expected number of zero to one transitions per data cycle, and f is the average data-rate, which may be the clock frequency in a synchronous system, then the effective frequency of nodal charging is given by the product of the activity and the data rate: αf. Hence, the average CMOS power consumption is given by equation 1.2.

$$P_{switching} = \frac{1}{2} \alpha C_L V_{dd}^2 f \tag{1.2}$$

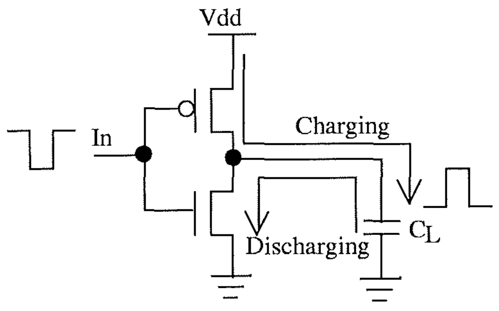


Figure 1.4: Charging and discharging for an inverter

For a well designed circuit, the first two terms in equation 1.1 can be kept below 20% of the total power [14]. Hence, in CMOS circuits, $P_{switching}$ is by far the most important. From equation 1.2, $P_{switching}$ is proportional to the switching activity, capacitance loading, data-rate (in synchronous systems f might correspond to the clock frequency), and the square of the supply voltage.

1.2.2 Design approaches of low power VLSI systems

From the above analysis of power dissipation sources, the vast majority of power reduction techniques concentrate on minimizing the dynamic power dissipation by reducing one or more factors on the right hand side of equation 1.2.

- Voltage scaling

One of the most obvious ways to reduce power is to reduce the power supply voltage of circuits because the dynamic power is proportional to the square of the supply voltage. Power savings are relatively independent of circuit function and circuit technology. Hence, it is applicable at different stages of the design development. Some methodologies have been proposed [24][25]. However, with the scaling of supply voltage and device dimensions, the transistor threshold voltage also has to be scaled to achieve the required performance. Unfortunately, such scaling does not come for free and can increase the leakage current. The leakage current occurs due to carrier diffusion between the source and the drain when the gate-source voltage, V_{gs}, has exceeded the weak inversion point, but is still below the threshold voltage V_{th}, where carrier drift is dominant. The current in the subthreshold region is given by equation 1.3[24]

$$I_{sub} = kexp(\frac{V_{gs} - V_{th}}{S/ln10})(1 - exp(-\frac{V_{ds}}{V_T})) \tag{1.3}$$

Where k is a function of the technology, V_T is the thermal voltage (KT/q), V_{th} is the threshold voltage and S is the subthreshold swing.

Due to the exponential nature of subthreshold leakage current with V_{th}, subthreshold current can no longer be ignored. The lower the V_{th} is, the higher the subthreshold leakage current will be. On the other hand, delay through a logic block is proportional to $V_{dd}/(V_{dd} - V_{th})^2$ where V_{dd} is the voltage of the power supply [24][25]. Lower V_{th} means longer delay. To compromise the two, it is suggested to use high supply voltage in the critical paths of a design to achieve the required performance while the off-critical paths of the design use lower supply voltage to achieve low-power dissipation [53][28]. It is also clear from the above discussion that reducing the threshold voltage allows the supply voltage to be scaled down to lower $P_{switching}$ without loss in speed. However, the limit on threshold voltage scaling is imposed by the noise margin and the increase of sub-threshold current. Scaling down threshold voltage trades off between dynamic power ($P_{switching}$) and static power ($P_{Leakage}$). In considering logic and memory circuit behavior, $V_{dd} = 1.0V$ appears to be a good compromise for small dynamic and static power dissipation [89]. A genius approach is architecture-driven supply voltage scaling based strategy which is to modify the architecture of the system so as to make it faster and reduce V_{dd} so as to restore the original speed resulting in reduced power consumption [24][25]. However, area overhead is required.

- Frequency reduction

The second obvious way to reduce power is to decrease the clock frequency f. Decreasing f causes a proportional decrease in power dissipation. However, slowing the clock will result in a slower computation. The power consumption over a given period of time is reduced, but the total amount of useful work is reduced as well. As a result, the energy dissipated to complete the task has not changed. This can be illustrated by an example. Suppose the system is clocked with a clock period T_1, and the task takes N clock cycles to complete. During each cycle, the system dissipates an average power P_1. If the frequency is decreased by half, the power dissipation over the original time period will be $P_2 = \frac{1}{2}P_1$ because average power is directly proportional to the clock frequency. However, it now takes a total time of $2NT_1$ to complete the task. As a consequence, the average energy consumed by the system is $E = P_1NT_1$ in both cases. However, this observation still incites some genius low power designs. One of them is double edge-triggered flip-flops targeting low power [49][19][68][102], which enables a halving of the clock frequency and hence reduces power dissipation on the clock line for a given data rate compared with the single edge-triggered flip-flops.

- Effective capacitance reduction

Effective capacitance, C_{Eff}, is defined as $C_{Eff} = \alpha C_L$. Design and synthesis techniques have been developed to reduce both the capacitive load, C_L, and the switching activity, α, at all stages of the design process[37][45][65][27][61][78][88][108][131]. It is obvious that once a technology and a supply voltage have been set, power savings come from the careful minimization of the effective capacitance. There are many effective applications of this idea. Some examples are shown as follows.

1. Dynamic power management: The sleep modes of operation in portable computers [37][45] are examples of this approach to reduce useless switching activity. Power is reduced by stopping the clock or shutting down the power supply of parts of the system that are not required to carry out the current task.

2. Algorithmic transformations for signal processing tasks [65][27]: Reducing the number of operations needed to carry out a given computation may not be always useful in terms of performance, but it is often useful for reducing power.

3. Communication protocol design [61]: Communication protocols can be modified to improve the activity patterns.

4. Memory allocation [78] and bus encoding [98] techniques to minimize the power dissipated in memories and system busses.

5. Logic optimization [88][108][114]: Area minimization can obtain global power savings because area is directly related to the capacitive load. Logic optimization can directly target power optimization.

6. Circuit topology [131][74]: This method aims to change circuit topology to minimize the switching activity of a circuit so that the effective capacitance is minimized.

- Other approaches

A revolutionary approach for low power design is called energy recovery/recycle techniques, which are addressed in [11][34]. Scaling down feature size is an important issue for high-performance and high-density VLSI circuits. However, some second order effects become serious and are becoming a major challenge in deep submicrometer devices and circuits. Corresponding low power techniques are being explored [52].

1.3 Low power design flow

Designing for low power is at least as difficult as designing for maximum speed or minimum area. Power dissipation is a pattern-dependent cost function, unlike area, which is constant with respect to input patterns. Since power dissipation becomes increasingly important as a design evaluation metric, a new generation of computer-aided design tools targeting power minimization is urgently needed by designers. In the last few years, significant research and development efforts have been undertaken in academia and industry targeting the creation of a new generation of computer-aided design (CAD) tools for low power. As a result, hundreds of papers have been published on the subject [88][104][12][108][77]. The wide range of ECAD tools for low power fall into four major categories based on the four levels, which are behavioral, architectural (Register Transfer/RT), logic and circuit level [89]. Countless commercial and academic design synthesis tools are available and some of them are shown in Table 1.1. It can be seen that VHDL [81] and Verilog [103] are the most popular hardware description languages (HDLs).

Table 1.1: Some commercial and academic low power design tools

Organization	System	Description	Input	Environment
Synopsys	PowerMill	Circuit Simulation	SPICE, Verilog	Unix
Synopsys	Power Compiler	RTL Power Estimation	Gate Netlist, Switching Activity, Constraints	Unix
Synopsys	PowerHogs	RTL Power Optimization	Verilog/VHDL	Unix
Cadence	PowerSim	Gate and Logic Level Estimation	Verilog	Unix
TransEDA	PowerSure	HDL/RTL Power Estimation	Verilog/VHDL	Unix
Mentor	Lsim Power Analyst	Circuit level Simulation	SPICE	Workstation
Veritools	Power-Tool	Logic level Power Estimation	Verilog	Win95, WinNT, Sun4.x
Berkeley University	SIS	Logic Level, RTL Power Estimation	Gate Netlist	Win95, Unix

A typical synthesis based VLSI low power design flow is shown in Fig. 1.5[60].

It can be seen from Fig. 1.5 that three steps are included in low power synthesis at each design level:

1. Estimate power according to unoptimized description. The methods of power estimation are different from level to level. This process is called power analysis.

2. Optimize the description through various available procedures by the criteria of power dissipation, area, speed or testability. This important process is called optimization.

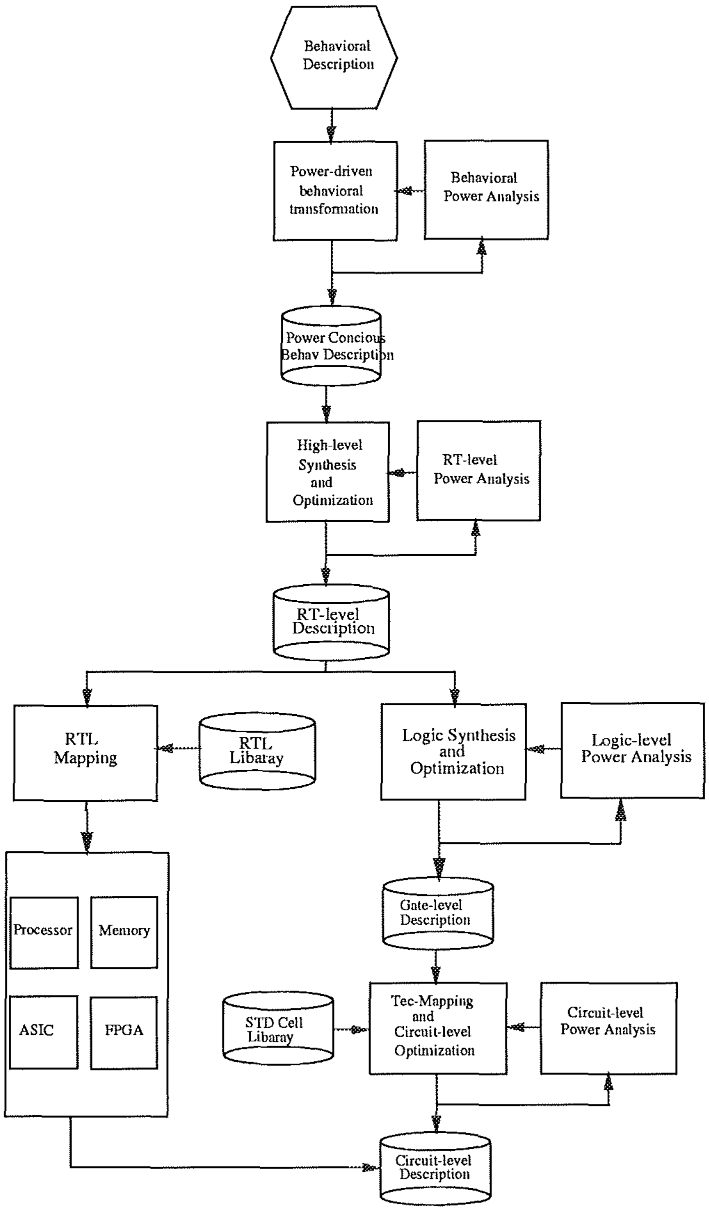

Behavioral Description
Power-driven behavioral transformation
Behavioral Power Analysis
Power Concious Behav Description
High-level Synthesis and Optimization
RT-level Power Analysis
RT-level Description
RTL Mapping
RTL Libaray
Logic Synthesis and Optimization
Logic-level Power Analysis
Processor
Memory
ASIC
FPGA
Gate-level Description
STD Cell Libaray
Tec-Mapping and Circuit-level Optimization
Circuit-level Power Analysis
Circuit-level Description

Figure 1.5: Low power design flow

3. Produce power optimization description.

Step 1 and Step 2 are very important for low power design. Accurate power estimation tools must provide feedback on the quality of each design choice. In the last ten years, at each level of abstraction, various design alternatives have been explored. The availability of a power estimator for each level of abstraction is fundamental in a low-power design flow. To avoid costly re-design steps, it is mandatory to be able to optimize the power dissipation during the early stages of the design process.

1.4 Low power digital design techniques

Low power digital design techniques cover a broad range of subjects. In this thesis, three topics are chosen to discuss.

1.4.1 State assignment for low power Finite State Machines (FSMs)

Most of VLSI circuits are sequential circuits. A general model of sequential circuit structures is called Finite State Machine (FSM), which is composed of two sections: Combinational section and register section. Register section usually is composed of flip-flops. Compared to combinational circuits, there are two working characteristics:

- An FSM has flip-flops to store state signals.
- An FSM has clock signals to synchronously trigger flip-flops and to realize the synchronous switching of state variables.

Thereby, its synthesis procedure is slightly different from combinational circuits. FSM synthesis can be divided into four stages[15]:

- Behavior synthesis: Obtain representation of machine behavior expressed as State Transition Tables (STTs) or State Transition Graphs (STGs).
- State assignment: Assign unique binary code to the symbolic states of an FSM and obtain a description of the circuit produced in terms of Boolean functions.

- Logic synthesis: Optimize Boolean functions with respect to the original cost metric.
- Library binding: Map the optimization Boolean functions to components from a standard gate library and produce a gate-level description.

State assignment is the critical step in low power design of FSMs and is one subject of this thesis. It has an important role in determining the number of nodes required to implement the output and next logic functions. The reduction of the number of nodes promises global power savings. On the other hand, state assignment directly determines the switching activities of the state variables and the interior variables in the combinational circuit.

The contribution of this work is in the formulation of the problem that links switching activities of an FSM to its power dissipation and in the study of Genetic Algorithms (GAs) for the search of optimal solutions to the problem of finding a state assignment that gives low power dissipation. Only minimizing switching activities on the state lines in the FSM does not guarantee the reduction of the total power dissipation because the power consumed in the combinational section is not taken into account. Instead, more accurate cost functions which take that of the combinational section into account have been developed. A methodology and a software package that combines ESPRESSO[23] are developed. Test results on MCNC benchmark circuits show that our package performs significantly better than other synthesis tools [94][111] in the majority of cases.

1.4.2 Low power flip-flop design

Flip-flops are the core components of FSMs. Reducing power dissipation of flip-flops can result in significant power reduction of FSMs. Traditional flip-flops are single-edge triggered flip-flops (SETFF), which are sensitive to the rising or falling edge of the clock. Narrow pulse sampling based single edge-triggered flip-flops are implemented using a clock chain to generate a series of narrow pulses. It suffers from significant redundant transitions in a long clock chain, which results in redundant dynamic power dissipation. To solve this problem, a clock-gating scheme is proposed to eliminate the redundant transitions and a low power single edge-triggered flip-flop is presented. On the other hand, for single edge-triggered flip-flops, half of the clock's transitions are redundant, which results in wasteful dynamic power dissipation. To improve this, double-edge triggered flip-flops (DETFFs) are proposed, which utilize both transition edges of the clock, achieving power savings. In this thesis, a low power SETFF is proposed, in which clock-gating techniques are used to reduce the redundant transitions of clock signals for saving power and the structure of low power DETFFs are explored and multiple valued flip-flops are investigated to explore a novel solution of low power flip-flop design.

1.4.3 Low power logic synthesis for FPRM functions

Any n-variable Boolean function f can be expressed by Shannon expansion based on AND/OR operation as follows.

$$f(x_{n-1}x_{n-2}\cdots x_0) = \overline{x_i}f_{x_i=0} + x_if_{x_i=1} \tag{1.4}$$

where $0 \leq i \leq n-1$, and $f_{x_i=0}$ and $f_{x_i=1}$ are the cofactors of f with respect to x_i. Correspondingly, a broad range of logic minimizers are available for SOP forms such as ESPRESSO [23] and SIS[94].

Alternatively, any Boolean function can be represented by AND/XOR operations, which is called Reed-Muller expansion.

$$f(x_{n-1}x_{n-2}\cdots x_0) = \overline{x_i}f_{x_i=0} \oplus x_if_{x_i=1} \tag{1.5}$$

$$f(x_{n-1}x_{n-2}\cdots x_0) = f_{x_i=0} \oplus x_i(f_{x_i=0} \oplus f_{x_i=1}) \tag{1.6}$$

$$f(x_{n-1}x_{n-2}\cdots x_0) = f_{x_i=1} \oplus \overline{x_i}(f_{x_i} \oplus f_{x_i=1}) \tag{1.7}$$

In logic synthesis, Reed-Muller logic methods are important alternatives to the traditional SOP approaches to implement Boolean functions. Reed-Muller realizations have some attractive advantages especially for functions that do not produce efficient solutions using SOP techniques. In addition, XOR based circuits have great advantage of easy testability. However, due to the lack of efficient conversion tools and Reed-Muller logic optimizer, applications of Reed-Muller implementations have not become popular. With the development of FPGAs, XOR gates are already manufactured as basic cell components, which encourages the research on Reed-Muller logic optimization. There has been extensive research on Reed-Muller methods targeting area minimization[4][6][8][79][64][114]. However, the research on low power Reed-Muller logic implementations is still in their early stage of development.

XOR Gate decomposition is the step before technology mapping, which decompose a multi input XOR gate into a two input XOR gate tree. Low power XOR gate decomposition is one solution of low power Reed-Muller logic implementations. Based on polarity conversion, a novel XOR gate decomposition targeting low power is proposed.

Any Boolean function can be represented canonically by a Fixed Polarity Reed-Muller (FPRM) form. Minimization of FPRM functions promises global power savings. A power estimation frame for FPRM functions is proposed. Based on polarity conversion, a power minimization algorithm is developed.

1.5 Outline

This thesis covers three main parts: Low power state assignment for FSMs; low power flip-flop design and power optimization for FPRM functions. The conventional low power design techniques are reviewed in chapter 2. The first part investigates low power state assignment for FSMs. The problem is formulated in chapter 3. Two cost functions are proposed, which take the power of the combinational section into account. Genetic Algorithm combined with ESPRESSO is used to search the optimal solution. The second part is for low power design of flip-flops. A new type of low power single-edge triggered flip-flop is proposed in chapter 4 while double-edge triggered flip-flops are presented in chapter 5. A multiple valued approach to design low power flip-flops is explored in chapter 6. The third part deals with Reed-Muller logic which is based on AND/XOR operations. Based on polarity searching, low power XOR gate decomposition is stated in chapter 7 while low power FPRM function minimization is described in chapter 8. Finally, the main improvements and contributions are summarized and some future work is suggested in the "conclusions and future work".

Chapter 2

Conventional low power design techniques

2.1 Low power design approaches of FSMs

Most of VLSI circuits are sequential ones. A sequential function can be represented by several models [50]. Usually, it is modeled by a finite state machine (FSM) [67].

2.1.1 Finite state machines and their representations

Low power design of FSMs involves tackling the problem of the power estimation method and power optimization strategy. They can be done at each of four design levels [15].

An FSM is defined by the following standard definition.

Definition 2.1. An FSM is characterized by a 5-tuple (X, Y, S, λ, η) where X, Y, S are the sets of primary inputs, primary outputs and internal states and λ, η are the output and next state functions, respectively. The FSM is represented by a state transition table (STT) $M = \{m_l | m_l = (x_l, (s_i)_l, (s_i)_l^{\prime}, y_l), l \in \{0, 1, ...L-1\}\}$, here, L is the number of product terms, x_l is the primary input, s_i, $s_i^{\prime} \in S$ are the present state and the next state and y_l is the corresponding output. Each entry $m_l \in M$ is a symbolic implicant of the FSM.

FSMs are categorized in two classes [14]:

Definition 2.2. A Moore machine is an FSM where $\lambda(x, s) = \lambda(s)$, i.e., the outputs do not depend directly on the input value, but they depend only on the state. A Mealy machine is an FSM for which this property does not hold.

FSMs can be incompletely specified. An incompletely specified FSM is one where $\eta(x, s)$ and / or $\lambda(x, s)$ are incompletely specified Boolean functions. An FSM can be represented by a graph or, equivalently, by a table. The two representations are called state transition graph (STG) and state transition table (STT), respectively. The states of the STG are labeled with the unique symbolic state name. The edges are labelled with the input and output values. The state table is simply the list of edges of the STG.

Both STG and STT completely define the input-output behavior of an FSM, but they do not provide any information about the circuit implementation. Hence, STG and STT are behavioral representations of the FSM. In order to obtain a representation which is closer to the circuit implementation, the concept of state encoding is needed to be introduced.

Definition 2.3. A state encoding is a one-to-one mapping from S to B^{N_s} (Boolean space), i.e., a function E: $S \rightarrow B^{N_s}$. The number of state variables is indicated by N_s and $N_s \geq \lceil log_2|S|\rceil$.

Once N_s and E have been specified, the state of an FSM is completely expressed by N_s binary variables called state variables. Once the state encoding has been specified, the structural model of an FSM is shown in Fig. 2.1. The representation of Fig. 2.1 is structural, because it refers to a particular circuit structure implementing the FSM.

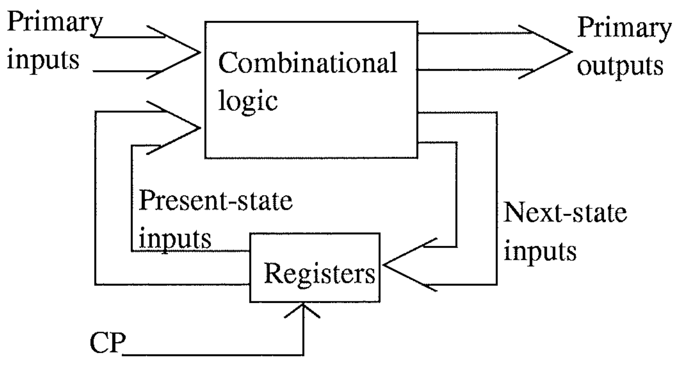


Figure 2.1: FSM structure model

The structural representation is useful since it may be much more compact than the state-based representations like STT and STG. However, the main limit of structural representations is that they are not unique.

Binary decision diagrams (BDDs) are a data structure developed for the compact representation of large Boolean functions. Several variants of BDDs have been developed by different groups of researchers [22][20]. It has been shown that the Reduced Ordered BDD (ROBDD) is a canonical form, i.e., two functions are equivalent if and only if they have the same BDD. There are some distinct advantages to represent an FSM by BDD [39].

2.1.2 FSM power estimation

The process of IC design involves a transformation of a high level behavioral specification to a lower level architectural (or RTL) specification, and then to a lower gate-level specification, and so on. If one were to get to the transistor or gate level design and only then discover that the power consumption is unacceptably high, it would be too expensive to make design changes. The circuit may require significant rework, involving perhaps changes to the overall architecture of the chip. For this reason, it would be very beneficial to have a power estimation capability at a high level of abstraction. However, estimation from a high level of abstraction is potentially inaccurate, while low-level power estimation can be very accurate. Therefore, a power estimation capability is needed at every level of abstraction in order to check the design at every step.

- SPICE

The acronym SPICE stands for Simulation Program with Integrated Circuit Emphasis [86]. It is a general-purpose circuit program that simulates electronic circuits and can perform various analysis of electronic circuits: the operating points of transistors, a time-domain response, a small-signal frequency response, and so on. SPICE contains models for common circuit elements, active as well as passive, and it is capable of simulating most electronic circuits. It is a versatile program and is widely used by industries and universities. The main frame versions are HSpice (Meta-Software), PSpice (MicrSim), AccuSim (Mentor Graphics) and Cadence-SPICE (Cadence Design). The PC-version, PSpice (MicroSim), is also available. PSpice allows various types of analysis, which include DC Sweep, Transient Analysis and AC Analysis. For the average power estimation of a circuit, the library function $avg(i \times v)$ can be called where i is the node current and v is node voltage. However, the power dissipation is input pattern-dependent. Hence, the utility of straightforward simulation can be limited because it is time consuming and can only work at transistor level.

- Monte-Carlo-based power estimation

The basic idea of Monte Carlo methods for estimating activities of individual nodes is to simulate a circuit by applying random-pattern inputs. The convergence of simulation can be obtained when the activities of individual nodes satisfy some stopping criteria. The procedure is outlined in Fig. 2.2 . The detail is discussed in [73].

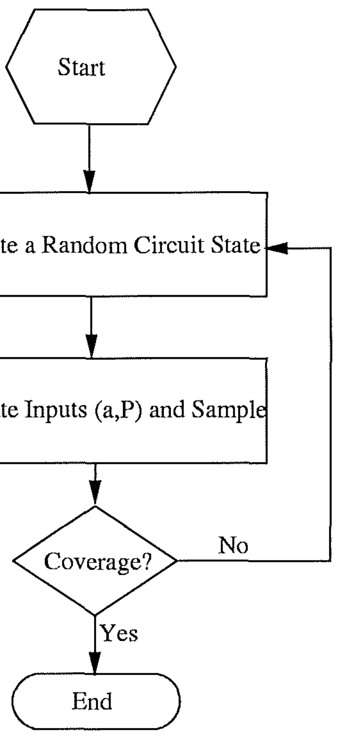

Start
Generate a Random Circuit State
Generate Inputs (a,P) and Sample
Coverage?
No
Yes
End

Figure 2.2: Monte-Carlo-based technique flow chart

- Encoding based power estimation

This targets the STT or STG level of abstraction. At the STT or STG level of abstraction, nothing has been decided about the structure of the combinational logic implementing the next state and output functions. The state assignment algorithm can exploit degrees of freedom that are lost at successive phases and produce an encoded state transition table that is an effective starting point for further power optimizations of the combinational logic. Some power estimation methods have been developed at this level.

Power is a strongly pattern-dependent cost function, since it depends on the switching activity of a circuit, which in turn depends on the input patterns applied to the circuit. Hence, some information about the typical input patterns applied to a circuit needs to be specified to estimate its power dissipation. The most straightforward way to provide information about input patterns is to actually provide a long input stream representing a typical usage pattern together with the specification of the circuit. But it suffers from two drawbacks that the input traces can be very large and cumbersome to manage and in many cases only incomplete information about the environment may be available. Hence, instead, input signals are described by input signal probabilities [89].

Definition 2.4. *Signal probability:* Let $I(t)$, $t \in (-\infty, +\infty)$, be a stochastic process that takes the values of logical 0 or logical 1, transitioning from one to the other at random times. The signal probability of signal $I(t)$ is given by

$$p(I) = lim_{T\to\infty} \int_{-T}^{+T} I(t)dt \tag{2.1}$$

Definition 2.5. *Signal Activity:* The signal activity of a logic signal $I(t)$ is given by

$$A(I) = lim_{T\to\infty} \frac{n_I(T)}{T} \tag{2.2}$$

where $n_I(t)$ is the number of transitions of $I(t)$ in the time interval between $-T/2$ and $+T/2$.

More detail state signal probability (briefly state probability) calculation is discussed in [89]. Two methods are developed to calculate the state probability: Explicit methods and ADD based methods.

Explicit methods use iterative method [44] or Gaussian elimination method [119] to compute the state probabilities. The main limitation of this method is that it is not applicable to very large size FSMs for which the state set is large enough to make even the storage of matrix P a formidable task.

ADD based methods use Algebraic decision diagrams (ADDs) data structure to compute the state probabilities to avoid manipulating the transition probability matrix. ADDs [13] are "BDD-like" data structure.

The main difference is that an ADD has multiple terminals while a BDD only has two terminals: 0 and 1. ADD based methods allow the manipulation of very large systems by representing the transition probability matrix with an ADD.

The STG of a finite-state machine is implicitly represented by a BDD (or, equivalently, by a 1/0-ADD) of its transition relation [30]. The transition relation is a Boolean function $T(x, s, s')$. The support of the transition relation consists of the input variables, the state variables and the next state variables. T has value 1 when the STG of the machine has a transition from state s to state s' with input x, zero otherwise. Similarly, the input probabilities can be represented by an ADD. The ADD $PI(x)$ is extracted from the array of input probabilities with the simple formula $PI(x) = \Pi_{i=0}^{m-1} PI_i(x_i)$ where each $PI_i(x_i)$ is a single-node ADD with two leaves with value p_i and $1 - p_i$.

Given T and PI, the implicit representation of matrix P can be obtained by the following symbolic formula:

$$P(x, s) = PI(x) \cdot \sum_{s'} T(x, s, s') \tag{2.3}$$

The ADD of P can be exponentially smaller than the traditional matrix representation.

Example 2.1. The STG of an FSM is given in Fig. 2.3(a) with input, output and conditional transition probability at each edge. The transition relation $T(x, s, s')$ is shown in Fig. 2.3(b) in the tabular format. The ADD of T is shown in Fig. 2.3(c). The ADD of the conditional input probability is shown in Fig. 2.3(d). The result of the conditional transition probabilities is shown in Fig. 2.3(e).

After computed conditional transition probabilities, the state probabilities can be computed using the symbolic version of the power method[44]. The symbolic representation based on ADD becomes useful when the STG and the truth table of T are unmanageably large.

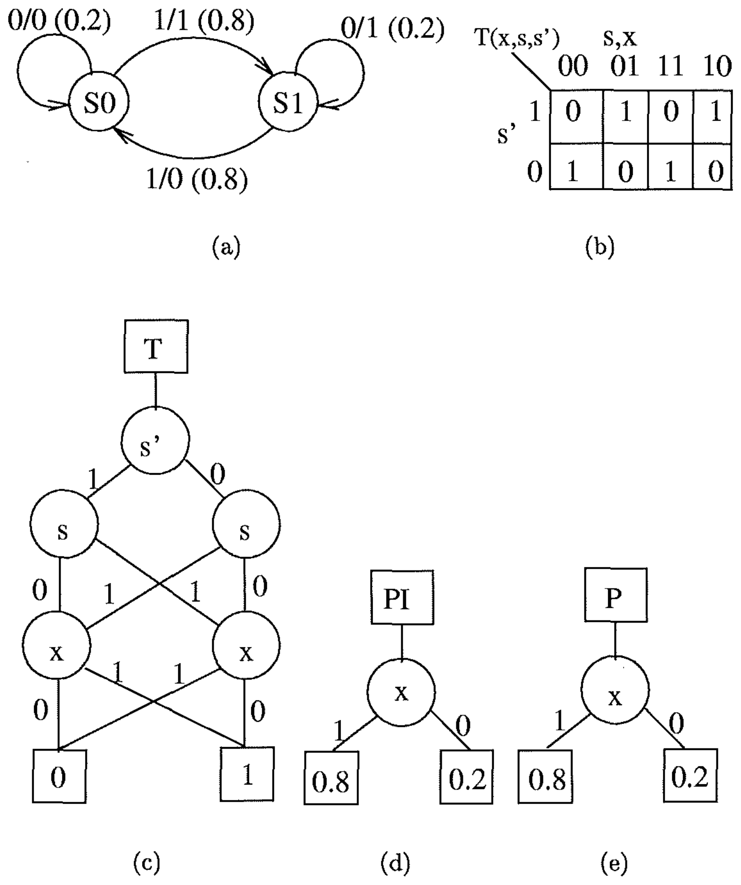


Figure 2.3: Symbolic computation of the conditional transition probability matrix

Based on conditional state transition probability and the state probability, the state transition probability can be calculated[89]. Because of no detail circuit structure available at this level, switching activity is used to measure the power dissipation [88][77][15][12][48][117]. Given the state encoding, each state is represented by binary code and Hamming distances can be calculated for each state transition. Then, the switching activity of an FSM can be computed based on the state transition probability and Hamming distances.

- Information-theory-based approaches

Recently, information theory has been used quite effectively to estimate power at the RT (register transfer) level of design abstraction [75][62]. The RT level abstraction assumes that the Boolean functionality of the circuit is known while the details of the implementation are unknown. Information-theoretic approaches depend on information-theoretic measures of activity (for example, entropy) to obtain quick power estimate.

Entropy characterizes the randomness or uncertainty of a sequence of applied vectors and thus is intuitively related to switching activity, that is , if the signal switching is high, it is likely that the bit sequence is random, resulting in high entropy. Suppose the sequence contains t distinct vectors and let p_i denote the occurrence probability of any vector v in the sequence. Obviously, $\sum_{i=1}^{t} p_i = 1$. The entropy of the sequence is given by

$$h = -\sum_{i=1}^{t} p_i log_2 p_i \tag{2.4}$$

This equation is only an upper bound on the exact entropy, since the bits may be depended. This upper bound expression is, however, the one that is used for power estimation purpose. Furthermore, in [75], it has been shown that, under the temporal independence assumption, the average switching activity of a bit is upper bounded by one-half of its entropy. Based on entropy, the methods to estimate the low bound and high bound of an FSM switching activity have been presented in [109][63].

- Power estimation included glitching

In the combinational section of the FSM, glitching power is inevitable. It has been observed that this additional power dissipation is typically 20% of the total power, but can be as high as 200% of the total power in some cases such as in a multiplier[123]. There are some publications on the power estimation included glitching for FSMs [38][84][123]. Unit delay and general delay models are proposed to estimate the glitching included power for circuits [42]. However, overestimation of the power dissipation is still possible under unit delay model. Furthermore, multiple-option delay models in a tool are not so convenient in practical applications. Monte-Carlo-Based approach can estimate the exact glitching included power dissipation provided exact signal probabilities and activities of primary inputs are known. However, accurate signal probability or activity values for primary inputs may not often be available. Since power dissipation strongly depends on the input signal properties, uncertainties in specifications of input signal properties make the estimation process difficult[73]. Hence, more work needs to be done.

2.1.3 FSM power optimization

Different power optimization strategies have been applied at the different design levels. Here, some important logic synthesis transformations will be discussed. Sequential logic optimization methods work at two levels of abstraction, namely the State Transition Graph level and at the logic-gate level. Several approaches have been developed in these levels as follows.

- State assignment

State assignment and the resulting combinational logic synthesis have been conventionally targeted at reducing area and critical path delay [67][111][33]. For the optimization of power dissipation, these methods have to be modified to target a power cost function, namely, weighted switching activity. State assignment significantly affects circuit power dissipation because different state assignments have different switching activities. Given an FSM, the state transition probabilities between any two states can be obtained [15][48]. A state transition in an FSM could be caused by single or multiple bit transitions. Therefore, the switching activity could be minimized if states associated with state transitions that appear most frequently are assigned codes that are close to each other. Several methods have been proposed for power-oriented state assignment [88][77][15][48], which are mainly based on the minimal average Hamming distance. Methods to encode State Transition Graphs to produce two-level and multilevel implementations with minimal power are described in [117] and [108]. A method to re-encode logic-level sequential circuits to minimize power dissipation is presented in [43]. The problem of finding the state assignment for the power optimization is computationally hard. There is no known method of predicting the optimum assignment for the states though many algorithms have been proposed [108][77][15][48][117].

- Clock-gating

This scheme is based on the observation that during the operation of an FSM there are conditions such that the next state and the output do not change (i.e., the machine is internally idle). Therefore, clocking the FSM only wastes power in the combinational logic and in the registers in this case. If the idle condition of the machine can be detected, the clock can be stopped until a useful transition must be performed and the clocking is resumed. Fig. 2.4(a) is an input latched FSM, which is different from the generic FSM structure in Fig. 2.1. Fig. 2.4(b) is a clock-gated FSM structure. F_a, activation function, is to selectively stop the local clock of the FSM when the FSM is internally idle. The block labeled "L" represents a latch, transparent when the CK is low. The presence of a gated clock has a two-fold advantage. First, when the clock is stopped, no power is consumed in the FSM combinational logic, because its inputs remain constant. Second, no power is consumed in the registers and the gated clock line. In an arbitrary sequential circuit, some parts of a circuit are not accessed in each clock cycle. A detailed discussion of this technique is given in [16]. However, for those which have no internal idle states, the limitation of this technique is obvious. To overcome this limitation, a new technique is presented in [90], which can create idle states so that clock-gating technique can be used to save power. If simple conditions that determine the inaction of particular registers can be determined, then power reduction can be obtained by gating the clocks of these parts [17]. An extension version of this technique is called precomputation technique which is presented in [2].

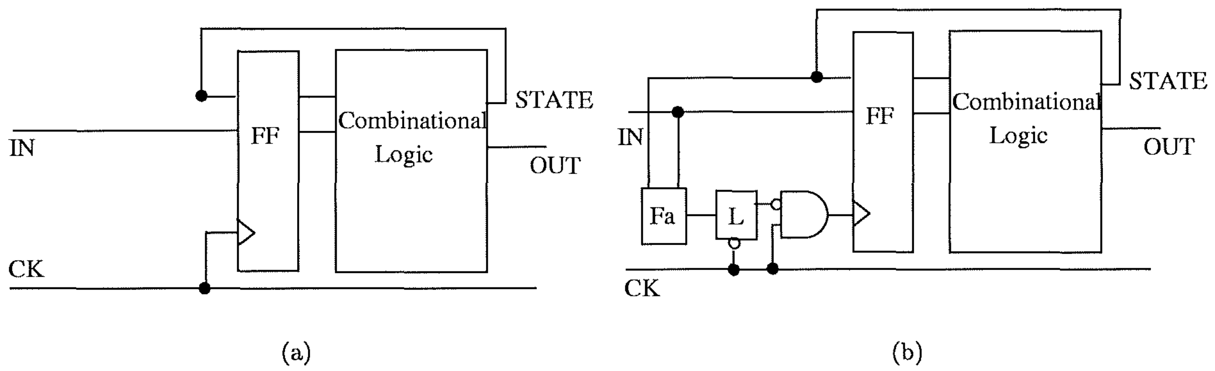


Figure 2.4: FSM structure model (a) single-clock, flip-flop based FSM model; (b) clock-gating version

- Partitioning

The fundamental intuition behind this technique is that a sequential circuit may be partitioned into a set of small interacting blocks. During operation, only one block is active at any given time and controls the input-output behavior. In the remaining blocks, the clock can be stopped and, consequently, the total power consumption is reduced. FSM partitioning has been extensively studied for several decades. Its theoretical foundations were laid down by Hartmanis and Stearns [46] in the sixties. More recent work [41] reported experimental results on the implementation of the partition procedures described in [46]. A different viewpoint on the problem was proposed in recent years by Ashar, Devadas and Newton [10] who presented numerous algorithms for the automatic partition of FSMs specified by an monolithic state transition graph (STG). However, all these techniques are for minimum-area implementation. Recently, partitioning techniques have been explored to reduce the power dissipation of FSMs. Chow et al. [29] proposed a low-power partition approach based on the relationship between state assignment and FSM partition that produced very promising results. Dasgupta et al [32] proposed an approach which is accomplished in two stages namely disjunctive partitioning and selective isolation encoding while Monteiro and Benini suggested approaches to combine partitioning technique with clock-gating[71][18]. The principle behind this technique is to partition an FSM into number smaller interacting submachines such that only one submachine is active during any clock cycle. The rest of circuit comprising of the other submachines is turned off, thus avoiding unnecessary power-dissipating switching.

- Retiming

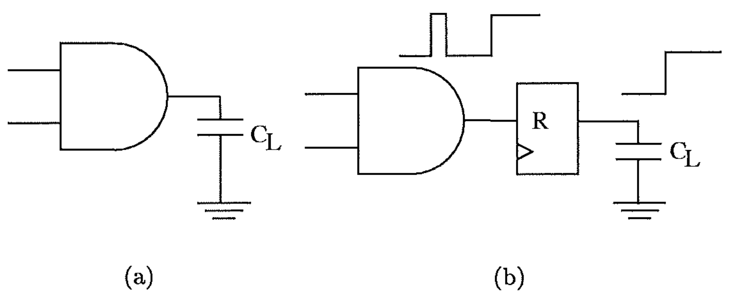


Figure 2.5: Reducing the switching activity by inserting register

The transformation that repositions the registers of a design without modifying its external behavior is called retiming. Monteiro et al. [70] have pointed out that register positions can also affect power dissipation. Consider the simple example of a logic gate belonging to a synchronous circuit as in Fig. 2.5(a), and call C_L the capacitance load driven by the output node of AND gate. In the case of CMOS technology, the power dissipated by AND gate is proportional to the product of the switching activity of the output node of the gate α_g and the output load C_L. Now consider the case in which a register R is connected to the output of AND gate. Let C_R be the input capacitance of the register, and let α_R be the switching activity of the register output [see Fig. 2.5(b)]. The total power dissipated by the new circuit is proportional to $\alpha_g C_R + \alpha_R C_L < \alpha_g C_L$ if both α_g and C_L are sufficiently high. Retiming repositions the flip-flops in a synchronous sequential circuit so that the spurious transitions at the inputs to the flip-flops can be filtered out by the clock as shown in Fig. 2.5. A retiming method that targets the power dissipation of a sequential circuits is described in [70].

2.2 Low power design of flip-flops

It is found that although the power distribution of VLSI's differs from product to product a clock system and its logic part consume almost the same power in various VLSI chips and the clock system consumes 20%~45% of the total chip power. In the clock system power, 90% is consumed by the flip-flops themselves and the branches of the clock distribution network which directly drives the flip-flops [54]. Hence, low power design of flip-flops has attracted many researchers [49][19][68][102][54]. Based on logic families, flip-flops can be grouped into pass transistor based flip-flops[49][19], true single phase clocking (TSPC) based flip-flops, differential designs and clocked CMOS (C^2MOS) flip-flops.

Low power design of flip-flops have been carried out along the following lines.

- Reducing the number of transistors used in flip-flops [49]

Reducing the number of transistors in flop-flops can reduce the internal nodes of signal transitions. Hence, it can not only save area but also implement global power savings.

- Using double edges of clock signal to trigger flip-flops [110][49][19][102][58][40][1]

Compared to single edge-triggered flip-flops, the clock frequency for double edge-triggered flip-flops can be reduced into half and hence clock system power is reduced. There are two kinds of schemes to implement double edge-triggering. One is to use two latches or flip-flops to receive input signals at both clock phases alternatively and use a multiplexer at the output section to selectively output the stored signal[49][19][80]. This scheme often trades with the increasing transistor number compared to the single edge-triggered flip-flops. The other is narrow pulse-triggered scheme. This kind of flip-flop is composed of two parts: a pulse generator and a latch (or a flip-flop cell). The pulse generator usually consists of a series of inverters (3 or 4 inverters) while a latch (or a flip-flop cell) uses clock racing signals to generate a narrow pulse corresponding to each of clock transition edges[102][72]. A good performance comparison for variety double edge-triggered flip-flops is shown in [68].

- Reducing clock signal swing to achieve power improvement[54]

This scheme is based on reducing clock signal swing to reduce the clock system power. However, if there are clocked pMOS transistors in the circuit, it will make power supply complicated to avoid static power dissipation increase.

- Using clock gating techniques to deactivate clock signal so that power dissipation is reduced [100][101]

Clock signal is the most active signal in a flip-flop if glitches are not taken into account. Clock signal triggers a flip-flop and also triggers most of internal nodes. However, if input signal is not changing or has low signal activity, the clock triggering is redundant. In these cases, if clock signal can be deactivated, the power can be saved. The principle behind clock gating techniques is based on this idea.

2.3 Power optimization of XOR gate based circuits

Any Boolean function can be expressed canonically based on AND and XOR operators using Reed-Muller (RM) expansions. Because Reed-Muller realizations have several attractive advantages especially for functions which do not produce efficient solutions using SOP techniques, research on Reed-Muller logic has attracted more and more investigators [4][64][114][8][115]. However, most of these work is on area minimization. The basic approaches to minimize area are:

- Polarity optimization methods to find the best polarity with the least number of product terms or literals.

To find the best polarity is computationally extensive in both space and time especially for large functions. Traditionally exhausting search is only suitable for the smaller functions which are less than 15 input variables[6][35][92]. However, progress has been made in [116], which can solve large functions which have up to 25 input variables. Several heuristic methods have been proposed which apply the simulated annealing [79] or genetic algorithm techniques[8][9].

- Decomposition

Decomposition method is based on the concept of $\frac{3}{4}$ majority cube [105]. The principle behind this method is that an m-dimensional cube covers at least $\frac{3}{4} \times 2^m$ on-set $\pi-$terms. The method is further generalized to very large multiple output functions in [114].

- Mixed polarity minimization

The product terms can be reduced with mixed polarity by combining the adjacent product terms such as using XOR-link operation[97].

Compared to the area minimization, little work has been presented on power optimization for XOR gate based circuits.

In [131], a multiple input XOR gate is decomposed into a tree of two input XOR gates. The aim is to search an optimized input signal combination which has minimum power dissipation. In [74], FPRM functions are implemented into XOR trees and AND trees and power dissipation is optimized with factorization and reduction rules. One limitation for these methods is that they only optimize power under the specific function form or specific polarity.

2.4 Genetic algorithm

In this section, some basic operations of Genetic Algorithms (GAs) are introduced, which will be further discussed in the next chapter.

2.4.1 Overview

First proposed by John Holland in 1975 [47] and then developed by his colleagues and students, genetic algorithms (GAs) have been an attractive class of computational models that mimic natural evolution to solve problems in a wide variety of domains. GA emulates biological evolutionary theories to solve optimization problems and composes of a set of individual elements (the population) and a set of biologically inspired operators defined over the population itself. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offsprings, thus transmitting their biological heredity to new generations.

The basic procedure is to create a population (breeding pool) of potential solutions to a problem. These solutions are encoded as "chromosomes" (data representation of the solution), and each chromosome is subjected to an evaluation function which assigns "fitness" depending upon how well the solution it encodes solves the problem at hand. Existing solutions are recombined by a process called crossover or breeding. The rational for this is that good solutions will contain good building blocks, rearrangement of which may produce even better solutions. Further a mutation process makes random changes in a few randomly selected chromosomes. This prevents premature convergence by maintaining the diversity of the population. GA operates through a simple cycle of stages:

- Creation of a "population" of chromosomes
- Evaluation of each chromosome
- Selection of "best" chromosomes
- Genetic manipulation to create new population of chromosomes

Fig 2.6 shows these four stages using GA.

2.4.2 Representation

Fundamental to the GA structure is the encoding mechanism for representing the optimization problem's variables. The encoding mechanism depends on the nature of the problem variables. In each case the encoding mechanism should map each solution to a unique binary string.

Example 2.2. Consider optimizing the function $f(x) = x^2$ where the continuous variable x is defined in a range from [0,2] with an accuracy of two decimal places after the decimal point.

The variable can be encoded by a binary code. The mapping from a binary code into a real variable value within the range [0, 2] can be implemented as in equation 2.5.

$$x = \frac{2x'}{2^8 - 1} \tag{2.5}$$

Here, x' is binary code $< b_7 b_6 \cdots b_1 b_0 >$ and 2 is the length of the domain. The reason to choose 8 bit binary code is to meet the accuracy because

$$128 = 2^7 < 200 < 2^8 = 256$$

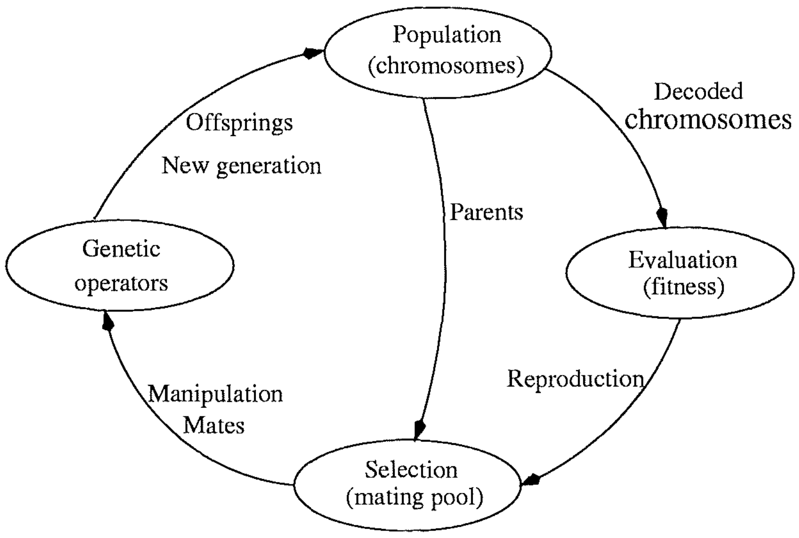


Figure 2.6: GA cycle

For example, a chromosome x' =(01101100) represents 0.85, because

$$x^{'} = (01101100)_2 = (108)_{10}$$

and

$$x = 2\frac{108}{255} = 0.85$$

The chromosomes (00000000) and (11111111) stand for the boundaries of the domain, 0 and 2.0, respectively.

2.4.3 Initial population

A population of chromosomes needs to be initialized. Any of possible 8-bit binary codes could be a chromosome in the above case, which is initialized randomly. The population size (the number of chromosomes) is set depending on the application.

2.4.4 Evaluation

An evaluation function returns a measurement of the worth of any chromosome in the population. Evaluation function $Eva(x^{'})$ is equivalent to the function $f(x)$.

$$Eva(x^{'}) = f(x)$$

For example, $x^{'} = (011011000)$ corresponds to $x = 216$.

$$Eva(01101100) = f(0.85) = 0.72$$

Usually, $Eva(x^{'})$ or $f(x)$ needs to be transfered to fitness function $fit(x^{'})$:

$$fit(x') = h(Eva(x')) = h(f(x))$$

The fitness stands for a measure of how good the chromosome is.

2.4.5 Parent selection

Selection models nature's survival-of-the-fittest mechanism. Fitter solutions survive while weaker ones perish. There are many ways to do this. The most popular parent selection scheme is the roulette wheel parent selection. It works by allocating pie-shaped slices on a roulette wheel to population members, with each slice proportional to the population member's fitness. Selected parent can then be viewed as a spin of the wheel, with the winning population member being the one in whose slice the roulette spinner ends up.

2.4.6 Crossover

Crossover recombines the genetic material in two parent chromosomes to make two children. The simple crossover is one point crossover, which occurs when parts of two parent chromosomes are swapped after a randomly selected point, creating two children. Fig. 2.7 shows an example of the above application of one point crossover supposing that the cut point is selected after the 6th bit (usually called gene).

$$\begin{array}{lcl} \text{Parent 1: } 011000 \mid \mathit{01} & & \text{Child 1: } 011000 \mid \mathit{10} \\ & \longrightarrow & \\ \text{Parent 2: } 110110 \mid \mathit{10} & & \text{Child 2: } 110110 \mid \mathit{01} \end{array}$$

Figure 2.7: Crossover operator

2.4.7 Mutation

Mutation is a secondary operator with the role of restoring lost genetic material. It also reduces the possibility of early convergence on a local optimum solution. Mutation of a bit involves flipping it from 0 to 1 or vice verse. For example, for child 1: 01100010, if the sixth gene is selected for a mutation, it would be flipped from 0 to 1 and result in a new chromosome 01100110.

2.4.8 Elitism

After crossover and mutation, the best members of the population may fail to produce good offsprings in the next generation. To restore the possible loss, the best members of each generation are copied into the succeeding one.

Chapter 3

State assignment for area and power optimization

In this chapter, a genetic algorithm (GA) based state assignment targeting area and power optimization is developed.

3.1 Finite state machines (FSMs)

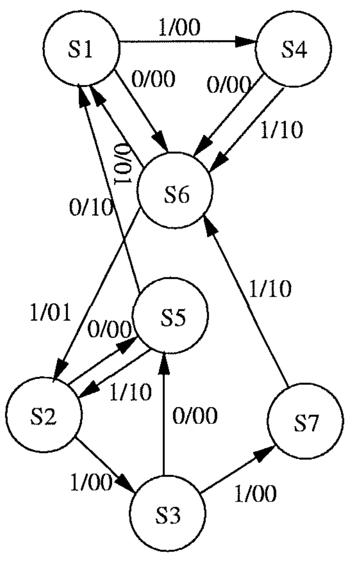
A finite state machine (FSM) is represented by a set of states and a set of their associated transitions.

An FSM can be represented by a graph or, equivalently, by a table. The two representations are called state transition graph (STG) and state transition table (STT), respectively. The states of an STG are labeled with the unique symbolic state names. The edges are labeled with the input and output values. The state table is simply the list of edges of the STG. Take DK27 in 1991 MCNC benchmarks [129] for example as shown in Fig. 3.1.

FSMs can be incompletely specified. An incompletely specified FSM is one where $\eta(x, s)$ and /or $\lambda(x, s)$ are incompletely specified Boolean functions. To synthesize an FSM, the symbolic state names must be encoded as unique binary codes. The binary codes assigned to the symbolic states determine the circuit's combinational logic. Then state assignment is defined in Definition 3.1.

Definition 3.1. State assignment is a mapping from the set of states of an FSM to the set of binary codes.

It is well known that an FSM's state assignment can significantly affect the quality of synthesized circuits. Much of state assignment research has been concentrated on reducing the circuit area. NOVA[111] makes state assignments which target minimal-area two-level logic while MUSTANG[33], JEDI[57] and MUSE[36] target multilevel-logic implementation. Recently, several researchers have focused on low-power designs using state assignment that reduces the average switching frequency of the states [88][12][108][77][15][48][117]. The Syclop [88] method considered conditional state transition probabilities as weight coefficients in the cost function, while Hong [48] and Wang [117] methods exploited the total state transition probabilities. The shortcoming of the above approaches is that they minimize the switching activity on the present state bits without any consideration of the area in the combinational section of FSMs. As a result, the area overhead is high compared to area-oriented state-of-the-art tools. To consider the area constrain in combinational section, Olson [77] and POW3 [15] introduced a convex combination



(a)

In	PS	NS	O
0	S1	S6	00
0	S2	S5	00
0	S3	S5	00
0	S4	S6	00
0	S5	S1	10
0	S6	S1	01
0	S7	S5	00
1	S6	S2	01
1	S5	S2	10
1	S4	S6	10
1	S7	S6	10
1	S1	S4	00
1	S2	S3	00
1	S3	S7	00

(b)

Figure 3.1: State transition graph and state table of DK27 (a) STG; (b) STT

of the switching activity and the area of the combinational logic as cost function. Wang [113] proposed a matching-based state assignment algorithm to minimize area and state transitions simultaneously. However, all of those results show that minimization of switching activity has to trade with area penalty. As we know, area overhead will result in many disadvantages such as cost and reliability of circuits. Further, high area overhead will in turn offset the power reduction in state registers.

The problem of finding the state assignment for the minimization of power consumption and area is computationally hard. The two primary techniques used to solve the problem are the greedy search and the simulated annealing. However, the search space appears to be too large with many local minima for these schemes to find the global minimum. The genetic algorithm (GA) technique has been successfully applied to a variety of computationally difficult problems which have a large search space. It has been shown that it can produce good results in reasonable computation time. A recent investigation showed that GA can find better assignments than commercial products for area minimization [5]. Olson [77] employed a genetic local search to perform a local optimization of FSMs and got encouraging results. In this Chapter, a new scheme is proposed and genetic algorithms (GAs) are employed to optimize both switching activities and area without the need to carry out an exhaustive search.

The remainder of the chapter is organized as follows. The terminology used here is defined in Section 2.1.1. Section 3.1 introduces the FSM calculation while Section 3.3 defines the cost function. The state assignment algorithms are described in Section 3.4. Experiment results and conclusions are given in Sections 3.5 and 3.6, respectively.

3.2 Terminology and parameter calculation of FSMs

Power dissipation is a strongly pattern-dependent cost function. Here, input signals are described by input signal probabilities.

Given a circuit, suppose the input signal probability, p_i, is known. However, in an FSM, the probability of a state transition depends not only on the inputs but also on the state information. Considering a transition between two state s_i and s_j. If state s_i is unreachable, the machine will never perform the transition because it will never be in state s_i. Similarly if the probability of being in state s_i is very low, a transition from state s_i to state s_j is very unlikely. A parameter to describe the probability of a transition is called conditional probability.

Definition 3.2. Given a set of inputs $\{I_0 I_1 \cdots I_{n-1}\}$, *Conditional State Transition Probability (CSTP)* p_{ij} associated with a transition from s_i to s_j (briefly ts_{ij}) is the ratio of the number of input minterms causing such transition to the total number of valid input minterms at state s_i. $i,\ j \in \{0,\ 1,\ 2, \cdots,\ n-1\}$ and n is the number of states.

If the input probability is not specified, a default input probability of 0.5 is used. In order to simplify the calculation of the conditional probability, assumption of Markov chain is employed [14].

Definition 3.3. A Markov chain is a representation of a finite-state Markov process, a stochastic model where the transition probability distributions at any time depend only on the present state and not on how the process has reached in that state. The Markov chain model for the STG is a directed graph isomorphic to the STG and with weighted edges.

Symbolically the parameter can be expressed as:

$$p_{ij} = Prob(Next = s_j | Present = s_i) = \frac{N_{ij}}{\sum_k N_{ik}} \tag{3.1}$$

Here, $i, j = 0, 1, 2, ..., n-1$, N_{ij} is the number of transitions ts_{ij} from S_i to S_j while $\sum_k N_{ik}$ is all transitions that begin in state S_i.

Given an FSM, calculation of the conditional state transition probability p_{ij} is straightforward assuming uncorrelated and equiprobable inputs for simplicity [15]. The following example shows how to calculate p_{ij} given input signal probabilities

Example 3.1. Consider the FSM shown in Fig. 3.1(a) with one input, In, and two outputs. Assume that the input probabilities are $Prob(In = 1) = 0.5$. The conditional transition probabilities are labelled on the edges of the Markov chain. For instance, consider the transition ts_{14}. Its CSTP is $p_{14} = Prob(In = 1) = 0.5$. The detail is shown in Fig. 3.2.

This process is simple if the inputs of the FSM are completely specified as in Example 3.1. However, if the inputs are incompletely specified, the cases of overlapping inputs and impossible inputs need to be dealt with, which is shown in the following example.

Example 3.2. Table 3.1 shows some lines of an incompletely specified FSM tav, which is one of MCNC benchmark circuits and given in the kiss file.

The first four lines tell the number of inputs, outputs, product terms and states, respectively. The machine has four inputs, four outputs, 49 product terms and four states. From the fifth line, each row consists of four sections, which give input encoding, present state, next state and output encoding. For the transition ts_{01}, the maximum number of the transition is sixteen. It can be seen that the input cases {0111, 1011, 1101, 1110, 1111} have been over calculated for the incompletely specified inputs. Thereby, the overlapping inputs need to be checked, which can be expressed into

$$I_l \bigcap I_r \neq \Theta \tag{3.2}$$

Here, I_l, I_r are input vectors resulting in the same transition ts_{ij}, $l \neq r$ are the line number of the circuit description file and Θ is an empty set. Algorithm 3.1 outlines the procedure used

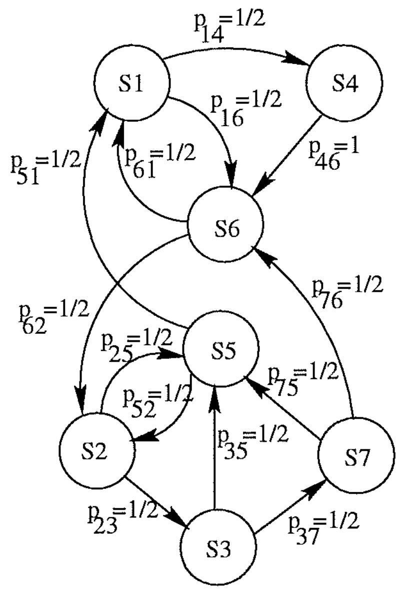


Figure 3.2: Conditional state transition probability of DK27

to calculate p_{ij}.

Algorithm 3.1. *Procedure for finding CSTP*

```
==========================================================
Find_ CSTP($s_i$,$s_j$)

        {
            $p_{ij} = \sum p(I_l)$;
            for ($l = 4$; $l < MaximumLineNumber - 4$; $l++$) //$MaximumLineNumber$
stands for the maximum line number of the circuit description file
                for($r = l + 1$; $r < MaximumLinNum - 4$; $r++$)
                {
                    if $I_l \cap I_{l-1} \neq \Phi$
                    {
                        $p_{ij} = p_{ij} - \sum_{I \in I_l \bigcap I_{l-1}} p(I)$;
                    }
                }
```

Table 3.1: Tav's kiss format

.i 4			
.o 4			
.p 49			
.s 4			
1000	s_0	s_1	1000
0100	s_0	s_1	0100
0010	s_0	s_1	0010
0001	s_0	s_1	0001
0000	s_0	s_1	0000
11–	s_0	s_1	0000
1-1-	s_0	s_1	0000
1–1	s_0	s_1	0000
-11-	s_0	s_1	0000
-1-1	s_0	s_1	0000
–11	s_0	s_1	0000
1000	s_1	s_2	1000
⋮	⋮	⋮	⋮

The CSTP is external input information. It does not depend on the structure of the Markov chain. For an FSM, this information can be assumed to be known. Some researchers used this information as a rough approximation to the transition probabilities [88]. However, this probability does not utilize the information of state probabilities, where the state probability, P_i, represents the probability that the machine is in a given state s_i.

Definition 3.4. *State probability* P_i of a state s_i, which is defined as the probability that the state is visited in an arbitrarily long random sequence, can be obtained by solving the corresponding Chapman-Kolmogorov equations and the normality condition equation in equation 3.3.

$$\begin{cases} \sum_{i=0}^{i=n-1} P_i = 1 \\ P_i = \sum_{j=0}^{j=n-1} P_j p_{ji} \end{cases} \tag{3.3}$$

By solving the above set of linear equations, the state probability P_i can be obtained, which is shown on the edges of the Markov chain as in Fig. 3.3.

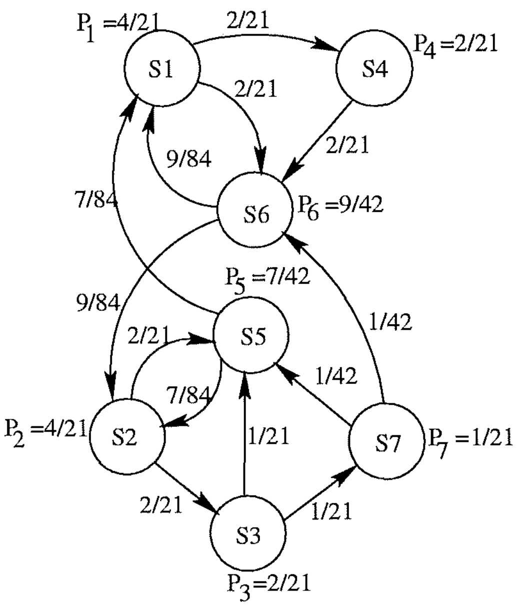


Figure 3.3: State probability labelled beside its state node and state transition probability labelled on each edge of DK27

The well-known Gaussian elimination method is used to solve the above equations[119]. The core of the method is to convert a matrix into an upper triangular form and solve for P_i in $AP = b$ using the back-substitution method. Here, A is an upper triangular matrix, P is state probability vector and b is a constant vector based on equations 3.3. Algorithms 3.2 and 3.3 outline the procedures.

Definition 3.5. An upper triangular matrix is one in which all elements below the main diagonal line are zero as follows

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0(n-1)} \\ 0 & a_{11} & a_{12} & \cdots & a_{1(n-1)} \\ 0 & 0 & a_{22} & \cdots & a_{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{(n-1)(n-1)} \end{pmatrix}$$

Algorithm 3.2. *Convert a matrix into an upper triangular matrix*

```
==============================================================
UpperTriangular(s) {
        for ($i$; ; ;) {
            if ($A(i,i) == 0$) {
              sort(A); }//sort the matrix so that the diagonal element is not zero
             else
               $pivot = A[i][i]$;
            for($j = i + 1$; ; ) {
               $mult = A[j][i]/pivot$;
               $A[j][i] = 0$;
               for($k = i + 1$; ; ) {
                   $A[j][k] = A[j][k] - mult * A[i][k]$;}
              $b[j] = b[j] - mult * b[k]$;}}}
```

Algorithm 3.3. *Back substitution*

```
==============================================
Backsubstitution(){
                    for($i$; ; ;) {
                        for($j + i + 1$; ; )
                            $sum = sum + A[i][j] * b[j]$;
                        $b[i] = (b[i] - sum)/A[i][i]$;}}//The answer is returned by $b$
```

State transition probability can be defined as follows.

Definition 3.6. *State transition probability* (STP) tp_{ij} between two states s_i to s_j occurs in an arbitrarily long sequence and is given by

$$tp_{ij} = P_i p_{ij} \tag{3.4}$$

The STPs of DK27 are shown in Fig. 3.3.

Definition 3.7. The *switching activity* of the state bit lines depends on the state encoding and the state transition probabilities. *Average switching activity* of an FSM can be calculated as follows:

$$SA = \sum_{i=0}^{i=n-1} \sum_{j=0}^{j=n-1} tp_{ij} \times HD(enc(s_i), enc(s_j)) \tag{3.5}$$

where $enc(s_i)$ is encoding of state s_i and $HD(enc(s_i),\ enc(s_j))$ is the Hamming distance between two encodings, $enc(s_i)$ and $enc(s_j)$. From equation 3.5, for an FSM with given input signal probabilities, $\{tp_{ij}\}$ is fixed while $HD(enc(s_i), enc(s_j))$ varies with different state assignments. Thereby, SA varies with different state assignments. This is the basic principle to optimize power dissipation using state assignment.

3.3 Cost functions

The implementation of an FSM consists of two parts: a combinational logic section and a register section. Both sections contribute to the power dissipation of an FSM. Traditionally, switching activity is used to measure the power dissipation of a circuit. Some researchers [48][117] suggested the use of switching activity as cost function. The approach implements the low power dissipation by minimizing equation 3.5. However, this only minimizes the switching activity on present state bit lines of the machine and does not consider the structure of the combinational section of the final synthesized FSM, which may lead to non-optimal area implementation and result in power overhead in the combinational section. Hence, to obtain low power dissipation in the final circuit, area should be taken into account.

In [15], the cost function is linear composition of the weighted area and the switching activity. Fan-in- or fanout-oriented method [33] based weighted area was used to indicate the desired Hamming distance between the state codes. The higher the weight is, the smaller the Hamming distance will be. A parameter $\alpha \leq 1$ was introduced, specifying the relative importance of switching activity with respect to area constraints. It was defined as

$$Cost = (1-\alpha)SA + \alpha \sum_{i=0}^{i=n-1} \sum_{j=0}^{j=n-1} \omega_{ij} \tag{3.6}$$

Here, ω_{ij} is weighted area between S_i and S_j.

In [77], literal based area was employed and the cost function is normalized to 1000.

$$Cost = \Delta_{literals} + \beta \times SA \tag{3.7}$$

There, β was taken as 100.

The cost function is one of very important factors to guide circuit optimization. In this work, two kinds of cost functions will be used and described as follows.

In Section 3.5, area and SA versus α and β will be studied. However, the number of cubes after the logic minimization of a machine is used instead of weighted area and literals used in equations 3.6 and 3.7. It will be shown that if the cost function is normalized to 1, SA is very sensitive to change in α while area is less sensitive (See Fig. 3.4(a)). Further, if the cost function is normalized to 1000, area is very sensitive to change in β while SA is less sensitive (See Fig. 3.4(b)). To influence both SA and area, a factor of 10 is used here and the first cost function is defined as in equation 3.8 [124].

$$Cost_a = noOfCubes + SA \tag{3.8}$$

Here, $noOfCubes$ is the number of cubes after the logic minimization of a machine.

On the other hand, for CMOS circuits, the dominant source of power dissipation is the charging and discharging of the node capacitance and is given by:

$$P_{aver} = \frac{1}{2}V_{dd}^2 f_{clk} \sum_{k=1}^{k=m} C_k SA_k \tag{3.9}$$

Where P_{aver} is the average power dissipation of the circuit, V_{dd} is the supply voltage, C_k is the capacitive load at the output of gate k, f_{clk} is the clock frequency, SA_k is the switching activity of gate k and m is the number of gates in a circuit and $k \in \{1, 2, \cdots m\}$. All of the parameters in the above equation can be determined from technology or circuit layout information except SA_k and C_k of the circuit which depend on the synthesis technology. Hence, equation 3.9 can be rewritten as in equation 3.10.

$$P_{aver} = K \sum_{k=1}^{k=m} C_k SA_k = KSA \sum_{k=1}^{k=m} C_k \tag{3.10}$$

Here, $K = \frac{1}{2}V_{dd}^2 f_{clk}$ while SA is the average switching activity of a circuit and defined as $SA = \frac{1}{m}\sum_{k=1}^{k=m} SA_k$.

Hence, we have $P_{aver} \propto Area \times SA$, where $Area$ is estimate of the circuit area that is representative of the capacitance $\sum_{k=1}^{k=m} C_k$ [75].

Given an STT of an FSM and a specific state encoding, SA can be calculated by equation 3.5. Under the circumstance of employing the minimal encoding length, the number of memory elements used in a specific FSM are constant so that their corresponding area is fixed. Hence, only the area of the combinational section varies with the state assignment. The area of the combinational section could be measured by the number of cubes. Therefore, the second cost function is defined as in equation 3.11.

$$Cost_b = noOfCubes \times SA \tag{3.11}$$

3.4 State assignment using Genetic algorithm

Genetic Algorithms (GAs) are based upon evolutionary adaptation in natural systems and attempt to generate useful solutions to a given problem by the application of the "survival of the fittest" principal. The basic idea is to create a population (breeding pool) of potential solutions to a problem. These solutions are encoded as "chromosomes", and each chromosome is subjected to an evaluation function which assigns "fitness" depending upon how well the solution it encodes solves the problem at hand. Existing solutions are recombined by a process called crossover or breeding. Further, a mutation process makes random changes in a few randomly selected genes. This prevents premature convergence by maintaining the diversity of the population. The best members of the population may fail to produce offspring in the next generation. The elitist strategy rectifiers this potential loss by copying the best members of each generation into the succeeding one.

3.4.1 Solution representation

An FSM with n states requires a minimum of s state variables for the assignment where $s = \lceil log_2 m \rceil$ and symbol $\lceil\rceil$ stands for taking the upper bound integer of $log_2 m$. The chromosome representation is a string of decimals. Take benchmark 'DK27' for example. It is a seven state machine and the states are named s_1, s_2, s_3, s_4, s_5, s_6, s_7. Assignment 6, 2, 1, 4, 3, 5, 7 is one possible chromosome. Then, s_1 is assigned the binary code 110, s_2 is assigned 010, etc. The population of solutions consists of state assignments of an FSM. They will be stored in a two dimensional array *population*[*i*][*j*], where, (*i*=0,1,2, ..., *populationSize*-1, *j*=0,1,2, ..., *numberOfStates*-1). The breeding pool of chromosomes is initially created by *InitialPopulation* () randomly. The sketch of *InitialPopulation* () is as shown in Algorithm 3.4:

Algorithm 3.4. *Initializing population*

```
====================================================
InitialPopulation (int numberOfBits, int numberOfStates )

{
   int maxInt, randNum, randRemain;
  int doubleEentryFlag = 0;
   maxInt = 1 << numberOfBits;
   for(i=0;i<populationSize;i++)
        for(j=0;j<numberOfStates;j++)
             population[i][j] = maxInt +1;
srand (seedValue);
for(i=0;i<populationSize;i++)
     for(j=0;j<numberOfStates;j++)
      {
         do
            {
               randNum = rand();
               randRemain = randNum%power;
               for(k=0;k<numberOfStates;k++)
                   if(randRemain==population[i][k])
                       doubleEntryFlag = 1;
           }while(doubleEntryFlag==1);
```

```
            population[i][j] = randRemain;
        }
    }
```

3.4.2 Evaluation

Based on the above assignment, the cost can be calculated according to equations 3.8 & 3.11. To a specific state assignment for an FSM, let the maximum state transition probability

$$tp = max(tp_{ij}) \tag{3.12}$$

and maximum Hamming distance between two state codes

$$maxHD = max(HD(enc(S_i), (S_j))) = lengthOfStateCode \tag{3.13}$$

The maximum SA of the FSM can be calculated as:

$$maxSA = \sum_{i=0}^{i=n-1} \sum_{j=0}^{j=n-1} tp \times maxHD \tag{3.14}$$

The area can be estimated by the number of cubes after state assignment. $maxCubes$ takes the product terms from the STT. Then the maximum costs will be:

$maxCost_a = maxSA + maxCubes$ and $maxCost_b = maxSA \times maxCubes$.

Hence, to a specific state assignment, the fitness can be defined as:

$$fitness_{mode} = (maxCost_{mode} - cost_{mode})/maxC_{mode} \tag{3.15}$$

Here, *mode* $\in \{a, b\}$, which corresponds to two cost models in equations 3.8 & 3.11, respectively. From equation 3.15, lower cost will result in higher fitness.

3.4.3 Crossover

Parent selection strategy follows the one proposed in [9]. Crossover is a primary method of perturbations in GA, which generates better solutions by exchanging the information contained in the present solutions. In this application, a position-based crossover (PBX) is employed [130]. This was modified, however, so that invalid offsprings are avoided and efficient crossover is reached. This is outlined as follows.

Randomly select a number of locations named location-indicator (LID) in binary codes whose length is equal to the number of states in the FSM. Where 1s appear in the LID, copy the states from Parent 1 to Child 1. Where there are 0s in the LID, copy those corresponding states from Parent 2 provided that they do not exist in Child 1. If the state from Parent 2 is already in Child 1, the position is filled by the first unassigned state from Parent 1 but checks are made to avoid duplicating the parent. Continuing this way, Child 1 is obtained. With the same LID, interchange Parent 2 and Parent 1 and repeat the process to produce Child 2. This is illustrated for a seven state machine.

Step 1. Randomly generate a seven-bit LID: 1 0 1 1 0 1 0

Step 2. Select two parents by the roulette wheel approach:

Parent 1: 6 2 1 4 3 5 7

Parent 2: 1 0 5 3 2 4 6

Step 3. Generate Child 1:

a. Where 1s appear in the LID, copy the states from Parent 1 to Child 1 and delete those copied states in Parent1.

Parent 1: - 2 - - 3 - 7; Child 1: 6 - 1 4 - 5 -;

b. Where there are 0s in the LID, copy the states from Parent 2 to Child 1 provided that they do not exist in Child 1 and delete the copied states in Parent 2. There are states 0, 2 and 6 from Parent 2 corresponding to 0s in the LID. However, state 6 is already in Child 1. Hence, only states 0 and 2 in Parent 2 are copied to Child 1.

Parent 2: 1 - 5 3 - 4 6; Child 1: 6 0 1 4 2 5 -;

c. The unfilled position in Child 1 is filled by the first unassigned state from Parent 1, which is state 3. Then state 3 is copied to Child 1.

Parent 1: - 2 - - - - 7; Child 1: 6 0 1 4 2 5 3;

Step 4: Swap Parent 1 and Parent 2 and generate Child 2 following the same procedure.

3.4.4 Mutation

Mutation in normal GA just flips a selected bit from 0 to 1 or vice verse. We mutate CHROMOSOMES state number one by one with a mutation rate 6%. The procedure is as follows:

For example, for a 7-state FSM, if the chromosome is 6 2 1 4 3 5 7, the following procedure will be executed. At first, the program will randomly generate a number, namely *randomNum*1, and check the first state 6, and if *randomNum*1 $<$ 6%, produce two different random numbers *randomNum*2 and *randomNum*3, for instance, *randomNum*2 $=$ 2 and *randomNum*3 $=$ 4, then, the state numbers in the third bit and fifth bit of the chromosome will be swapped and the chromosome would become 6 2 3 4 1 5 7. Then it will check the second state, if *randomNum*1 $>$ 6%, no exchange takes place in the chromosome during this step. The program will check the generations one by one until the last generation. The detail is shown in Algorithm 3.5.

Algorithm 3.5. *Mutation operator*

```
==================================================
Mutation(numberOfStates)
{
    for(i=0;i<numberOfStates;i++)
      {
        randomNum1 = rand()%100;
        if(randomNum1 < MUTATION_RATE)
          {
            randomNum2 = rand()%(numberOfStates);
            do
            {
                randomNum3 = rand()%numberOfStates;
            }while (randomNum2 !=randomNum3);
swap(population[randomNum1][randomNum2], population[randomNum1][randomNum3]);
        }
    }
}
```

3.4.5 Elitism

The best member of the population may fail to produce offspring in the next generation. Hence, it will be kept in the array $eletistMember[i]$ and stored in the array $population[populationSize/2-1][i]$ of the next generation, where $i = 0,\ 1\,, 2,\ ...,\ numberOfstates - 1$.

3.4.6 Outline of the algorithm

The outline of the algorithm is illustrated in Algorithm 3.6.

Algorithm 3.6. *Outline of the algorithm for state assignment*

```
==========================================================
Step 1. Read the benchmark
Step 2. Generate an initial population-brooding pool
Step 3. Calculate state transition possibilities $tp_{ij}$
Step 4. Assign states and create a Berkeley standard PLA file for each chromosome
Step 5 Calculate the number of cubes by ESPRESSO minimization for each chromosome
Step 6. Calculate the fitness of each chromosome
Step 7. Select parents by the roulette wheel approach.
Step 8. Perform crossover to generate offsprings
Step 9. Mutate chromosomes
Step 10. Assign states and create a Berkeley standard PLA file for each chromosome
Step 11. Calculate the number of cubes by ESPRESSO minimization
Step 12. Calculate the fitness of each chromosome
Step 13. Generations = generations + 1
Step 14. If generations≤max_numberGenerations, goto Step 7.
Step 15. Output the results:
(1) The half chromosomes with higher fitness
(2) The best state assignment, its switching activity and the number of cubes
```

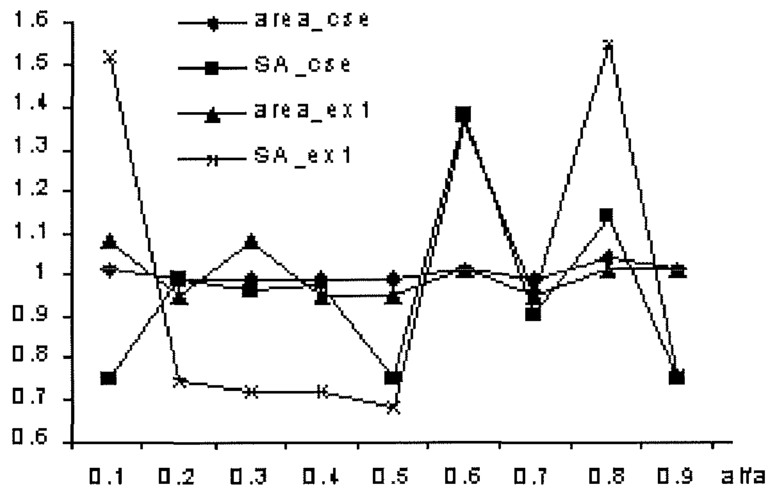
3.5 Experimental results

The above algorithm is implemented in C and applied to MCNC benchmark circuits. Gaussian elimination method [119] is used to find the total transition probabilities according to the STT of an FSM. ESPRESSO is used to minimize the circuit after state assignment and obtain the cubes. This is done by generating a Berkeley standard PLA file for each chromosome and passing it to ESPRESSO for minimization. The product terms from this minimization determine the number of cubes for that assignment. Switching activity is calculated using equation 3.5. Two sets of experiments have been conducted.

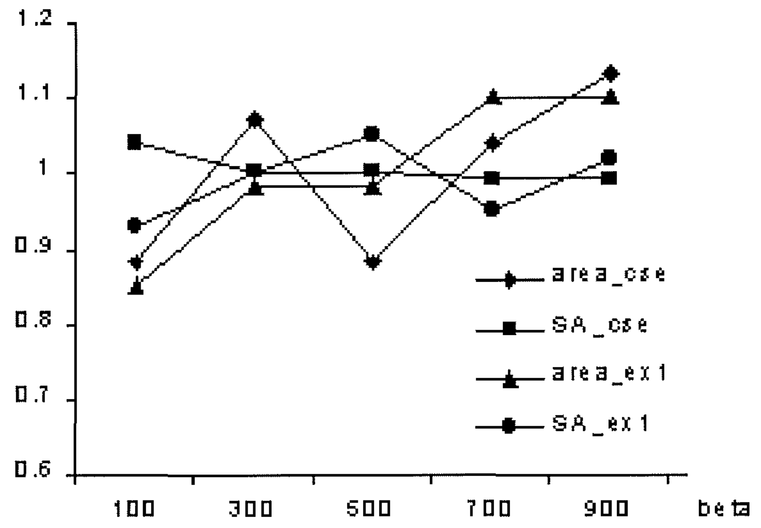
First, we want to find out how the combination of area and SA, two cost functions in equations 3.6 and 3.7, affect the optimization quality. In order to compare them, we use $noOfCubes$ instead of ω in equation 3.6 and $\Delta_{literals}$ in equation 3.7, respectively. Benchmark circuits 'ex4' and 'cse' are chosen as study cases. The reason for choosing them is that 'ex4' has 14 states and is an incompletely specified machine while 'cse' has 16 states and is a completely specified machine. Both are medium size FSMs. Figs. 3.4(a) and (b) show that area and SA vary with α and β, respectively. The plots show:

- The area ratio $A_{noOfCubes}/A_{averageOfCubes}$
- The SA ratio $SA/SA_{average}$

From Fig. 3.4(a), we notice that area varies much less than SA when α varies from 0.1 to 0.9. From Fig. 3.4(b), SA varies much less than area when β varies from 100 to 900. It makes sense that equation 3.6 based cost function does not have too much influence on area minimization while equation 3.7 based cost function has little influence on SA minimization.



(a)



(b)

Figure 3.4: Relationship between cost function and optimization quality (a) Area and SA versus α in equation 3.6; (b) Area and SA versus β in equation 3.7

Second, fourteen MCNC benchmark circuits, whose number of states are between 10 and

48, are chosen to test the proposed methods. Then an area-oriented state assignment program, NOVA [111], is run on the same benchmarks, following the same procedure to get the switching activity and the number of cubes. Tables 3.2 summarizes the test results. The results given by Hong et al and Wang et al are listed in the same table for comparison. In Table 3.2, $cubes_a$ and SA_a are obtained using $cost_a$ in equation 3.8 while $cubes_b$ and SA_b are obtained from $cost_b$ in equation 3.11. Compared to NOVA, Hong and Wang gave 47.5% and 42.3% reduction in switching activity while 10.3% and 12.6% extra product terms are used, respectively. Using $cost_a$, GA gives a 32.2% reduction in switching activity and 4.4% reduction of product terms. Among the 14 circuits, our method gives better area results in 11 cases compared to NOVA and in 10 cases compared to Hong's and Wang's. Using $cost_b$, GA gives a 43.8% reduction in switching activity but needs 5.7% extra product terms. The area penalty, however, is much smaller than Hong's and Wang's. Table 3.3 gives the two set of codes, namely a and b, for the best state assignment by GA, which are based on $cost_a$ and $cost_b$, respectively.

The above results are obtained using the following parameters: population size = 70, crossover rate = 60%, mutation rate = 6%, and maximum number of generations = 200. These parameters were determined after testing various population sizes and different crossover and mutation rates.

3.6 Summary

A genetic algorithm for finding good state assignments targeting minimization of power and area for finite state machines has been developed and implemented in C. Tests as to the effectiveness of this approach to the problem are conducted by comparison of performance against the state-of-the-art commercially available software and some published results when operating upon MCNC FSM benchmark circuits. Two options are available for optimizing FSMs targeting area or power dissipation. The results shows that significant saving in power and/or area can be achieved.

Table 3.2: Experimental results showing power & area improvement

benchmarks		GA				Wang [117]		Hong [48]		NOVA [111]	
Name	States	$cubes_a$	SA_a	$cubes_b$	SA_b	cubes	SA	cubes	SA	cubes	SA
bbara	10	22	0.317	24	0.279	26	0.279	26	0.295	24	0.495
bbsse	16	27	0.783	28	0.788	31	0.776	31	0.856	30	1.500
cse	16	43	0.355	44	0.252	48	0.239	50	0.292	46	0.604
downfile	24	36	1.600	47	1.438	45	1.125	40	1.083	28	1.750
keyb	19	46	0.674	48	0.573	58	0.556	52	0.647	48	1.466
modulo12	12	12	0.583	11	0.583	12	0.500	12	0.583	12	1.000
planet	48	86	2.424	92	1.682	103	0.984	101	1.153	87	2.831
s1	20	66	1.480	76	1.184	91	1.175	85	1.131	80	1.698
sand	32	89	0.765	101	0.664	109	0.610	110	0.604	97	1.085
styr	30	88	0.943	98	0.586	99	0.553	101	0.578	94	1.278
ex1	20	52	0.842	59	0.750	47	1.135	49	0.755	44	1.358
ex4	14	14	0.421	16	0.467	18	0.957	16	0.495	19	1.316
opus	10	15	0.556	16	0.417	17	0.712	17	0.524	16	0.812
train11	11	10	0.339	10	0.339	10	0.714	9	0.360	9	0.619
Total		606	12.082	670	10.002	714	10.315	699	9.356	634	17.812
aver.%red.		4.4	32.2	-5.7	43.8	-12.6	42.3	-10.3	47.5	0	0

Table 3.3: Best state assignment codes by GA: Assignments a & b are based on $cost_a$ & $cost_b$ respectively

	a: 8, 0, 2, 7, 4, 6, 15, 14, 10, 12 ; b: 8, 0, 4, 6, 2, 3, 7, 14, 12, 10
bbsse	a: 8, 2, 6, 14, 3, 10, 15, 13, 9, 12, 11, 0, 4, 1, 7, 5
	b: 4, 14, 10, 3, 6, 2, 9, 1, 8, 11, 0, 12, 13, 15, 5, 7
cse	a: 0, 1, 13, 5, 15, 6, 2, 3, 9, 7, 11, 10, 8, 14, 4, 12
	b: 0, 4, 12, 6, 14, 11, 8, 10, 1, 2, 3, 7, 5, 15, 9, 13
donfile	a: 2, 3, 0, 4, 1, 5, 6, 7, 10, 11, 8, 9, 13, 12, 14, 15, 21, 23, 17, 16, 18, 19, 25, 27
	b: 11, 5, 12, 28, 1, 8, 15, 7, 14, 26, 31, 29, 13, 20, 6, 30, 0, 4, 3, 24, 23, 21, 16, 22
keyb	a: 0, 4, 16, 8, 21, 12, 20, 19, 7, 17, 14, 1, 15, 23, 3, 9, 28, 10, 13
	b: 0, 4, 2, 8, 31, 10, 27, 25, 24, 17, 15, 3, 13, 29, 9, 28, 16, 21, 5
modulo12	a: 0, 9, 13, 15, 11, 3, 7, 5, 4, 6, 14, 12; b: 9, 8, 10, 11, 15, 7, 0, 2, 3, 1, 5, 13
planet	a: 26, 11, 7, 0, 4, 10, 18, 24, 22, 15, 48, 43, 59, 14, 9, 12, 46, 44, 8, 63, 51,6, 5, 13, 47, 34, 23, 52, 2, 1, 58, 45, 61, 37, 25, 35, 42, 32, 31, 60, 39, 56, 50, 30, 57, 49, 62, 21
	b: 50, 36, 52, 20, 16, 2, 14, 6, 54, 53, 46, 49, 40, 51, 9, 11, 3, 19, 25, 56, 57, 35, 7, 32, 24, 28, 29, 21, 8, 33, 12, 45, 5, 37, 13, 39, 61, 63, 31, 23, 15, 62, 58, 18, 22, 4, 48, 26
s1	a: 20, 18, 10, 26, 5, 27, 16, 24, 0, 8, 30, 25, 29, 13, 22, 1, 19, 9, 4, 21
	b: 13, 5, 28, 29, 10, 31, 1, 21, 0, 17, 19, 23, 14, 15, 16, 2, 20, 7, 18, 8
sand	a: 22, 7, 6, 2, 3, 30, 31, 21, 0, 16, 19, 17, 23, 5, 12, 13, 14, 15, 10, 11, 8, 9, 4, 25, 28, 27, 24, 1, 20, 18, 29, 26
	b: 18, 16, 1, 17, 9, 26, 24, 8, 0, 4, 25, 12, 19, 14, 6, 22, 20, 21, 5, 13, 15, 23, 7, 3, 2, 10, 11, 27, 31, 29, 28, 30
styr	a: 0, 8, 13, 24, 28, 29, 9, 5, 22, 4, 11, 19, 2, 18, 16, 26, 7, 12, 14, 20, 15, 3, 27, 17, 21, 31, 23, 25, 6, 1
	b: 9, 25, 24, 10, 4, 28, 26, 30, 27, 19, 13, 29, 1, 0, 11, 12, 3, 18, 22, 6, 23, 2, 21, 16, 8, 31, 5, 20, 17, 14
ex1	a: 0, 4, 2, 16, 12, 18, 6, 15, 26, 11, 23, 19, 27, 30, 3, 31, 7, 5, 22, 1
	b: 0, 1, 4, 8, 3, 5, 12, 11, 23, 7, 28, 22, 18, 15, 19, 14, 21, 6, 17, 2
ex4	a: 11, 3, 6, 15, 1, 4, 0, 9, 12, 14, 8, 10, 13, 5; b: 4, 2, 1, 3, 6, 13, 0, 5, 15, 7, 12, 10, 11, 9
opus	a: 6, 2, 11, 0, 15, 8, 10, 4, 5, 1; b: 9, 11, 15, 0, 7, 8, 12, 2, 1, 10
train11	a: 0, 1, 2 ,3, 7, 9, 8, 10, 14, 6, 4; b: 0, 1, 2, 3, 7, 9, 8, 10, 14, 6, 4

Chapter 4

Differential CMOS single edge-triggered flip-flop

Flip-flops are the basic building blocks of synchronous digital circuits and, to a large extent, determine circuit power dissipation. Hence, low power design techniques of flip-flops are essential to design low power sequential systems. As a consequence, many genius techniques have been recently proposed to reduce the power dissipation of the flip-flops [49][19][68][102][100][54][72]. Categorized by the input sampling, edge-triggered flip-flops can be grouped into two types: pulse sampling [102] and level sampling [100]. The former only needs one latch and hence has a simpler structure, however, it suffers from long clock chain used to generate narrow pulses. The latter needs two latches or flip-flops and hence needs more transistor count, which results in the increase of its area and power dissipation [102]. Structurally, they can be categorized into differential and non-differential flip-flops. The structure characteristic of differential flip-flops is that they are composed of one differential amplifier, flip-flop cell, and some auxiliary circuits. Hence, differential flip-flop has some advantages over the non-differential one since it has complementary outputs, can amplify a small voltage signal and at the same time latch data. Much effort has been paid to improve its performance [54][59][72]. In [54], reduced clock-swing flip-flop is proposed and power saving is achieved by lowering the voltage swing of the clock system. However, a backgate bias is needed, which complicates the power supply. In [100], a flip-flop design with clock-gating on both master and slave latches is presented. However, the area penalty for the additional circuit is very high and the power saving is significant only when the switching activity of the input signal is very low. Looking at the published circuits, their designs suffer from two disadvantages.

1. For the version clocked with NMOS and PMOS transistors in flip-flop cell, clock load is heavy and static power dissipation is significant if reduced-swing clock singles are applied to PMOS transistors [54][59].

2. For the version clocked with NMOS transistors in flip-flop cell, it suffers from long clock chain, which consumes significant clock power[72].

4.1 Differential CMOS single edge-triggered flip-flop

In [59], two $N - C^2$MOS output latches were used to improve the speed of flip-flops but this is traded with power dissipation penalty. There are two disadvantages for this design. One is that the clock load is too heavy because it uses many clocking transistors. The other is difficult to apply for reducing the signal swing due to clocking with NMOS and PMOS transistors. To overcome these problems, in [72], a differential CMOS Single Edge-Triggered Flip-Flop (SET-FF) based on clock racing was proposed.

The flip-flop in Fig. 4.1 consists of a differential amplifier called a cell and a clock chain which is composed of three inverters [72]. The clock chain generates a delayed CP_1. CP and CP_1 are applied to the cascaded transistors, m7 and m8, respectively, which generate a narrow pulse at the rising edge of each clock pulse. Hence, the flip-flop is triggered at the rising edge of the clock pulse. Transistor sizes, which are expressed in W/L, are marked beside the inverters. To obtain proper width of the narrow pulse, transistor sizes, W and L, of the second inverter in the clock chain are the double of the other inverters. If D, Q and Q' are used to represent the input, present state and next state signals respectively, the next state equation can be expressed into

$$Q' = D\alpha_{cp} + Q\overline{\alpha_{cp}} \tag{4.1}$$

Here, α_{cp} represents the rising edge of clock pulse.

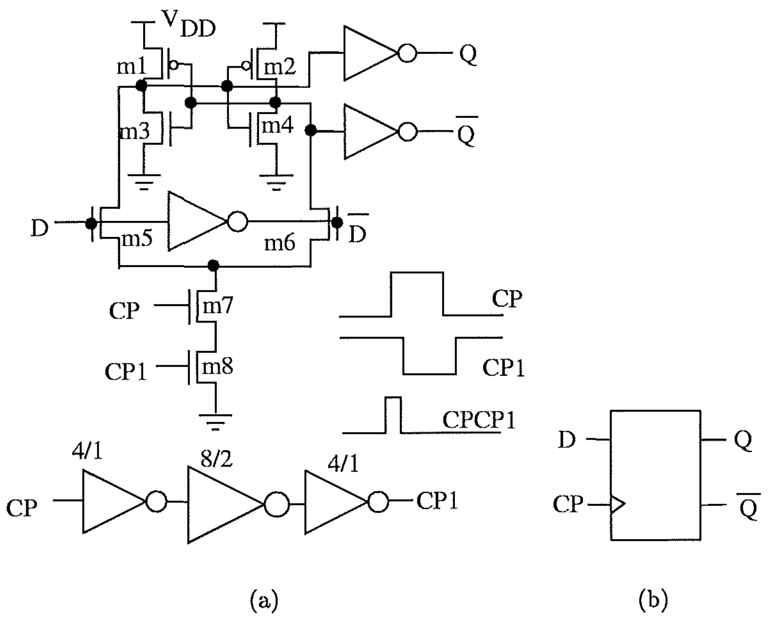


Figure 4.1: Differential CMOS SET-FF (a) Circuit implementation; (b) Logic symbol

Because the cell is driven by a narrow pulse, the output latches can be removed and only two NMOS transistors are clocked. Hence, this design has a simpler structure and a feature to apply for data and clock signals with reducing swing. The experimental results show that it has better performance in power and speed than those used in commercial processors. On the other hand, the advantage of edge-triggered flip-flops is that the setup time for data input is independent of the clock pulse width. It is also less sensitive to noises. This makes system design simpler.

4.2 Circuit parameter optimization

For a flip-flop, speed and power dissipation are two key parameters. However, they mainly depend on the circuit structure and technology parameters. For a specific circuit structure, power delay product (PDP) is used to measure its performance. Therefore, circuit optimization aims to optimize PDP.

Definition 4.1. Power Delay Product (PDP) is defined by the following equation:

$$PDP = T_{delay} * P_w \tag{4.2}$$

Here, T_{delay} is the delay of a flip-flop while P_w is its power dissipation.

4.2.1 Delay

For the clarification, it is helpful to give some definitions here.

Definition 4.2. D_{CQ} : Clock-to-Q time, the propagation delay from the C to the Q, assuming that the D signal has been set early enough relative to the triggering edge of the clock pulse. Here C refers to the Clock.

Traditionally, the delay of flip-flops is referred to Clock-to-Q delay. However, some authors pointed out that using D_{CQ} delay as a relevant performance parameter for a flip-flop is misleading because D_{CQ} delay does not take the setup time into account [1][99].

Definition 4.3. T_S : Setup time, the minimum time between a D change and the triggering edge of the clock pulse such that the output Q will be guaranteed to change so as to become equal to the new D value.

The setup time is the delay between the data input of the flip-flop and the storage element as the data takes a finite time to travel to the storage point.

Thereby, it is proposed to use D-to-Q as the delay of a flip-flop. This is defined as follows:

$$T_{DQ} = D_{CQ} + T_S \tag{4.3}$$

In this thesis, we will accept both D_{CQ} and T_{DQ} to measure the delay of a circuit.

For the measurement of T_{DQ}, the methodology proposed in [1] is followed.

4.2.2 Circuit optimization

For a high-performance and low-power application, speed and power are equally important. Usually, there is tradeoff between power and speed. To compromise the two, the concept of the minimum power-delay product (minPDP) is used. The minimum power-delay product is the optimal energy utilization at a given clock frequency.

Take Fig. 4.1 for example to explain this. Fig. 4.2 is the same version of Fig. 4.1. The main difference is that all inverters are substituted by nMOS and pMOS transistors. However, it is not easy to optimize PDP because theoretically PDP is a multiple variable function of circuit parameters. Given the variety of designs, it is not always simple to express the PDP as a function of one common variable. For simplification, we take the transistor channel width as a common variable. Then, delay, power and PDP can be expressed into functions of one variable. Set the minimum transistor length to 1μ and channel width is marked beside the transistors using w. Some transistors are pre-optimized based on the parameters referred to in reference [72].

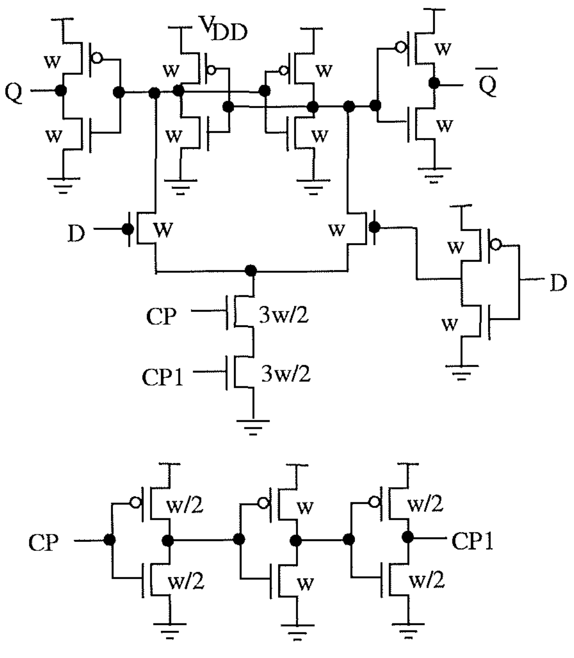


Figure 4.2: Power delay product optimization

Fig. 4.3 shows the linear relationship between the power and transistor width w while Fig. 4.4 indicates the nearly inversely proportional relationship of delay versus width w. Fig. 4.5 shows the curve of PDP versus width w, which shows that there is a point of the optimum power-delay tradeoff. Based on this analysis, the circuit parameters can be optimized to the point of minimal PDP.

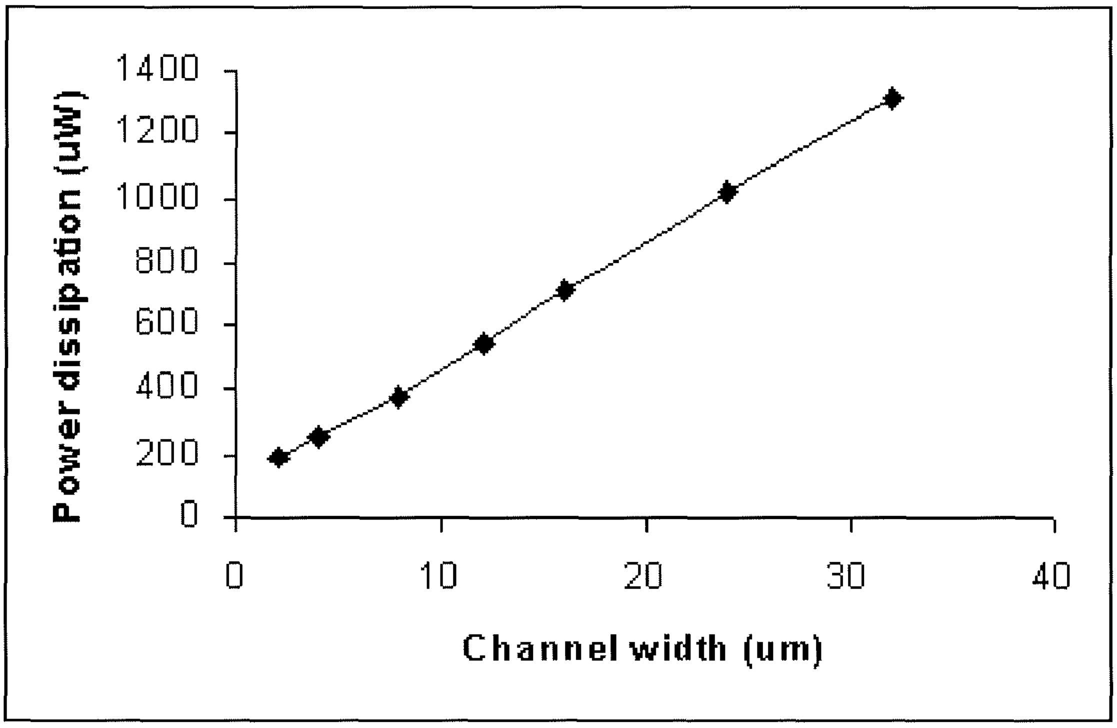


Figure 4.3: Power versus width w

The procedure presented above is used to optimize circuits. Although the procedure could be developed into an automatic tool, it is not our main point here. Further details can be found in reference [99].

4.3 Differential CMOS Single Edge-Triggered Flip-Flop with Clock-Gating (CG-SETFF)

As known, clock signal is the most frequent transition signal in the circuit. In one clock cycle, clock signal has two transitions while other signals in the circuit (exclude glitches) have one transition at most. Hence, in the clock chain, transition is the most frequent and correspondingly it consumes significant power. According to the working principle of D flip-flop, next state Q' follows the input signal D. If D does not change, the flip-flop does not need to be triggered. If we can detect the idle conditions of input signal D and stop the clock signal, then power dissipation can be saved. Based on this idea, a new design is proposed in this section.

In Fig. 4.1, it is observed that whether the input signal D varies or not, there is always a narrow pulse to trigger the flip-flop. However, when the input signal D is in low switching activity, it is unnecessary to trigger the flip-flop for each clock pulse. For example, when D keeps at 1, the flip-flop does not need to be triggered. In this case, those narrow pulses to the flip-flop are redundant and the transitions that happen in the clock chain are also redundant. If the clock of the clock chain can be deactivated, power can be saved. Hence, if a non-redundant transition clock chain can be designed, then it can be expected to save significant power when the D input transitions are low.

Fig. 4.6 is a schematic diagram of a non-redundant transition clock chain. In Fig. 4.6, D and Q are the input and the output signals of a flip-flop respectively and CP is its clock. XOR gate is used to compare the input D and the output Q. When D=Q, the output of the XOR is 0 (1 when D≠Q). The output of the XOR gate is used as an enable signal for the tri-state inverter.

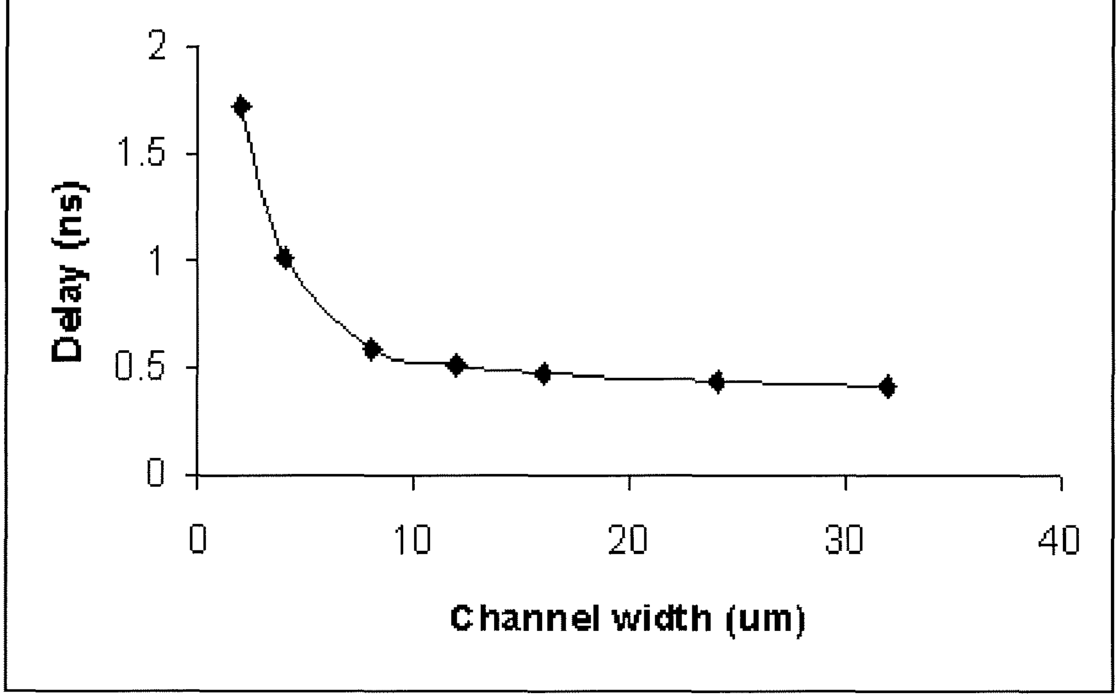


Figure 4.4: Delay versus width w

The clock chain is only driven by CP when D≠Q. Hence, compared to the conventional flip-flop, redundant CPs are eliminated. If CP_1 and CP_2 are used to control the gates of m7 and m8 the narrow pulses generated are also non-redundant. Since the narrow pulse is generated at the falling edge of CP, the flip-flop will sample the input signal at the falling edge of CP. Using the dual trail signals D and Q from the flip-flop, one possible circuit implementation is shown in Fig. 4.7 [125]. m9~m13 make up a control circuit, which is used to selectively pass CP to the clock chain. When $D = Q$, m11 is off and CP is blocked to the clock chain. When $D \neq Q$, inverted CP is passed to the clock chain.

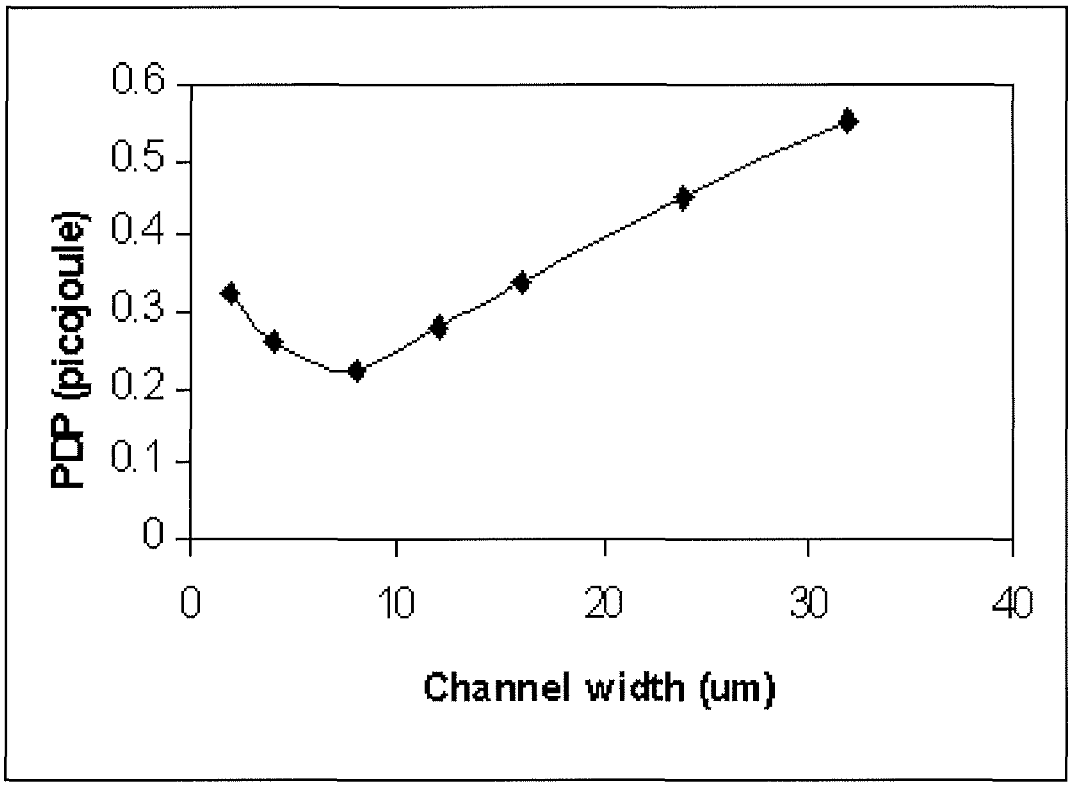


Figure 4.5: PDP versus width w

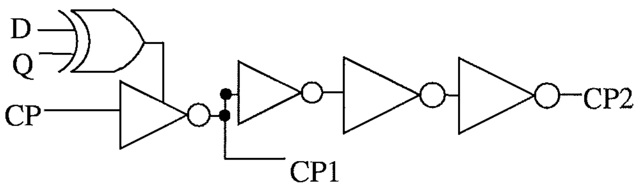


Figure 4.6: Non-redundant transition clock chain

4.4 Circuit simulations and power dissipation measurement

Circuit simulations are conducted using PSpice II with Level 3 at $5V$, 1μ technology and $50MHz$ clock. SPICE parameters for 1μ technology are shown in Table 4.1.

Table 4.1: SPICE parameters of MOS transistors for a generic $1.0\mu m$ process

.MODEL CMOSN NMOS LEVEL=3 PHI=0.6 TOX=2.03E-08 XJ=0.15U TPG=1
+VTO=0.7333 DELTA=9.445E-01 LD=1.0E-09 KP=1.2964E-04 U0=762.1 THETA=5.246 E-02
+RSH=2.3650 GAMMA=0.4481 NSUB=1.75E+16 NFS=2.356E+12 VMAX=1.487E+05
+ETA=1.485E-01 CJ=1.1962E-04 MJ=0.4398 CJSW=4.6953E-10 KAPPA=9.51E-02
+CGDO=2.5516E-12 CGSO=2.5516E-12 CGBO=3.0108E-10 MJSW=0.123994 PB=0.8
.MODEL CMOSP PMOS LEVEL=3 PHI=0.6 TOX=2.03E-08 XJ=0.15U TPG=-1
+VTO=-0.9679 DELTA=4.3070E-01 LD=1.0E-09 KP=4.3207E-05 U0=254 THETA=1.7060E-01
+RSH=2.5530 GAMMA=0.497 NSUB=2.153E+16 NFS=4.566E+12 VMAX=1.82E+05
+ETA=1.8290E-01 KAPPA=3.225 CGDO=2.5516E-12 CGSO=2.5516E-12 CGBO=3.5207E-10
+CJ=5.3093E-04 MJ=0.5074 CJSW=7.8757E-11 MJSW=0.077193 PB=0.85

The transistor lengths are minimum for the above technology, while the transistor widths

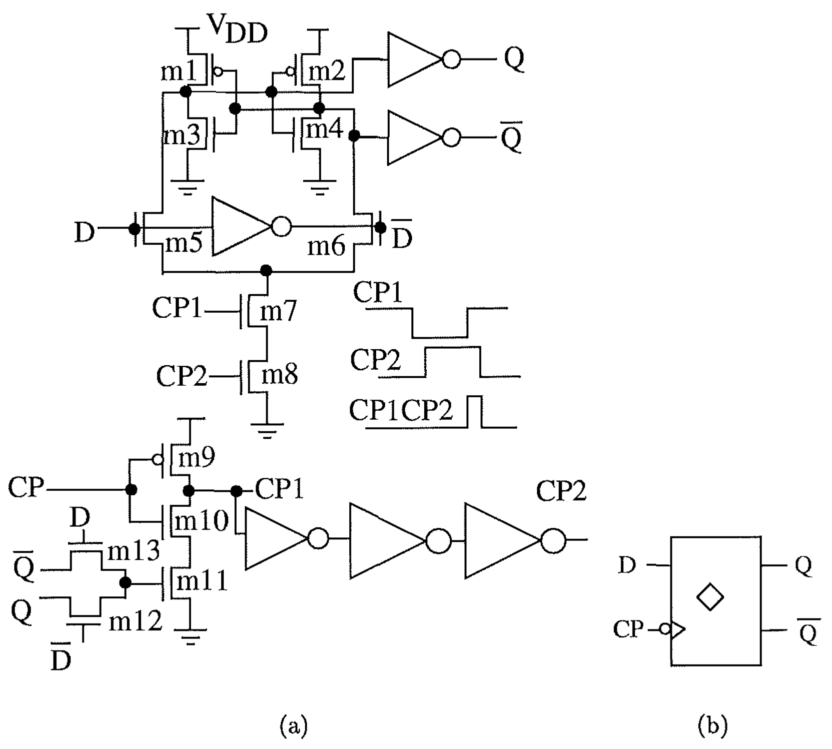


Figure 4.7: Differential single edge-triggered CMOS flip-flop with clock-gating (CG-SETFF)(a) Circuit implementation; (b) Logic symbol

are 8μ except m3, m4, m7, m8, m12, m13 and those in the clock chain. The widths of m7 and m8 are 12μ. The sizes of m3 and m4 are: $W = 2\mu$, $l = 4\mu$ while the sizes of m12 and m13 are: $W = 4\mu$, $L = 1\mu$. The transistor sizes of the clock chain, which are expressed in W/L, are indicated beside the inverters in Fig. 4.1. These parameters are determined for optimizing both speed and power as stated in Section 4.2.2. The PSpice simulation shows that the proposed flip-flop has the correct behavior as shown in Fig. 4.8.

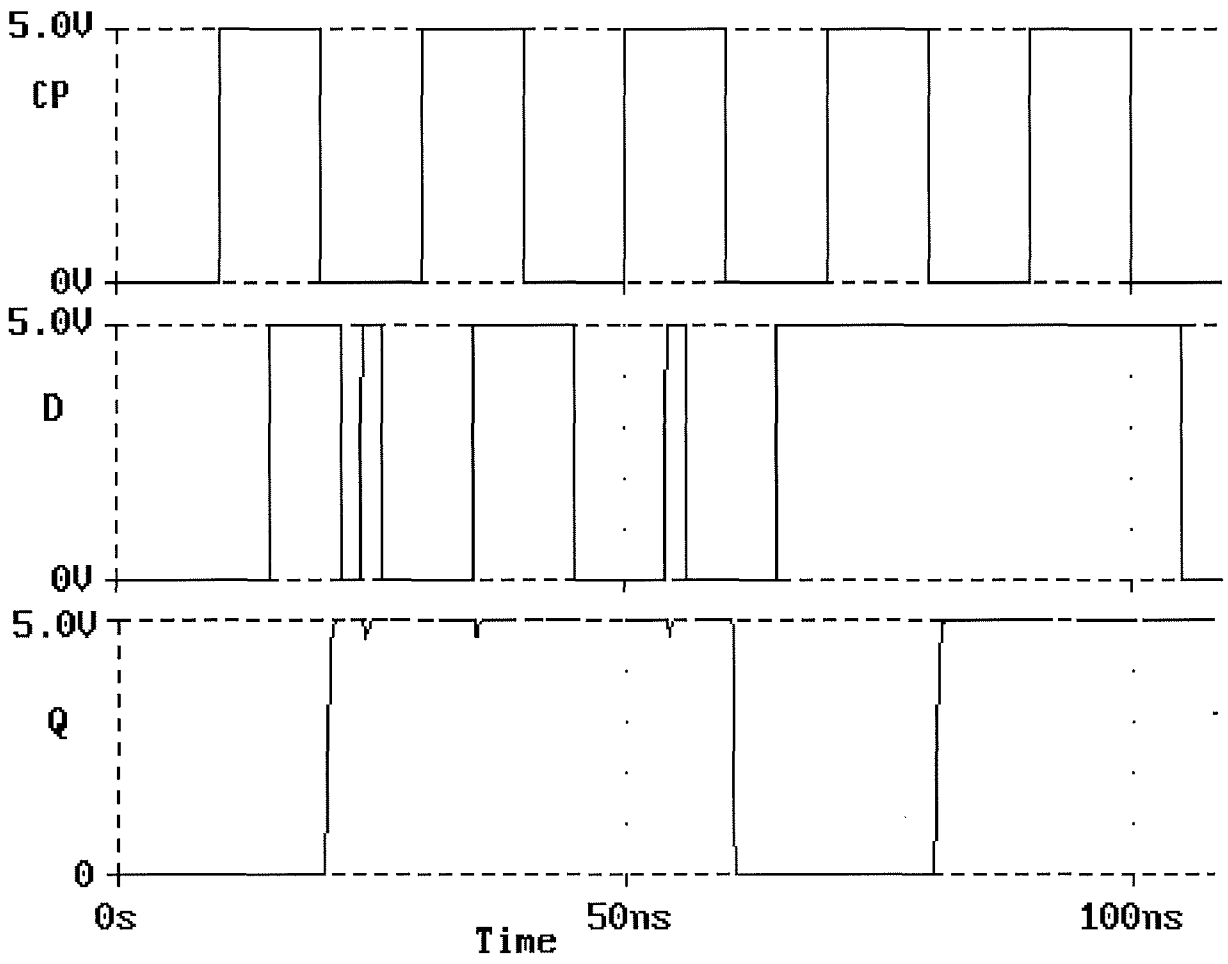


Figure 4.8: Transition behavior of proposed flip-flop

Average power dissipation is measured when each output is loaded with $C_L = 0.1 picofarad$ [72]. The power consumption is calculated as:

$$P_w = \frac{1}{T_m} \int_{T_m} V_{DD} I_{DD} \tag{4.4}$$

Here T_m is the measured time period of the D signal.

The ideal input signal D and clock signal CP are buffered by two series inverters. The test bench for measuring power dissipation is shown in Fig. 4.9. The reason why this test bench is chosen is to provide realistic data and clock signals, which themselves are fed from ideal voltage sources. Capacitive load at the data input simulates the fanout signal degradation from the previous stages while capacitive load at the outputs simulate the fanout signal degradation caused by the succeeding stages. The circuit with heavy load is to estimate its driving capabilities. To estimate the power consumption, the simulation is performed by varying the switching activity of the D input. The switching activity is defined here as the average number of transitions of D in a clock cycle. The maximum switching activity for a glitch free signal is 1 while switching activity could be greater than 1 if the presence of glitches is taken into account. The switching activity for a signal that behaves as clock signal is equal to 2.

Power consumption is reported as a function of input signal switching activity, assuming that the input signal has no glitches. Six input cases are considered for the switching activity of 0, 0.2, 0.4, 0.6, 0.8, 1.0. The methodology proposed in [99] is followed to measure the power dissipation. The power spent on the capacitative load needs to be excluded to get a fair picture of the circuit's power behavior. For the given load $C_L = 0.1 picofarads$, that portion of power reaches the values presented in Table 4.2.

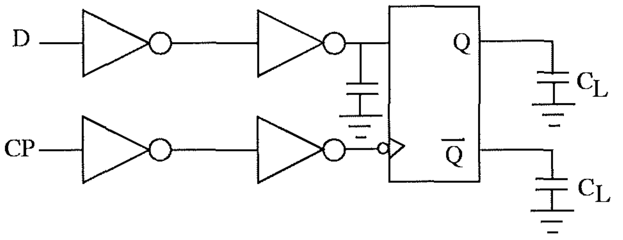


Figure 4.9: Test bench for measuring the power dissipation of flip-flops

The power dissipation is evaluated for $T_m = 400ns$. The measure statement of average power dissipation in PSpice is used to measure the interested power dissipation.

For a fair comparison of the proposed circuit and that reported in [72], the same PSpice simulator and 1μ technology are used to simulate the circuits. The result of simulations is shown in Fig. 4.10. From Fig. 4.10, it can be seen that the proposed flip-flop consumes lower power than the conventional one when the D input switching activity is < 0.65. When D is idle, the power dissipation of the proposed circuit is only 14% of that of a conventional flip-flop. The reason is that when D is idle both the cell of the flip-flop and the clock chain are shutdown and hence the dynamic power dissipation is nearly zero. However, when the switching activity of the input signal is greater than 0.65, the extra devices consume the significant power. As a result, the total power is greater than that of the conventional one.

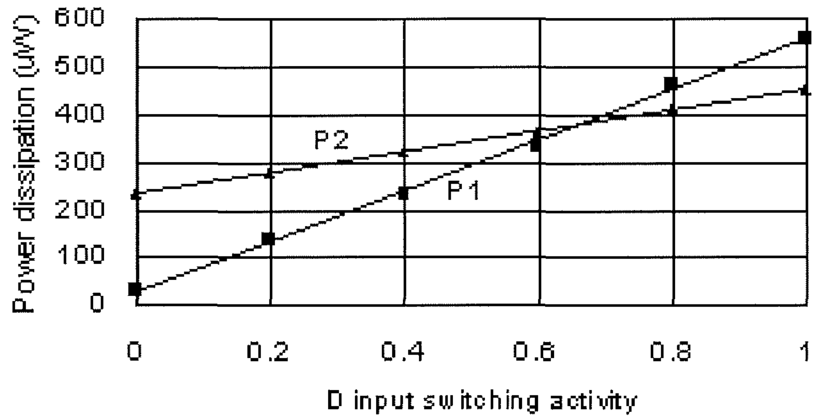


Figure 4.10: Power dissipation against D input switching activity. P1: Proposed flip-flop; P2: Conventional flip-flop

Table 4.2: Power dissipation for load capacitors versus switching activities

$P_{out} = \alpha\frac{fC_{out}V_{dd}^2}{2}$ $f = 50MHz$, $V_{dd} = 5V$	Differential structures $C_{out} = 2C_L = 0.2picofarad$
$\alpha = 1.0$	$125\mu W$
$\alpha = 0.8$	$100\mu W$
$\alpha = 0.6$	$75\mu W$
$\alpha = 0.4$	$50\mu W$
$\alpha = 0.2$	$25\mu W$
$\alpha = 0.0$	$0\mu W$

4.5 PDP measurement

To obtain PDP defined in equation 4.2, the delay needs to be measured. The methodology in [1] is followed and the test bench in Fig. 4.11(a) is employed. The principle behind this method is that the test bench forms a ring oscillator. To measure the time figures for two flip-flops, the frequency of the clock pulse is gradually increased up to a point where the oscillator fails to operate. This is the minimum clock period of the flip-flop, which has:

$$T_{min} = D_{CQ} + T_S + D_L \tag{4.5}$$

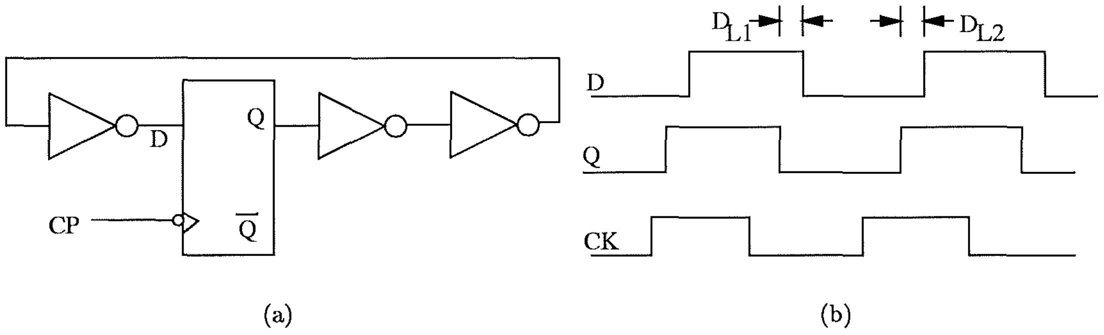


Figure 4.11: Timing test circuit (a) Bench circuit; (b) Timing diagram

Here, D_{CQ} and T_S are as defined previously, D_L is the delay of the combinational circuit.

The delay of the flip-flop can be obtained as long as D_L can be determined. From the test bench, it is known that D_L is the inverter chain delay. Fig.4.11(b) shows the timing diagram. Then, to find the delay of the flip-flop, D_{L1} and D_{L2} need to be subtracted from the clock period. We measured that the delay of the conventional flip-flop is 0.355ns while the delay of the proposed circuit is 0.550ns. Then, PDP can be calculated based on the definition in equation 4.2. Fig. 4.12 presents the relationship of PDP versus the switching activities for the proposed flip-flop and the conventional flip-flop. It can be seen that if using PDP to compare, the proposed flip-flop has higher energy efficiency than the conventional one when the switching activity of the input signal is lower than 0.4.

4.6 Low power binary counters

4.6.1 4-bit binary counter

The proposed structures are suitable for applications with reduced switching activity. A typical application is in synchronous counter design. In a synchronous counter, the input switching activity in each flip-flop is different from place to place and is known beforehand. Hence, if the proposed flip-flop is used in the low switching activity bits of the counter, a low power counter can be obtained. Take a 4-bit binary counter as an example. It shows as follows.

If A, B, C and D are named for four flip-flops from the Least Significant Bit (LSB) to the Most Significant Bit (MSB), then the state table for this counter is shown in Table 4.3. D, C, B and A are used to represent present state variables while D', C', B' and A' to represent next state variables.

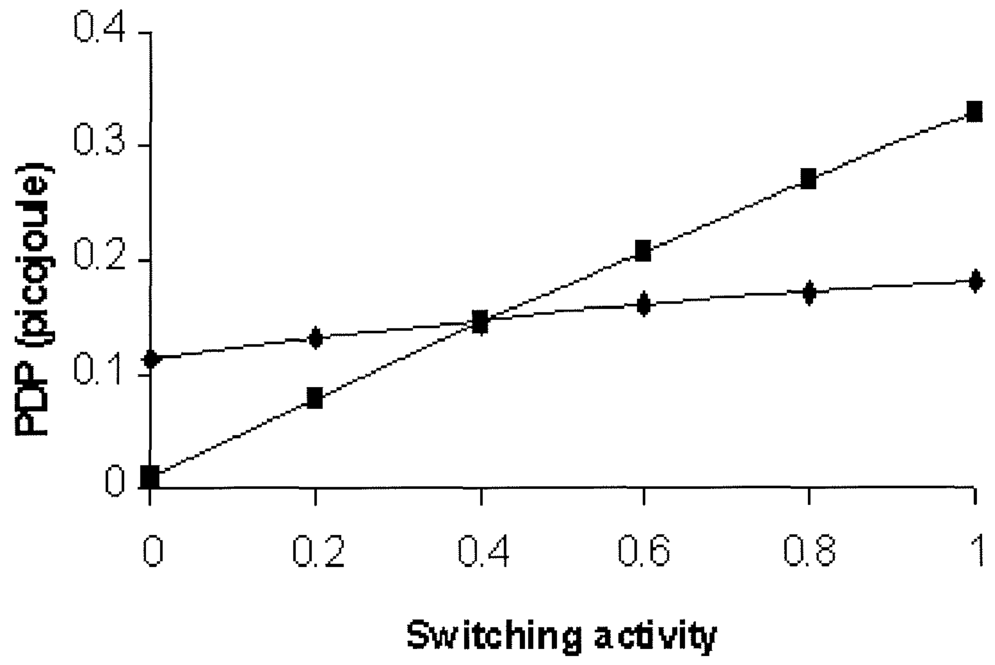


Figure 4.12: PDP versus switching activities

Assuming that D flip-flops are used, the following equations are obtained.

$$\begin{cases} D_A = \overline{A} \\ D_B = \overline{A \oplus \overline{B}} \\ D_C = \overline{C \oplus \overline{AB}} \\ D_D = \overline{D \oplus \overline{ABC}} \end{cases} \tag{4.6}$$

Hence, using D type SET-FFs, the counter can be implemented in Fig.4.13. From the working behavior of the counter, it can be known that the switching activity for each flip-flop input is 2^{-k} for the kth bit, that is, $\alpha_A = 1.0$, $\alpha_B = 0.5$, $\alpha_C = 0.25$ and $\alpha_D = 0.125$. It can be seen that except the lowest-order bit (with switching activity 1.0) the other bit switching activities are less than and equal to 0.5. In Section 4.4, it shows that using the proposed flip-flop can save power if the input switching activity is less than 0.65. If we use the SET-FF for the lowest bit while the proposed flip-flops (CG-SETFFs) are used for the remaining bits, then a lower power counter can be obtained. However, for convenience, we use four CG-SETFFs instead.

Table 4.3: State table for a 4-bit binary counter

	Present state				Next state			
Count	D	C	B	A	D'	C'	B'	A'
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	1	0
10	1	0	1	0	1	0	1	1
11	1	0	1	1	1	1	0	0
12	1	1	0	0	1	1	0	1
13	1	1	0	1	1	1	1	0
14	1	1	1	0	1	1	1	1
15	1	1	1	1	0	0	0	0

In order to measure the power dissipation, NAND gate and XNOR gate need to be implemented as well. The traditional design for 2 input and 3 input NAND is employed while for XNOR, the low power design in [118] is used, which are shown in Fig. 4.14 for convenience. All transistor sizes are $W = 8\mu m$, $L = 1\mu m$.

Once again, the above two circuits are simulated using PSpice with level 3 at $5V$, 1μ technology and $50MHz$ clock. The power dissipation was evaluated for $T_m = 400ns$. The experiment shows that 24% of the power is saved using the proposed flip-flop to implement the counter. However, the saving varies with the number of bits. Because the circuit has a complicated combinational section, it is expected that the saving significance will be reduced with increasing the bit number of the counter.

4.6.2 Binary twisted ring counters

It is well known that binary ring counter has simple combinational circuit while there are many redundant transitions. If the proposed flip-flop is used in these applications, the power saving will be much more significant than binary counters because the latter has complicated combinational circuits. Using SET-FFs and CG-SETFFs, we implemented 4-bit, 6-bit and 8-bit twist ring counters. Then, the same parameters as in Section 4.6.1 are used to simulate the circuits. Fig. 4.15 shows the power comparison results for the SET-FF based and CG-SETFF based twist ring

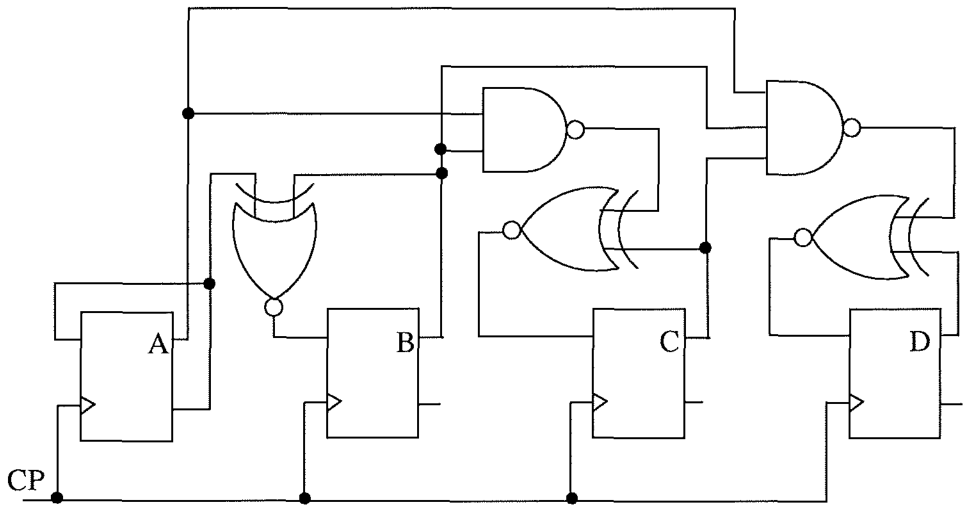


Figure 4.13: SET-FF based binary counter

counters. It can be seen that the counter, using the proposed flip-flops to build, has much smaller power than that using the conventional flip-flops. This is because there are many redundant transitions in the twisted ring counter, which are blocked by new flip-flops. Furthermore, it can be expected that the power savings become even more significant with the increase of the counter

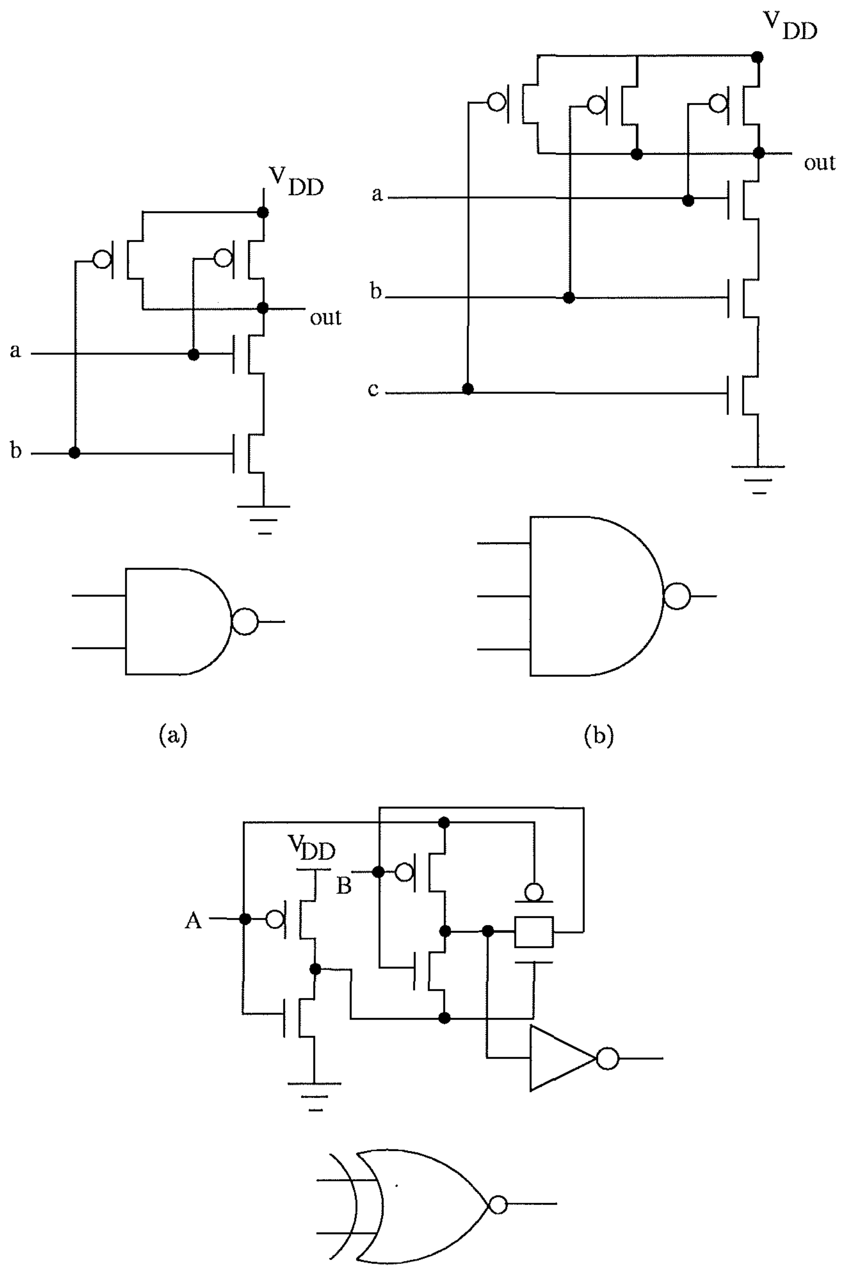


Figure 4.14: Gate circuit implementation (a) 2 input NAND and its logic symbol; (b) 3 input NAND and its logic symbol; (c) XNOR and its logic symbol

bit number. The reason is that the redundant transitions increase with the bit number of the counter. For example, a possible state diagram for a 4-bit twisted ring counter could be as in Fig. 4.16, which has eight states. For each state transition between two states, only one bit needs to be changed while the other three stay at the same state. The redundant transition rate is 75%. For an 8-bit twist ring counter, it could be up to 87.5%.

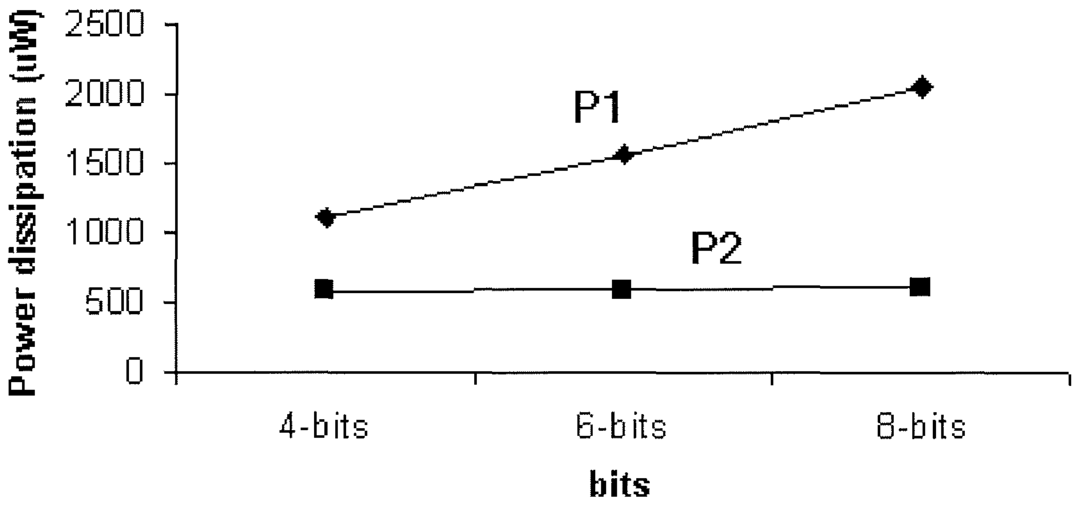


Figure 4.15: Power dissipation for twisted ring counters: P1- SET-FF based counters; P2-CG-SETFF based counters

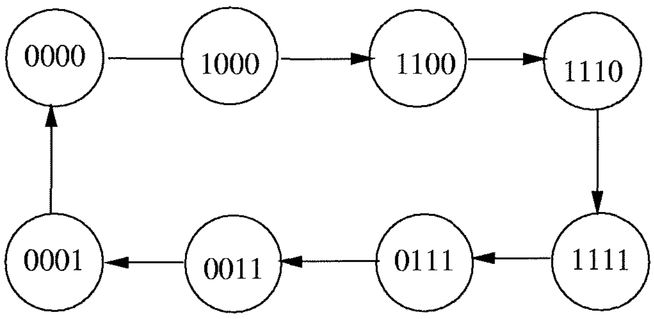


Figure 4.16: State diagram for a twisted ring counter

4.7 Summary

A non-redundant transition clock chain for differential CMOS single edge-triggered flip-flops is presented. The experimental results show that compared to the conventional flip-flop, the proposed flip-flop can achieve significant power savings when the D switching activity is < 0.65. Reduction of power consumption can be as high as 86% when the input is idle. However, using PDP to measure them, the proposed flip-flop has higher energy efficiency than the conventional one when the D switching activity is < 0.4. Since the dual trail signals are provided by differential edge-triggered flip-flops, the additional circuit to implement non-redundant transition chain is simpler. In this design, the increased transistor count is 25% while in [100] it is 144%. Hence, the clock-gating technique is more suitable to differential edge-triggered flip-flops with clock-racing than to level sampling flip-flops. The comparison is also made for some applications. The result shows that significant power savings can be obtained. However, the saving varies from application to application.

Chapter 5

Differential CMOS double edge-triggered flip-flops

In a digital system, the sequential part is the main contributor to power dissipation. The reason is that one of the inputs to a sequential circuit is the clock pulse (CP) signal. Each CP has one rising transition and one falling transition in one CP period. Hence, the switching activity is 2. However, other signals in the circuit have at most one transition in one CP period if the glitches from signal racing are not taken into account. On the other hand, the load of CP is alway the highest in a digital system. For example, to distribute CPs and control the skew of CPs, one needs to build a clock tree which consists of buffers, which results in increasing the total node capacitance of a CP tree. As a result, research confirms that in a digital system 20~45 % of the power is consumed by CPs.

However, traditional flip-flops are only triggered by a specific edge (rising edge or falling edge) of a CP, which is called Single Edge-Triggered Flip-Flops (SET-FFs). On the other edge, it is just wasteful to charge or discharge the capacitive load of the global clock line in a system. This is particularly true in CMOS because the dynamic power is the dominant power. Therefore, reducing the CP power consumption should reduce the total power dissipation for a system. If flip-flops can be triggered by two edges, then half of power dissipation from CPs can be saved. This leads to the design of double edge-triggered flip-flops [49].

5.1 Previous work

The double edge-triggered (DET) idea was presented as early as 1981 [110]. At that time, it targeted high performance for flip-flops. With the increasing size of VLSI systems, power dissipation gained the equal priority to area and speed. The double edge-triggered idea was developed to reduce the power dissipation of VLSI systems [49][19].

Based on input sampling mode, DET flip-flops can be classified as level sampling and pulse sampling. Level sampling DET flip-flops can be implemented using two latches[49][19]. Two latches sample and store input signal alternatively under different clock phases. Pulse sampling flip-flops usually consist of one latch and one clock chain [102]. The clock chain is used to generate narrow pulses. The input signal is sampled during each narrow pulse.

For differential DET-FFs, level sampling version has been proposed in [68][58]. The disadvantage of DET flip-flops has been the substantial increase in the number of transistors required to build such flip-flops compared to single edge-triggered (SET) flip-flops. For example, a typical SET-FF needs 16 transistors while DET flip-flop proposed in [58] needs 28 transistors. A recently proposed circuit in [68] needs 26 transistors, which is shown in Fig.5.1 and is called BALCS-DETFF.

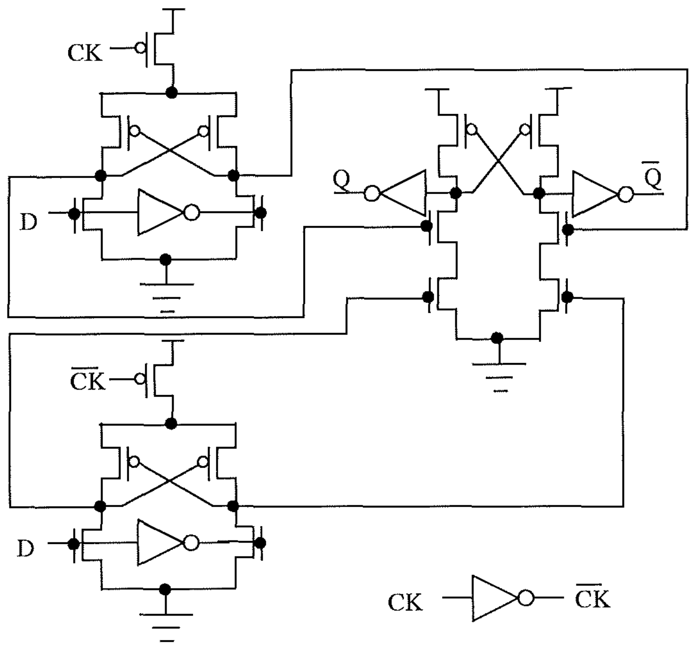


Figure 5.1: Differential CMOS DET flip-flop proposed in [68]: BALCS-DETFF

5.2 Clock Chain Based Double Edge-Triggered Flip-Flop (CCB-DETFF) using single latch

Single-Edge-Triggered Flip-Flops (SET-FFs) change their states either at the rising edge or at the falling edge of the clock pulse. Take the differential CMOS SET-FF proposed in [72] for example. For convenience, it is redrawn in Fig. 5.2, which consists of a flip-flop cell and a clock chain. The clock chain generates a delayed CP_1. Then, CP and CP_1 are applied to the cascaded transistors, m7 and m8, respectively, which generate a narrow pulse at every rising edge of each clock pulse. Hence, the flip-flop is triggered at the rising edge of the clock pulse. However, this design suffers from one disadvantage. The flip-flop responds only once per each clock pulse cycle. The other of the two clock transitions accomplishes nothing for the flip-flop. But this transition may cause transitions of some logic gates within the flip-flop, which consumes power.

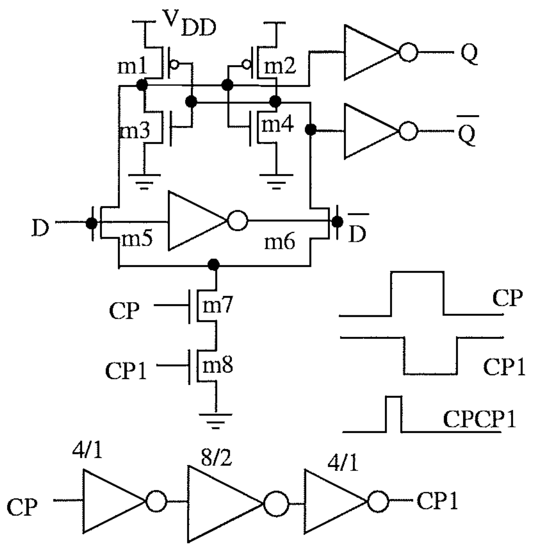


Figure 5.2: Differential CMOS single-edge triggered flip-flop (SET- FF)

To solve this problem, there are two solutions. One is to refrain from redundant clock transitions, which is proposed in [125]. The other is to use the other transition to change states. With both edges able to cause state transitions, the redundant clock transitions can be eliminated. Moreover, the clock period can be shortened since there is no need to wait for the clock signal to toggle. This is the design principle of Double Edge-Triggered (DET) scheme which enables a halving of the clock frequency and hence reduces power dissipation [49].

5.2.1 CCB-DET flip-flop

From Fig. 5.2, a clock chain consisting of three inverters can generate a delayed CP, CP_3. If we use four inverters to implement a clock chain, then four delayed CPs can be generated. The timing diagram is shown in Fig. 5.3 (b) where CP_4 is a delayed CP_1. If CP and CP_3 are logically ANDed while CP_1 and CP_4 are ANDed, then two narrow pulses corresponding to two edges of a clock pulse can be obtained as shown in Fig. 5.3. To even two narrow pulse width, W/L of transistors in the clock chain is taken as indicated beside four inverters. This can be proved as follows.

If t_1 is the delay of a 4/1-inverter while t_2 is the delay of a 8/2-inverter. Then the delay between CP_3 and CP is:

$$t_{cp_3-cp} = t_1 + 2t_2$$

Also, we have the delay between CP_1 and CP_4 :

$$t_{cp_4-cp_1} = 2t_2 + t_1$$

Hence, two narrow pulses have the same pulse width.

The proposed double edge-triggered flip-flop cell is as in Fig. 5.4. There are two branches, which are composed of four transistors, m7, m8, m9 and m10, at the common source terminal of the amplifier. CP and CP_3 are used to control the gates of m7 and m8 to generate a narrow pulse corresponding to the rising edge of clock pulse while CP_1 and CP_4 are used to control the gates of m9 and m10 to generate the other narrow pulse corresponding to the falling edge of clock pulse. In this way, the flip-flop can change its state triggered by two clock transitions.

5.2.2 Circuit simulations

Circuit simulations are conducted using PSpice II with Level 3 at 5V, $1.0\mu m$ process. The transistor lengths are minimum for the above technology, while the widths are $8\mu m$ except those specified as follows. The widths of m7, m8, m9 and m10 are $12\mu m$ while the sizes of m3 and m4 are $W = 2\mu m$, $L = 4\mu m$. The transistor sizes of the clock chain, which are expressed in W/L, are indicated beside the inverters in Figs. 5.2 & 5.3. These parameters are determined

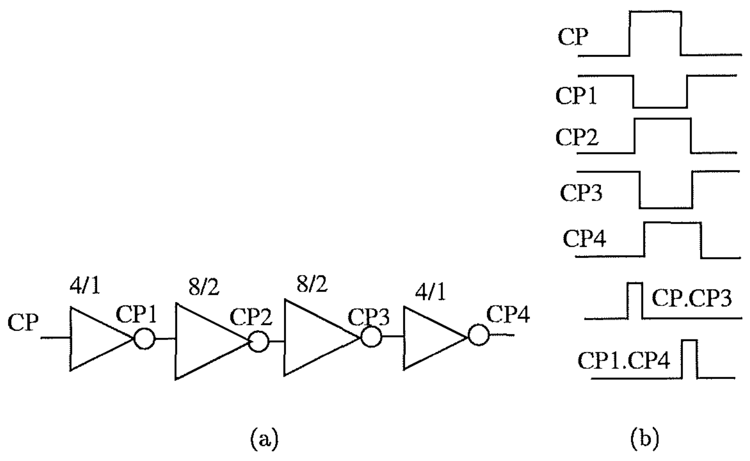


Figure 5.3: Double edge pulse generation (a) Clock chain; (b) Timing diagram

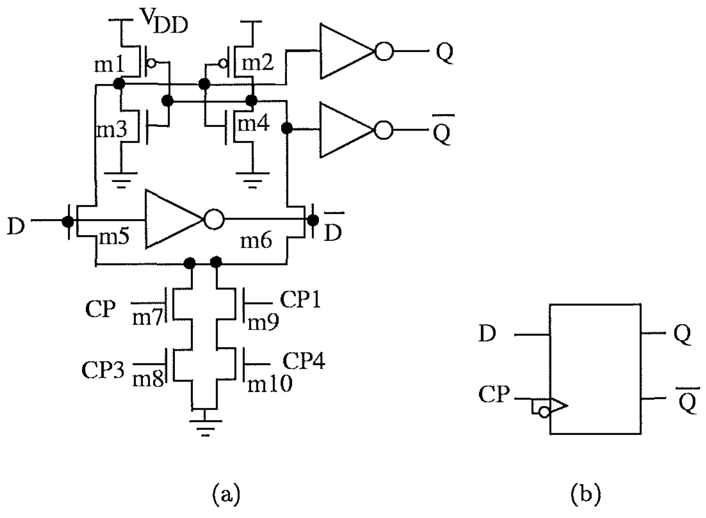


Figure 5.4: Clock Chain Based double Edge-Triggered Flip-Flop (CCB-DETFF) cell (a) Flip-flop cell; (b) Logic symbol

for minimum power as stated in Section 4.2.2. The PSpice simulation shows that the proposed flip-flop has correct transition behavior as shown in Fig. 5.5. In Fig. 5.5, the transient behavior of the single edge-triggered flip-flop is also shown for comparison.

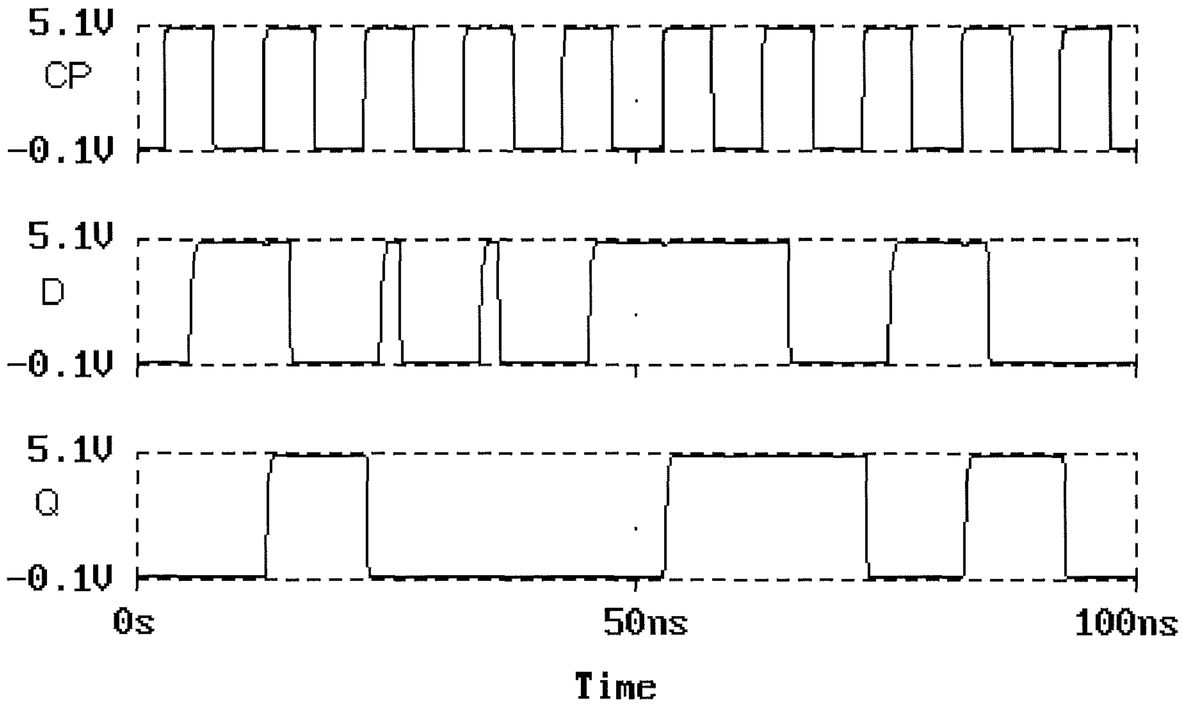
The test bench in Fig. 4.9 is used to simulate circuits. Circuits in Figs. 4.7 & 5.2 are used to be reference circuits. Once again, six input cases, the switching activity of 0, 0.2, 0.4, 0.6, 0.8, 1.0, are considered to test the power dissipation of circuits. However, to get the same output throughput as the double edge-triggered flip-flop does, the clock frequency to the SET flip-flops should be $100MHz$ while the clock frequency for the DET counterpart is $50MHz$. The results are shown in Fig. 5.6 and Table 5.1.

Table 5.1 shows the power saving rate compared to SET-FF in Fig. 5.2. From Fig. 5.6, it can be seen that CCB-DETFF always has lower power dissipation than SET-FF with different input switching activities. However, the saving rate varies with the input switching activity α. From Table 5.1, when $\alpha = 0$, the saving rate is up to 44.7%. Ideally, the saving rate should be 50% because the clock frequency is reduced half and the power is consumed mainly by the clock chain. The reason which is not equal to 50% is that two cascaded transistors, m9 and m10, and the extra inverter in the clock chain consumes the power. When $\alpha = 1.0$, the saving rate is reduced to 20.8% from 44.7% when $\alpha = 0$. This is because the flip-flop cell also consumes significant power and the weight of the clock power in the total power is decreased.

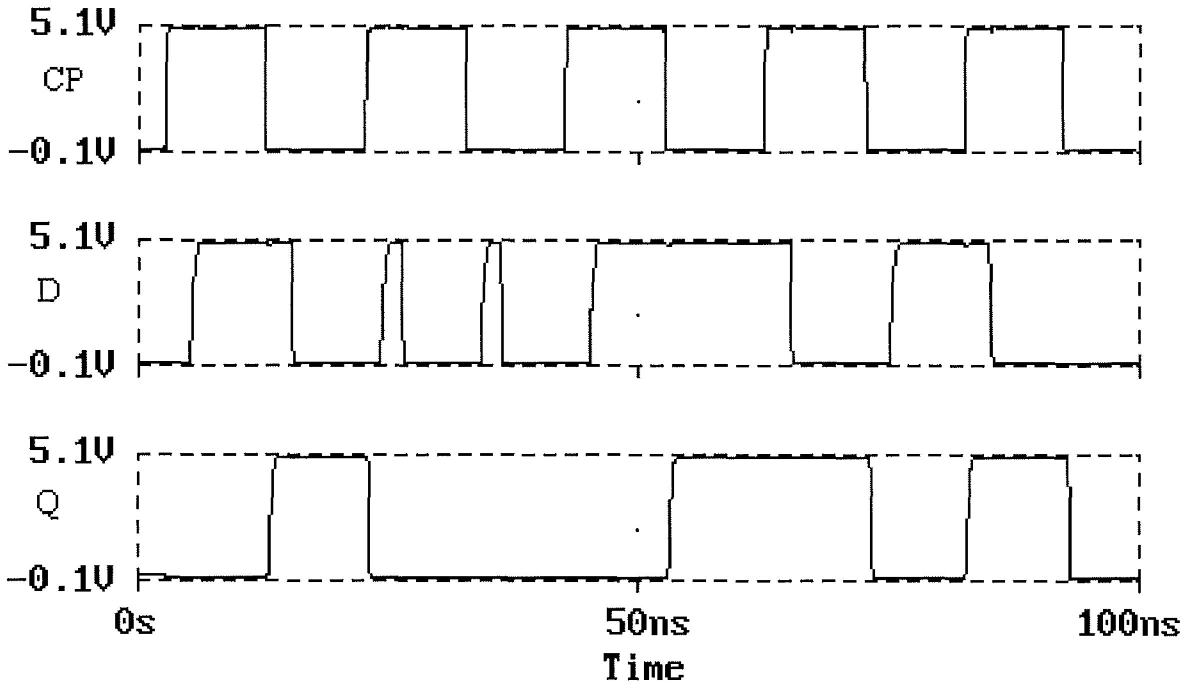
Table 5.1: Power savings compared to the circuit in Figs.4.7 & 5.2

Switching activities (α)	0	0.2	0.4	0.6	0.8	1.0
SETFF (%)	0	0	0	0	0	0
CG-SETFF (%)	86.5	54.2	28.4	5.5	-14.8	-30.1
CCB-DETFF (%)	44.7	37.7	30.8	27.3	23.7	20.8

From Fig. 5.6, among three flip-flops, CG-SETFF has the lowest power dissipation when $\alpha < 0.38$ while CCB-DETFF has the lowest power dissipation when $\alpha > 0.38$. In addition, SET-FF uses 20 transistors while CG-SETFF uses 25 transistors. However, CCB-DETFF uses 24 transistors.



(a)



(b)

Figure 5.5: Transient behavior (a) Differential CMOS SET-FF in Fig. 5.2; (b) Differential CCB-DETFF in Fig. 5.4

5.3 Pass Transistor Based Differential CMOS Double Edge-Triggered Flip-Flop (PTB-DETFF)

5.3.1 Voltage scaling property for CCB flip-flops

A common disadvantage for the above designs is that there are cascaded transistors in the common source terminal of the differential amplifier (briefly called clocked transistor), which introduces the extra resistance. This degrades the input threshold of the amplifier. This is investigated by measuring the input threshold versus the gate voltage of the clocked transistors in Fig. 5.2. The experiment is conducted as follows.

DC characteristic of the circuit in Fig. 5.2 is simulated using PSpice with 1 um technology and 5V power supply. The Input Threshold Voltage (ITV) is defined as the input voltage when the output high level of the circuit is 2.4V. Varying the gate voltage of clocked transistors, the ITV versus the gate voltage is measured. Then, reduce the number of clocked transistors in Fig. 5.2 to one and repeat the above experiment. For the simplification, the former circuit is

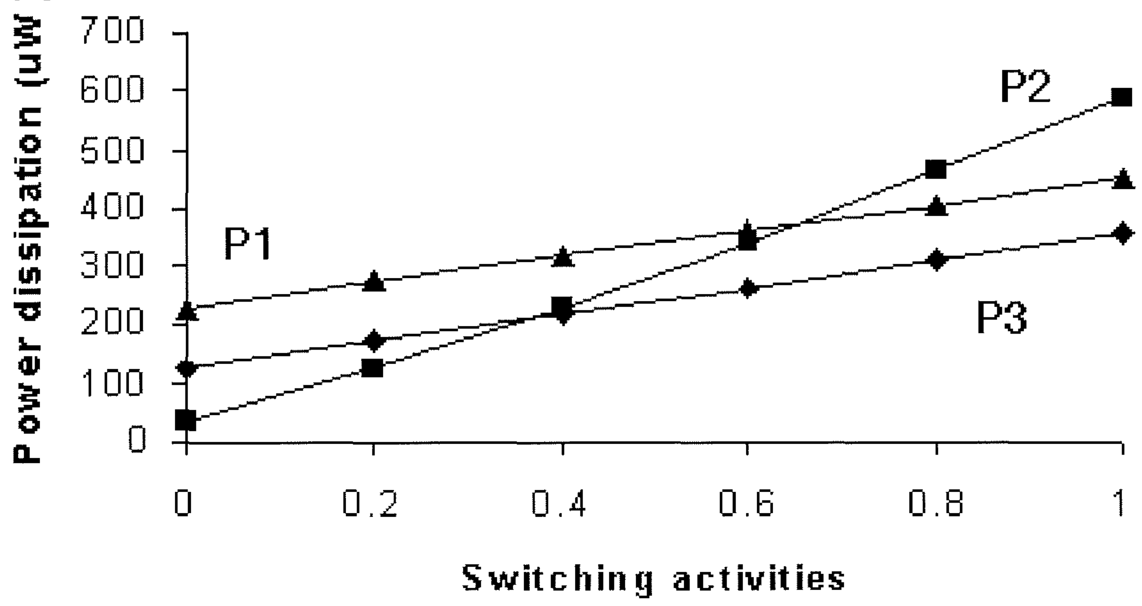


Figure 5.6: Power dissipation versus switching activities: P1- result for Fig. 5.2; P2-result for Fig. 4.7; P3-result for the proposed circuit

called Cir_{2ct} while the latter Cir_{ct}. The results are shown in Fig. 5.7. It can be seen that the circuit, Cir_{ct}, which has one clocked transistor has lower ITV than the circuit, Cir_{2ct}, which has two clocked-transistors and furthermore the former is still workable when the gate voltage of the clocked transistor is 2.0V .

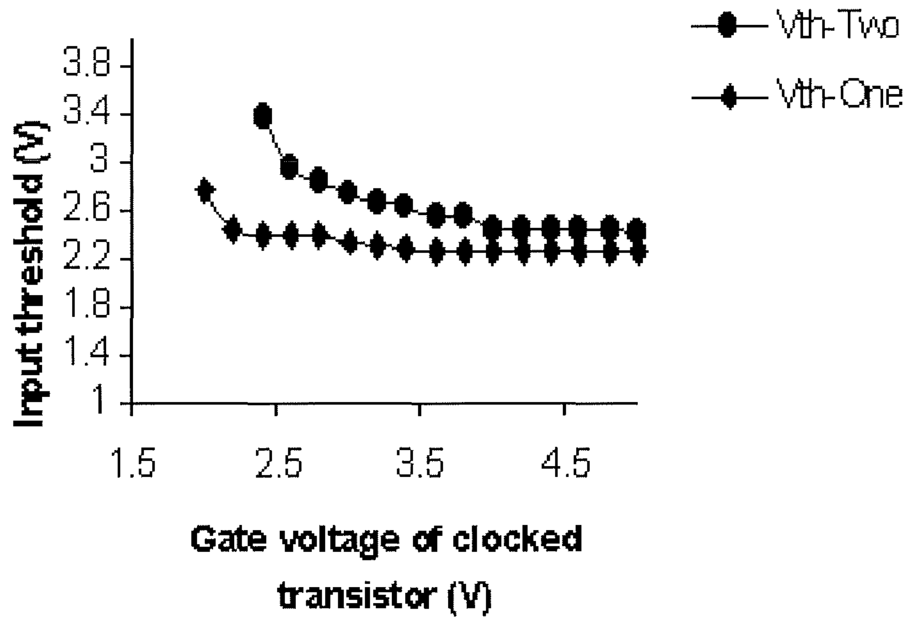


Figure 5.7: Input threshold voltage versus gate voltage of clocked transistors. NB: Vth-Two stands for threshold voltage of Cir_{2ct} while Vth-One for that of Cir_{ct}

This confirms that the cascaded transistors in the common emitter degrade the ITV of the amplifier. From Fig. 5.7, it also shows that reducing the gate voltage of clocked transistors has less effect for the ITV of Cir_{ct} than for that of Cir_{2ct}. The ITV will increase fast when the gate voltage of the clocked transistors is reduced to 4.0V for Cir_{2ct}. However, for Cir_{ct}, the ITV remains the same when the gate voltage is reduced to 2.0V. Hence, it is possible for Cir_{ct} to use low clock swing, which will significantly save the clock power. As known, dynamic power is proportional to voltage squared. Hence, a good voltage scaling property is important to design a low power differential flip-flop.

5.3.2 PTB-DETFF circuit

The proposed circuit is based on the following idea. If a narrow pulse corresponding to each edge of clock pulse is available, the two cascaded transistors in the common source terminal in Figs. 5.2 & 5.4 can be substituted by one transistor. Then the input threshold degrading can be avoided. Furthermore, if two narrow pulses corresponding to two edges of the clock pulse are generated to drive the transistor, double edge-triggered differential flip-flop can be implemented.

Narrow pulses can be generated by the clock-racing. A possible scheme is shown in Fig. 5.8. There, CP_1 is a delay CP and the delay time is determined by two cascaded inverters. If they are input into an XOR gate, then narrow pulses (NPs) corresponding to two edges of clock pulse can be obtained. This is explained in Fig. 5.8.

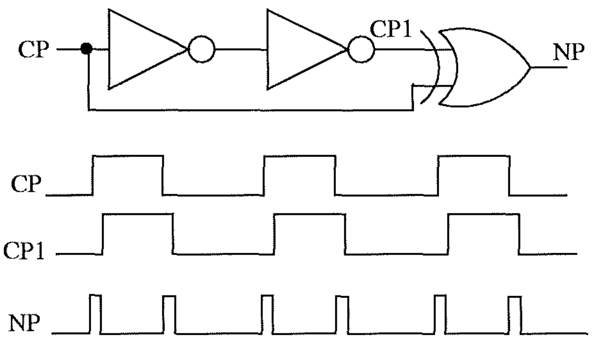


Figure 5.8: Narrow pulse generating scheme

Using transmission transistors, a possible circuit implementation is shown in Fig. 5.9 (a). All branches are ANDed together as the output. From Fig. 5.8, CP & CP_1 have four combinations, namely 00,10,11 & 01, labelled as A, B, C & D respectively. As known, transmission of a logic zero is not degraded through nMOS transistor while transmission of logic 1 is not degraded through pMOS transistor[118]. Take Case A for example to analyze the working principle of the circuit. For Case A, $CP = 0$ and $CP_1 = 0$. CP is applied to the input of pass transistor mt4 while $CP_2 = 1$, inverted CP_1, is applied to the gate of mt4. Then, logic zero is transmitted to the output of mt4. The other cases can be analyzed similarly. However, it should be noticed that narrow pulse width is widened because of the delay of inverter 3. Hence, when determining the parameters of inverter 3, this factor should be considered.

Having this pulse generator, the flip-flop cell can be as shown in Fig. 5.9 (b) where only one transistor m7 is required at the common source terminal. Because the generator only drives one nMOS, the driving capability of Fig. 5.9 (a) is not a problem.

5.3.3 Circuit simulations

Circuit simulations are conducted using PSpice II with level 3 at $5V$, 1μ technology and $50MHz$ clock. The transistor lengths are minimum for the above technology, while the widths are $8\mu m$ except those specified as follows. The widths of m7 and m8 are $12\mu m$ while ones of mt1, mt2, mt3 and mt4 are $2\mu m$. The sizes of m3 and m4 are $W = 2\mu m$, $L = 4\mu m$. The transistor sizes of the clock chain, which are expressed in W/L, are indicated beside the inverters in Fig. 5.9. These parameters are determined for optimizing the power. The PSpice simulation shows that the proposed flip-flop has correct transition behavior as shown in Fig. 5.10.

The test bench in Fig. 4.9 is used to simulate the circuits and the same methodology is followed. Average power dissipation is measured when each output is loaded with $C_L = 0.1picofarad$. Four input sequence cases are considered: Case 1, Case 2, Case 3 and Case 4. Case 1 has an input change at every clock phase while Case 2 has an input change at every-other clock phase. Case 3 and Case 4 are constant but with a glitch at every and every-other clock

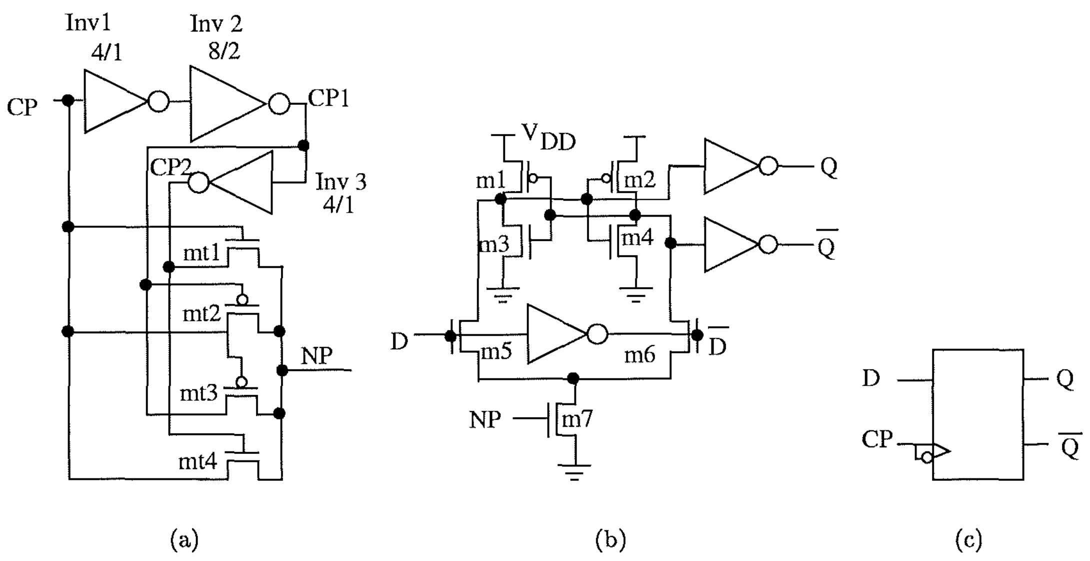


Figure 5.9: Proposed differential CMOS double edge-triggered flip-flop (DET-FF) (a) Narrow pulse generator; (b) Double edge-triggered flip-flop cell; (c) Logic symbol

phase, respectively. The power dissipation is evaluated for a time period of 400 ns. The power spent on output loads is excluded from the total power to get a fair picture of the power behavior of the circuit.

The same PSpice simulator and 1μ technology are used to simulate SET-FF circuit given in Fig. 5.2. However, to get the same output throughput as the double edge-triggered flip-flop, the clock frequency to the single edge-triggered flip-flop should be $100MHz$.

Table 5.2 shows power reduction compared to the power dissipation of the circuit in Fig. 5.2, which includes clock power dissipation. It can be seen that the proposed circuit has significant power savings compared to the single edge-triggered counterpart. However, the power saving is not equal to 50%. The reason is that three extra pass transistors consume power.

Table 5.2: Power dissipation comparison between SET-FF and PTB-DETFF

Input sequences	Case 1	Case 2	Case 3	Case 4
SET-FF in Fig. 5.2 (μW)	489.7	383.3	279.2	240.2
PTB-DETFF in Fig. 5.9(μW)	318.9	231.7	175.0	155.4
Power Reduction (%)	34.9	39.6	37.3	35.3

Table 5.3 shows the property of voltage scaling for two circuits. Case 2 is chosen as a test sequence. It can be seen that the proposed circuit has better property of voltage scaling. The

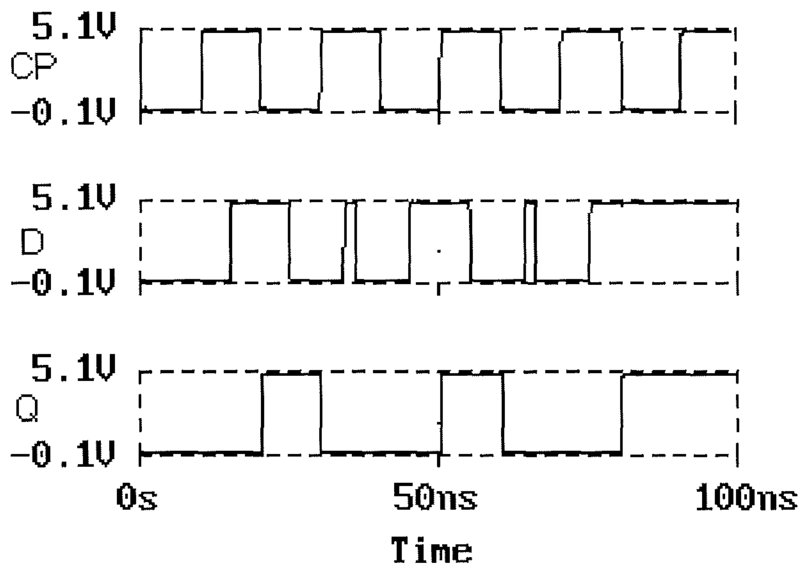


Figure 5.10: Transition behavior of the propose circuit

proposed circuit can work well even if the supply voltage is as low as 1.2V while the circuit in Fig. 5.2 does not work when the supply voltage is lower than 1.8V. It can also be seen that the power reduction for the proposed circuit is up to 39.2% compared to the single edge-triggered one under workable supply voltages. The area penalty is that PTB-DETFF needs 3 transistors more than SET-FF.

Table 5.3: Voltage scaling property between SET-FF and PTB-DETFF

Supply voltage (V)	3.0	2.0	1.8	1.5	1.2
SET-FF in Fig.5.2(μW)	120.7	50.3	40.6	-	-
PTB-DETFF in Fig. 5.9(μW)	74.2	31.1	24.7	16.6	6.9
Power Reduction (%)	38.5	38.2	39.2	-	-

Then, BALCS-DETFF in Fig. 5.1 is simulated using 1μ technology. The width of transistors is $8\mu m$ while the length is minimum. The experiment is carried out as follows. Under the specified power supply voltage, the minimum clock swing is determined. Then the power is measured with corresponding input switching activities. Two power supply voltages, 5.0v and 2.5v, are selected. The results are shown in Table 5.4. In the table, the second row gives the minimum clock swing for given power supply voltage. The third row gives the input switching activities while the fourth row gives the measured flip-flops. The fifth row gives the power dissipation. The most bottom row shows the power reduction compared to BALCS-DETFF.

Table 5.4: Voltage scaling property between BALCS-DETFF and PTB-DETFF

Power supply (V)	5.0				2.5			
Mini V_{clock} swing (V)	2.0				1.4			
Switching activities	$\alpha = 1.0$		$\alpha = 0.5$		$\alpha = 1.0$		$\alpha = 0.5$	
DETFFs	BALCS	PTB	BALCS	PTB	BALCS	PTB	BALCS	PTB
Power (μW)	402.4	298	242.5	231.9	88.3	71.9	53.5	51.1
Power reduction (%)	25.9		4.4		18.6		4.5	

Both flip-flops have the same minimum clock swing under the same power supply. With $\alpha = 1.0$ and power supply 5.0v, the power reduction of PTB-DETFF can be up to 25.9% compared to BALCS-DETFF in Fig. 5.1. The reason is that when a clock is "high" at the voltage of V_{clock}, two clocked pMOS transistors in Fig. 5.1 do not switch off completely, leaving leakage current flow through either of two clocked pMOSs. However, PTB-DET flip-flop has no such problem. With $\alpha = 0.5$, the power reduction is 4%, which is much smaller than the case $\alpha = 1.0$. This is because PTE-DETFF has a long clock chain with pass transistor branches, which consumes significant power and this is taken into account. When the input signal has lower switching activity, the weight of the clock power cell in the total power is increased and dominates, which results in the decrease of the power savings. This can be confirmed by experiments. Table 5.5 shows clock power and flip-flop cell power for BALCS-DETFF and PTB-DETFF. The rate of clock power to cell power for PTB-DETFF is 0.69:1 when $\alpha = 1.0$ while it goes up to 1.37:1 when $\alpha = 0.5$. However, for BALCS-DETFF, it is 0.09:1 when $\alpha = 1.0$ while it is 0.17:1 when $\alpha = 0.5$. There are two solutions to further reduce the clock power. One is to refrain the redundant clock transitions, which is proposed in [125]. The other is that a number of flip-flops share a clock driver so that the clock power for each flip-flop is reduced [102]. Hence, if this is taken into account, the power saving for PTB-DETFF is even more significant compared to BALCS-DETFF.

Table 5.5: Power dissipation from clock chain and flip-flop cell with $V_{DD} = 5.0V$ and $V_{clock} = 5.0V$

Switching activities	$\alpha = 1.0$		$\alpha = 0.5$	
Power	Clock power	Cell power	Clock power	Cell power
BALCS-DETFF (μW)	26.4	301.0	26.2	150.6
PTB-DETFF (μW)	131.9	192.4	133.8	97.9

Finally, it is noticed that PTB-DETFF only uses 23 transistors while BALCS-DETFF needs 26 transistors. Compared with BALCS-DETFF, the area saving of the proposed circuit is 11.5%.

5.4 Summary

Two differential CMOS double-edge triggered flip-flops, CCB-DETFF and PTB-DETFF, are proposed in this chapter. CCB-DETFF could be the complementarity of the CG-SETFF to design a low power system when the input switching activity is greater than 0.38. However, further work needs to be done on how to design a system using both SET-FFs and DET-FFs. PTB-DETFF is composed of a narrow pulse generator and a flip-flop cell. It only needs one clocked NMOS transistor for the flip-flop cell. Hence, PTB-DETFF has good properties of voltage scaling and clock swing scaling, which are very important to design a low power system.

Chapter 6

Multi-valued flip-flop approach

In the binary logic, the logic 1 is nearly set to the supply voltage V_{dd}. Each charging or discharging to a node consumes $\frac{1}{2}V_{dd}^2C_L$. Hence, reducing V_{dd} is reducing logic level voltage and is an efficient approach for low power design. However, there are some limitations to further scale down V_{dd} as discussed in Section 1.2.2. If high supply voltage V_{dd} is still used and multiple logic levels are set between 0 and V_{dd}, then logic level voltage between two adjacent logic values can be reduced. Power dissipation for each charging and discharging to a node can be reduced. Take quaternary logic for example. If $V_{dd} = 5V$, then four logic levels can be: 5V (V_{DD}), 3.33V ($2V_{DD}/3$), 1.67V ($V_{DD}/3$), and 0V (V_{SS}). Corresponding logic values will be {3, 2, 1, 0}. Each charging or discharging to a node transitioning between two adjacent logic levels only consumes $\frac{1}{2}(\frac{1}{3}V_{dd})^2C_L$. From the signal transmission and storage point of view, using multi-valued signal can reduce transmission capacitance and result in power saving or improve transmission speed. Fig. 6.1 shows a comparative result to transmit an eight bit binary signal 01110010 transmitted in parallel and serial in binary and quaternary valued logic. Fig. 6.1(a) shows that using quaternary signal only half of signal lines are needed and hence transmission capacitance can be reduced while 6.1(b) shows that transmission time can be reduced to half.

Furthermore, multiple valued logic may resolve the serious pin-out problems encountered in some VLSI circuit designs since if signals are used in four or more states rather than only two the pin-outs could be reduced to less than half. Many logical and arithmetic functions have been shown to be more efficiently implemented with multiple valued logic because fewer operations, gates, signal lines, etc. are required [31]. Yet, for multiple valued logic to be in wide use the key problem is to design a less complicated multiple valued memory [112]. In recent years, many ingenious multiple valued flip-flops (MVFFs) have been proposed [31][82][121][122][132].

Hence, multi-valued flip-flops may be a new approach to design low power flip-flops. However, as a first step, the structure should be explored.

Looking at previous research, it can be seen that multiple valued flip-flop designs are analogous to binary flip-flops and have the following three characteristics.

(1) Structurally, MVFFs have similar characteristics to their binary counterparts and usually are the extension of their binary version.

(2) Functionally, the next state behavior of the MVFFs are derived from their binary counterpart.

(3) In terms of working mode, the state transition is controlled by a binary clock pulse (CP), which is similar to their binary counterpart as well.

The third characteristic contradicts the aim of using multiple valued signals to increase information content. Since all signals used in multiple valued circuits are multiple valued, the clock signal should be multiple valued as well. For example, to a quaternary flip-flop, the clock signal should be $CP \in \{0, 1, 2, 3\}$ rather than a binary signal. In order to increase information content of the clock signal, ternary clock signal has been proposed by Wu [122]. There, besides the standard binary clock signal, an additional signal is used to implement the preset function. This effort, however, did not change the basic working mode of the MVFFs which are controlled by a binary clock signal. In this paper, a new MVFF, controlled by a multiple valued clock pulse,

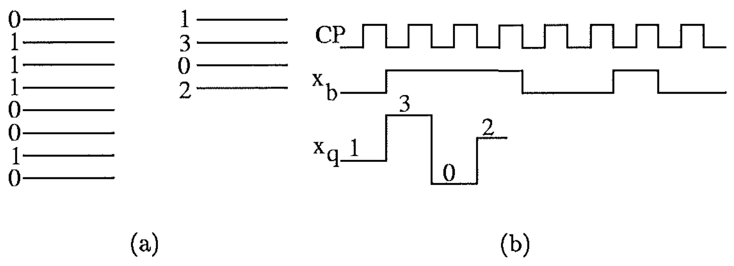


Figure 6.1: Comparison between binary and quaternary signal transmission. (a) Parallel transmission; (b) Serial transmission

is proposed.

6.1 Traditional Binary CP (BCP) controlled multiple valued master slave flip-flops in series

Similar to binary master slave flip-flops, the structure of multiple valued master slave flip-flops is composed of two multiple valued latches in series and controlled by two inter-inverting binary clock signals. The logic design of multiple valued latches can be classified by whether they have transmission switches or not [133]. If not, the logic structure can be implemented by crossing feedback from outputs to inputs. If yes, it can be implemented by employing the logic structure of buffers with controlling switches. A quaternary CMOS master slave D flip-flop is proposed by Prosser [82], which is composed of two quaternary buffers and two 2 to 1 multiplexers as in Fig. 6.2(a). Here, D is the inciting input, Q is the output, CP is the clock pulse, and Q, $D \in \{0, 1, 2, 3\}$, $CP \in \{0, 3\}$. Its working principal is: $CP = 3$, the input of the master latch receives the input signal D while the slave latch is in the storing state, and vice versa when $CP = 0$. Hence, the state transition of the flip-flop happens at the falling edge of CP $(3 \rightarrow 0)$ and the next state equation is $Q' = D$. Because it is composed of two quaternary latches in series and controlled by a binary CP, it is called a binary clock pulse (BCP) controlled quaternary master slave flip-flop in series. Fig. 6.2 (b) shows a binary shift register, which is composed of two D flip-flops of the type in Fig. 6.2 (a). The output of the shift register is expressed as: $Q'' = D$. It means that after two falling edges of CP the output equals the original inciting input signal D.

With this traditional design, a shift register of n digits is able to store n quaternary signals in series, and is composed of $2n$ quaternary flip-flops controlled by a binary CP.

6.2 Multiple valued *CP* (MVCP) controlled multiple valued flip-flops in parallel

Under the control of two levels in a *CP*, the master slave flip-flop shown in Fig. 6.2 (a) is in receiving and storing states alternately so that its "invalid toggle" is avoided. This meets the requirement of one state transition each *CP* for flip-flops. However, if two latches are connected in parallel rather than in series, then one latch receives the input signal while the other stores and outputs the stored signal. If *CP* is multiple valued such as a quaternary valued signal, the four levels can be used to select four latches to work alternately, that is, one of them is in the receiving state while the others are in storing states. This methodology is used to design multiple valued

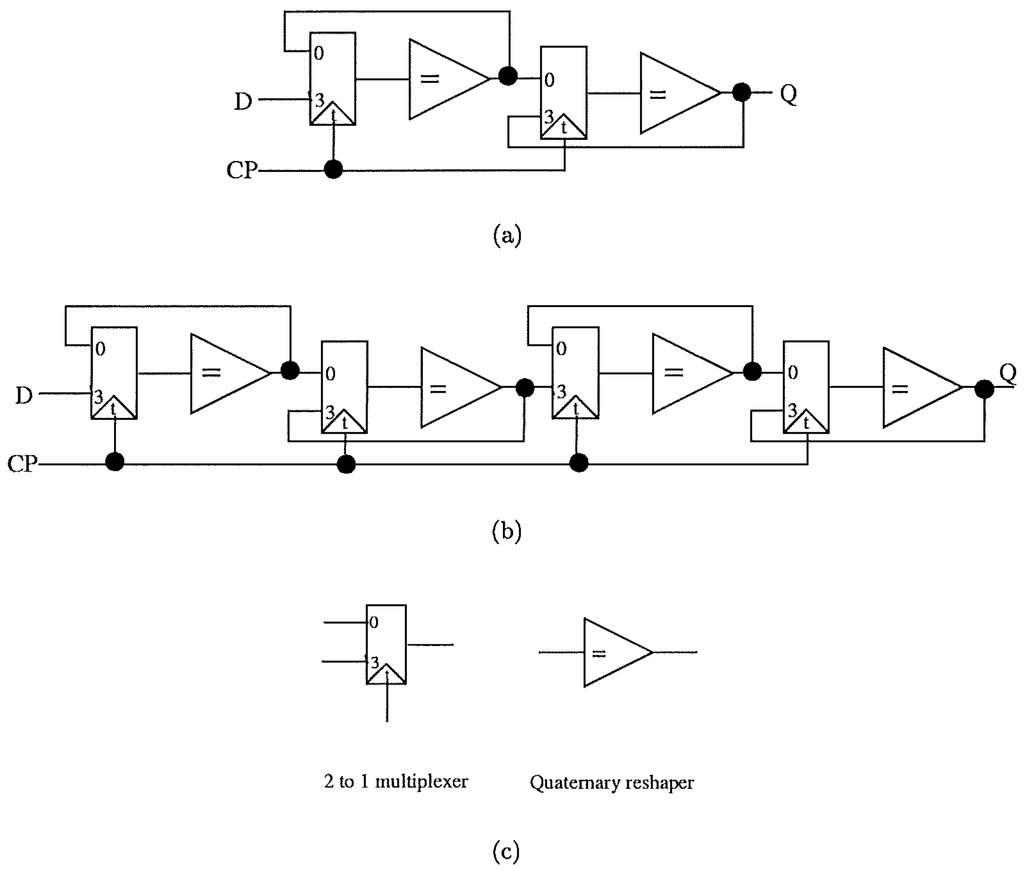
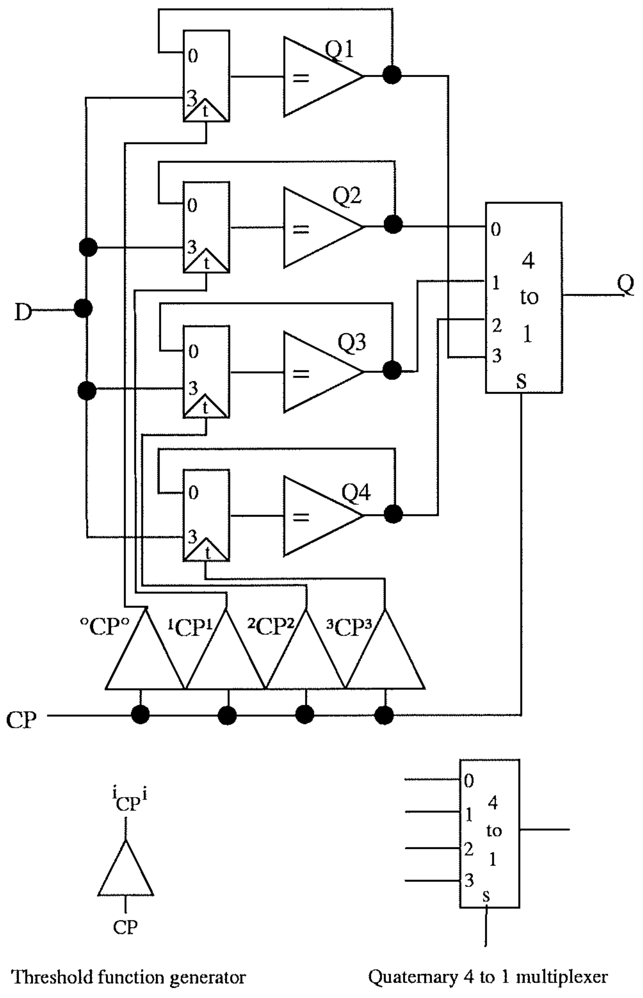


Figure 6.2: BCP controlled quaternary master slave D flip-flop in series and two digit quaternary shift register (a) BCP controlled quaternary master slave D flip-flops in series; (b) Two digit quaternary shift register; (c) Logic symbols

CP (MVCP) controlled multiple valued parallel flip-flops which are proposed in this chapter. Based on the above observation, a quaternary D flip-flop can be constructed as shown in Fig. 6.3 (a). It consists of four 2 to 1 multiplexers, four quaternary buffers, one quaternary threshold function generator and one 4 to 1 multiplexer. The implementation of those components can be found in the published literature [122][133][121]. Take the 4 to 1 multiplexer for example. If four transmission gates are controlled by four outputs of a quaternary threshold function generator, its circuit implementation is shown in Fig. 6.3 (b), where all inverters are binary.



(a)

Figure 6.3: Logic structure of MVCP controlled multiple valued flip-flop in parallel (a) Logic circuit; (b) Logic symbols

The working principal of the flip-flop in Fig. 6.3 (a) is: for any of the four values of the quaternary CP, one of the four latches receives the input signal D while the others are at storing states. The 4 to 1 multiplexer selects any one latch at storing state and its stored state becomes the output of the flip-flop. For the specific connection in Fig. 6.3 (a), when $CP = 0$, latch 1 receives the input signal and latch 2 outputs the stored signal; when $CP = 1$, latch 2 receives the input signal while latch 3 outputs the stored signal; when $CP = 2$, latch 3 receives the input signal and latch 4 outputs the stored signal; when $CP = 3$, latch 4 receives the input signal and latch 1 outputs the stored signal. Hence, if the clock pulse changes from $0 \to 1 \to 2 \to 3 \to 0$, the stored signal when $CP = 0$ can be output when $CP = 3$. Let Q_1, Q_2, Q_3, Q_4 be initial states for four latches, Q_1, Q_2, Q_3 and Q_4 respectively. During the first three upward level transitions of the CP, the outputs will be $Q2$, $Q3$, $Q4$, respectively. Therefore, the next state equation of the D flip-flop shown in Fig. 6.3 (a) is expressed as:

$$Q''' = D \tag{6.1}$$

The above equation states that the input signal will be output in three level transitions during one CP.

6.3 Circuit implementation

The circuit implementation principle of all the components in Fig. 6.3 can be found in the published literature [122][133][121]. However, their corresponding CMOS circuits will be implemented here based on transmission switch theory[122].

In the transmission switch theory, two algebraic systems are developed: Switching algebra to describe switching state with variable $\in\{T, F\}$ and signal algebra to describe signal value with variable $\in \{0, 1, 2, 3\}$ if quaternary valued logic is taken for example. Two sets of connection operations are introduced. The first set is for describing the physical process of how the signal controls the state of an element. They are defined as follows.

Definition 6.1. High-threshold comparison operation

$$ {}^{t}x \equiv \begin{cases} T & if\, x > t \\ F & if\, x < t \end{cases} \tag{6.2}$$

Definition 6.2. Low-threshold comparison operation

$$ x^{t} \equiv \begin{cases} T & if\, x < t \\ F & if\, x > t \end{cases} \tag{6.3}$$

where x is control signal and $t \in \{0.5, 1.5, 2.5\}$.

The second set is for describing how the state of a MOS transistor decides whether the source signal is transmitted to the drain or not.

Definition 6.3. Transmission operation

$$ c * \alpha \equiv \begin{cases} c & if\, \alpha = T \\ \Phi & if\, \alpha = F \end{cases} \tag{6.4}$$

where Φ describes the high-impedance state, c is called transmission source and α is the switching state of a transmission switch network.

Definition 6.4. Union operation

$$ c_1 * \alpha_1 \# c_2 * \alpha_2 \equiv \begin{cases} c_1 * \alpha_1 & if\, c_2 * \alpha_2 = \Phi \\ c_2 * \alpha_2 & if\, c_1 * \alpha_1 = \Phi \end{cases} \tag{6.5}$$

In equation 6.5, the case $c_1 \neq c_2$ and $\alpha_1 = \alpha_2 = T$ is not allowed because a voltage conflict arises between sources $c_1$and c_2. In addition, the transmission operation $*$ takes priority over the union operation $\#$. Based on the above definitions and some derived properties, the function expression can be obtained according to the logic functionality. Take a threshold 1.5 inverter for example. The logic functionality is that if $x < 1.5$ the output will be 3 while if $x > 1.5$ the output will be 0. Then, Taking threshold $t = 1.5$, low-threshold comparison operation and high-threshold comparison operation can be used to describe cases $x < 1.5$ & $x > 1.5$. Hence, based on definitions 6.4 & 6.5, the function expression is:

$$\bar{x}(1.5) = 3 * x^{1.5} \# 0 *^{1.5} x \tag{6.6}$$

Take case $x < 1.5$ for example to explain the above equation. When $x < 1.5$, $x^{1.5} = T$ while ${}^{1.5}x = F$; since ${}^{1.5}x = F$, $0*^{1.5}x = \Phi$; based on union operation, $\bar{x}(1.5) = 3*x^{1.5}$; since $x^{1.5} = T$, $\bar{x}(1.5) = 3$. According to the action principle of MOS transistors, when $x < 1.5$, a source voltage of 3 must be transmitted by a pMOS transistor while when $x > 1.5$ a source voltage of 0 must be transmitted by an nMOS transistor. Hence, the circuit realization of the threshold 1.5 inverter can be constructed as shown in Fig. 6.4. In Fig. 6.4, the number beside the transistor denotes its on threshold. Thus, the circuit implementation with pMOS and nMOS transistors easily follows based on the given function representation.

Similarly, the function expressions of all the logic components in Fig. 6.3 can be obtained as follows:

- Quaternary 2 to 1 multiplexer

$$y_{2to1} = y_0 * x^{1.5} \# y_1 *^{1.5} x \tag{6.7}$$

- Quaternary reshaper function

$$x_{res} = 0 * x^{0.5} \# 1 *^{0.5} x^{1.5} \# 2 *^{1.5} x^{2.5} \# 3 *^{2.5} x \tag{6.8}$$

- Threshold function ${}^{y}x^{y}$

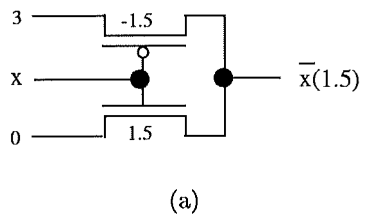
$${}^{y}x^{y} = \begin{cases} 0 & y \neq x \\ 3 & y = x \end{cases} \tag{6.9}$$

Here, x, $y \in \{0, 1, 2, 3\}$. Then, the function expressions are:

$${}^{0}x^{0} = 3 * x^{0.5} \# 0 *^{0.5} x \tag{6.10}$$

$${}^{1}x^{1} = 3 *^{0.5} x^{1.5} \# 0 * (x^{0.5} \#^{1.5} x) \tag{6.11}$$

$${}^{2}x^{2} = 3 *^{1.5} x^{2.5} \# 0 * (x^{1.5} \#^{2.5} x) \tag{6.12}$$



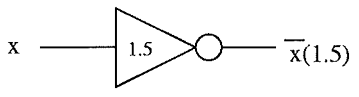


Figure 6.4: Threshold 1.5 inverter (a) Circuit implementation; (b) Logic symbol

$$^{3}x^{3} = 3 *^{2.5} x \# 0 * x^{2.5} \tag{6.13}$$

- Quaternary 4 to 1 multiplexer

$$y_{4to1} = y_0 * x^{0.5} \# y_1 *^{0.5} x^{1.5} \# y_2 *^{1.5} x^{2.5} \# y_3 *^{2.5} x \tag{6.14}$$

Their corresponding circuit implementations are shown in Fig. 6.5. However, for threshold functions, only the implementation of $^{0}x^{0}$ is shown to save space. In addition, for quaternary 4 to 1 multiplexer, if four threshold function signals from four threshold function generators are used, a simpler modified circuit can be obtained as shown in Fig. 6.5 (d). Then, the CMOS circuit of the flip-flop in Fig. 6.3 can be constructed.

6.4 Experimental results

The proposed flip-flop in Fig. 6.3 has been simulated using PSpice with a supply voltage of 5V for 0.5 μm technology. The SPICE parameters for a generic 0.5μm process are shown in Table 6.1.

Table 6.1: SPICE parameters for a generic 0.5μm process

.MODEL CMOSN NMOS LEVEL=3 PHI=0.7 TOX=10E-09 XJ=0.2U TPG=1
+VTO=0.65 DELTA=0.7 LD=5E-08 KP=2E-04 U0=550 THETA=0.27 RSH=2
+GAMMA=0.6 NSUB=1.4E+17 NFS=6E+11 VMAX=2E+05 ETA=3.7E-02
+CJ=5.6E-04 MJ=0.56 CJSW=5E-11 KAPPA=2.9E-02 CGDO=3.0E-10
+CGSO=3.0E-10 CGBO=4.0E-10 MJSW=0.52 PB=1
.MODEL CMOSP PMOS LEVEL=3 PHI=0.7 TOX=10E-09 XJ=0.2U TPG=-1
+VTO=-0.92 DELTA=0.29 LD=3.5E-08 KP=4.9E-05 U0=135 THETA=0.18
+RSH=2 GAMMA=0.47 NSUB=8.5E+16 NFS=6.5E+11 VMAX=2E+05
+ETA=2.45E-02 KAPPA=7.96 CGDO=2.4E-10 CGSO=2.4E-10 CGBO=3.8E-10
+CJ=9.3E-04 MJ=0.47 CJSW=2.9E-10 MJSW=0.505 PB=1

Four logic level voltages are set to: 5V (V_{DD}), 3.33V ($2V_{DD}/3$), 1.67V ($V_{DD}/3$), and 0V (V_{SS}).

For all nMOS devices, $W/L = 4\mu m/0.5\mu m$, while for all pMOS devices, $W/L = 8\mu m/0.5\mu m$. The PSpice simulation shows correct circuit operation as shown in Fig. 6.6.

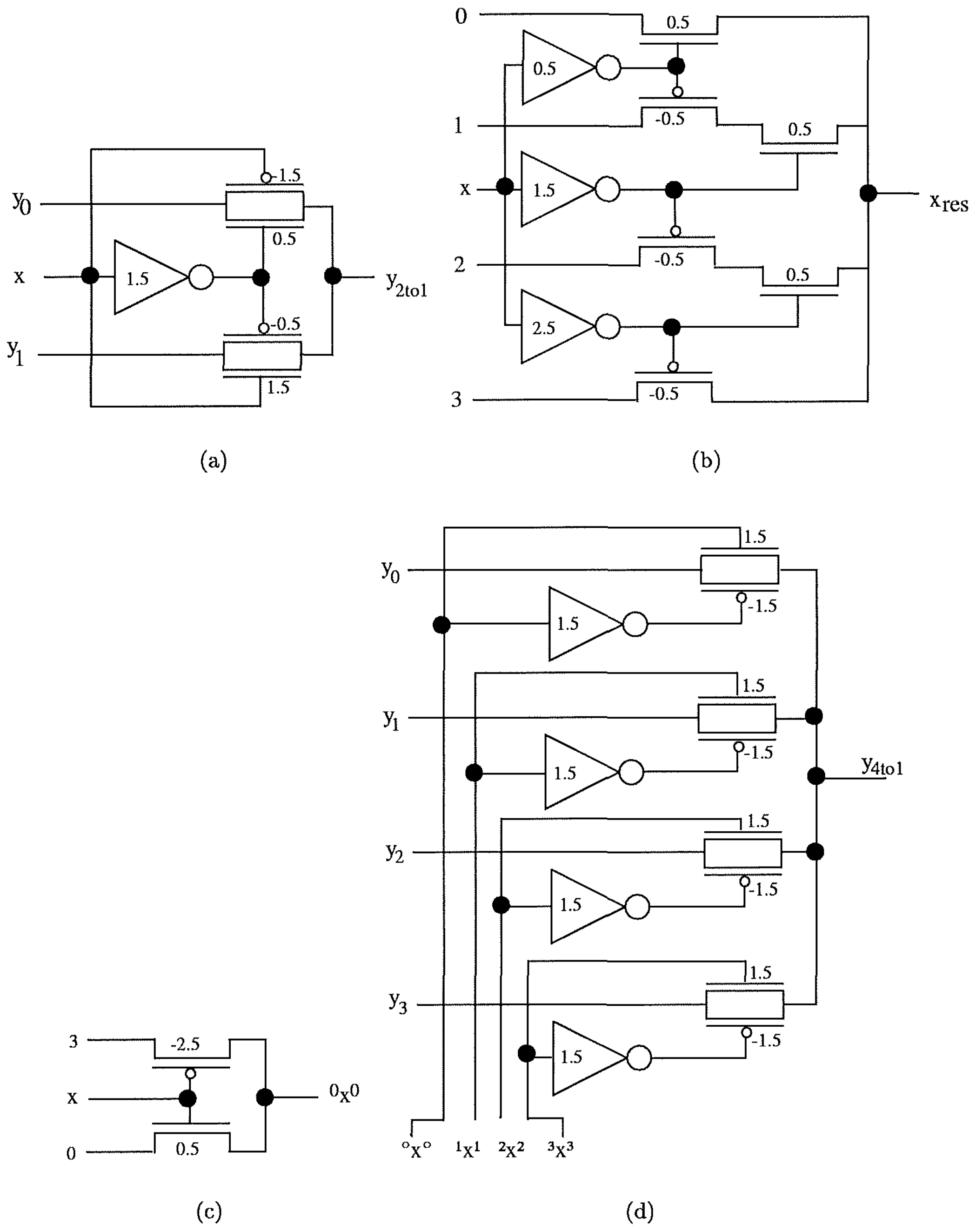


Figure 6.5: Circuit implementation (a) Quaternary 2 to 1 multiplexer; (b) Quaternary reshaper; (c) Threshold function generator; (d) Quaternary 4 to 1 multiplexer

The performance is measured when output is loaded with $C = 0.1picofarad$. The circuit with heavy capacitive load is to estimate its driving capabilities. The clock-to-Q time is measured from the 50% point of the clock logic-level transition of the associated output transition. The clock-to-Q time for the best-case output transition from logical-One to logical-Zero is $0.15ns$ while the time for the worst-cased from logical-Two to logical-One is $0.59ns$. The rising time and falling time are measured from the 10% point to 90% point of the transition level, which borrows from the conventional definition in the binary logic measurement. It can be seen that the falling time is much longer than the rising time for this design. Table 6.2 shows the average value.

Table 6.2: Performance measurement of the MVFF (ns)

Transition of logic levels	Clock-to-Q time	Rising time	Falling time
$0 \leftrightarrow 1$	0.25	0.14	0.34
$1 \leftrightarrow 2$	0.42	0.05	0.14
$2 \leftrightarrow 3$	0.29	0.06	0.06
$0 \leftrightarrow 2$	0.28	0.24	0.17
$1 \leftrightarrow 3$	0.27	0.08	0.25
$0 \leftrightarrow 3$	0.33	0.08	0.11

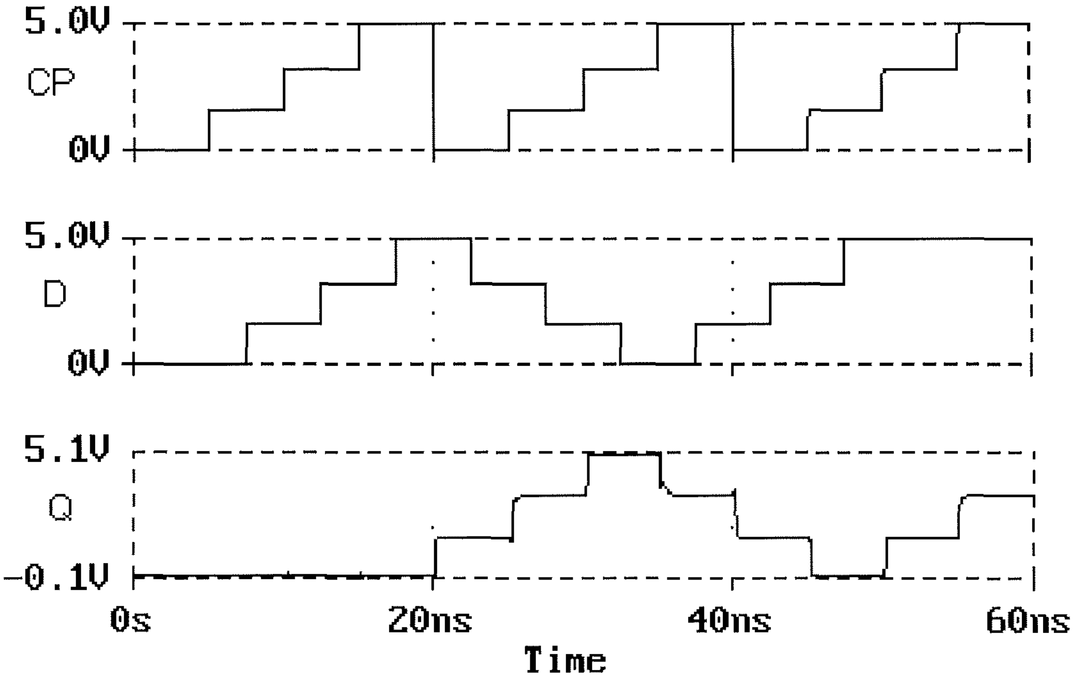


Figure 6.6: PSpice simulation of the proposed flip-flop

A comparison for the storage capacity between the MVCP controlled quaternary D flip-flop in Fig. 6.3 and the two-digit shift register in Fig.6.2 (b) can be made. They use the same number of latches. However, the former can output a four digit quaternary signal in one clock cycle while the latter can only output a two digit quaternary signal in two clock cycles. The reason is that the proposed flip-flop can receive input signal when the quaternary CP takes any of its four values while the latter can only do so when $CP = 3$.

Average power dissipation is measured with 5V and 50MHz clock for the proposed circuit and evaluated for a time period of 200ns, which is 1.49mW. For the fair comparison of the proposed circuit and that in Fig. 6.2, the same PSpice simulator and 0.5um technology are used to simulate the circuit in Fig. 6.2. The power dissipation is 2.56mW with 5V and 200MHz clock. The reason of using 200MHz clock is that BCP based flip-flop samples the input date only when $CP = 3$ and the same data throughput as that of the proposed circuit is required. The power reduction is 41.3% with the area trade of 29.8%.

Table 6.3: Area and Power dissipation for the proposed circuit and the circuit in Fig. 6.2(b)

	No of Transistors	Power dissipation (mW)
Circuit in Fig. 6.2(b)	80	2.56
Proposed circuit	114	1.49
Improve (%)	-29.8	41.8

6.5 Summary

In this chapter, the present BCP controlled multiple valued master slave flip-flops in series are analyzed first. Then MVCP controlled multiple valued flip-flop in parallel is proposed. Based on transmission switch theory, the CMOS circuit of the proposed flip-flop is implemented. PSpice simulation shows that it has the correct operation. Although only the quaternary D flip-flop is designed, the result can be extended to the design of ternary and higher radix flip-flops. This design changes the traditional design method of multiple valued flip-flops. It uses each level of a quaternary clock signal to receive an input signal, which makes the logic structure of multiple valued flip-flops more flexible. The proposed flip-flop has the following characteristics:

1. Employing multiple valued clock signal ensures the consistency of processing signals for

multiple valued flip-flops, which eliminates the requirement to provide the same circuit with both multiple valued signals and binary signals.

2. An n-valued CP controlled n-valued flip-flop can shift out an n-digit n-value signal during one upward clock cycle while a BCP controlled multiple valued flip-flop can only shift out one-digit n-valued signal during one binary clock cycle. The proposed circuit is characterized by reduced power dissipation.

3. Because there are possibly multiple connections between each latch and the output multiplexer, various logic functions can be obtained by changing the connections, which adds flexibility to the design of the flip-flops.

Chapter 7

Low power XOR gate decomposition

7.1 Introduction

With the rapid increase in the density and the size of chips and systems, power dissipation is becoming critical concern in VLSI design. Since CMOS technology is predominant in the realization of today's ICs and CMOS devices are intrinsically low-power consuming, CMOS has become the reference technology. Power dissipation in digital CMOS circuits is dominated by dynamic dissipation, which results mainly from the charging and discharging of the node capacitances [89]. It can be modeled as in equation 7.1

$$Power = \frac{1}{2}V_{dd}^2 f_{clk} \sum_i C_i SA_i \tag{7.1}$$

where V_{dd} is the supply voltage and f_{clk} is the clock frequency which are determined by the technology, C_i is the physical capacitance at the output of the node (i is the node number) and SA_i (referred to as the switching activity) is the expected number of output transitions per clock cycle[131]. The summation is taken over all nodes of the logic circuit. As is known, C_i and SA_i can be optimized during the design process.

Logic functions can be expressed canonically based on AND/OR (NAND & NOR) operators. Extensive research has been done on developing low-power techniques in AND/OR or NAND

& NOR based circuits [91]. However, in certain applications, XOR realizations have attractive advantages over the conventional AND/OR NAND&NOR logic especially for functions which do not produce optimization solutions based on these operators. The XOR circuit is easier to test and may require fewer gates and interconnections [3]. Unfortunately, the techniques for synthesis and optimization of logic based on Reed-Muller are much more difficult than those based on the standard Boolean expressions.

With recent development and increasing use of various field programmable gate array (FPGA) devices[21] where the XOR gate is already manufactured as a basic component, synthesis and optimization of XOR gate based circuits are receiving much more attention than before. More recently, there has been some success in achieving area reduction by employing optimization techniques specifically targeted towards initial AND/XOR representations in the well known Fixed Polarity Reed Muller (FPRM) form [114][107]. The low power optimization of XOR gate based circuits is still in its infancy.

In technology mapping, the subject netlist is usually first decomposed into a netlist composed of only inverters and two-input XOR gates. The decomposition can have a significant impact on the power dissipation of the final implementation [74][131]. Looking at the existing work done by Narayanan and Liu [74] and Zhou and Wong [131], the former is for static logic and the latter is for dynamic logic. However, as we will show later, the algorithms in [74][131] are not optimal. In this chapter, we will deal with the low power XOR gate decomposition problem.

The rest of this chapter is organized as follows. In section 7.2, some definitions and terminology are introduced. In Section 7.3, some previous work is reviewed. Section 7.4 presents a new algorithm for low power XOR gate decomposition while Section 7.5 shows the experimental results. Finally, a brief summary is given in Section 7.6.

7.2 Definitions and terminology

To implement a complex function involving many inputs, a form of decomposition into smaller subfunctions is required such that the subfunctions fit into the primitive elements to be used in the implementation.

Definition 7.1. Low power XOR gate decomposition: a multi input XOR gate is decomposed into a tree of two-input XOR gates, which is of optimal power dissipation.

A tree is defined by the following definition.

Definition 7.2. Let $T = (V, E)$ represent a decomposition tree, and $p_r(v)$, for any $v \in V$, denotes the output signal probability of node v. Each node has two children. The primary inputs are called leaves of the tree and the primary output is called the root of the tree.

Each primary input signal into a tree is treated as a random variable and its probability is defined as follows.

Definition 7.3. Signal probability for signal x is defined as the probability of x being 1.

To estimate the power dissipation of a tree, according to equation 7.1, C_i and SA_i need to be known. Each node in the tree corresponds to a two-input XOR gate. Suppose C_i is constant for each two-input XOR gate. Then, the only parameter we need to compute is SA_i.

Under the assumptions of zero delay model with temporal and spatial independence [74][131], given input signal probabilities and a decomposition tree, the probabilities of internal signals can be computed according to the following definition of XOR operator.

Definition 7.4. XOR operator: $f(x_1, x_0) = x_1 \oplus x_0 = x_1\overline{x_0} + \overline{x_1}x_0$ if x_1 and x_0 are two input variables. Hence, the probability of $f(x_1, x_0)$ is as shown in equation 7.2 [74][131]:

$$P_{r,f} = P_r(p_{r,x_1}, p_{r,x_0}) = p_{r,x_1} + p_{r,x_0} - 2p_{r,x_1}p_{r,x_0} \tag{7.2}$$

The calculation of the corresponding switching activity depends on whether the circuit is implemented static logic (which does not take into account the timing behavior and is strictly a function of the topology and the signal statistics) or in dynamic logic (which takes into account the timing behavior of the circuit).

For static logic, the transition probability assuming independent inputs is the probability that the output will be in the zero state multiplied by the probability that the output will be in the one state. For example, if the signal probability being zero is p_0 and p_1 represents the signal probability being one, the transition probability from zero to one is $p[0 \to 1] = p_0 p_1 = p_0(1-p_0)$. In general, for a digital signal, $p[0 \to 1]$ is equal to $p[1 \to 0]$. Therefore, the switching activity for gate i can be expressed as 7.3 [131]

$$SA_i = P_{r,i}[0 \to 1] + P_{r,i}[1 \to 0] = 2P_{r,i}(1 - P_{r,i}) \tag{7.3}$$

The main difference between static logic and dynamic logic is that dynamic CMOS works using a precharging circuit. The output of a circuit can be precharged high or low then the resulting value is evaluated in each clock cycle. Therefore, the output transition probability does not depend on the sate of the inputs but rather on just the signal probabilities[24]. There are two implementation models, which are precharged to 0 or 1 and evaluated to the result value. Take the former for example, that is, a gate output is first precharged to 0 at the leading edge of the clock signal and then evaluated to the result value at the falling edge. Therefore, if the output probability is $P_{r,i}$, the switching activity for gate i will be $2P_{r,i}$ where one $P_{r,i}$ comes from the precharging and one $P_{r,i}$ comes from the evaluation.

Then, for dynamic logic, the switching activity for gate i is given by equation 7.4 [89]

$$SA_i = 2P_{r,i} \tag{7.4}$$

Fig. 7.1 shows the switching activity as a function of signal probability both in dynamic logic and static logic. It can be seen that if only considered in term of switching activity, static logic is always better than dynamic logic. If the two input signal probabilities of an XOR gate are $p_{r,x_0} = 0$ and $p_{r,x_1} = 1$ respectively, then the output signal probability of XOR gate is $P = 1$ according to equation 7.2. One transition, evaluated to 1 at the falling edge of the clock, is followed by another transition, precharged to 0 at the leading edge of the clock. Hence, in this particular case, the switching activity of an XOR gate is equal to that of a clock signal, which is 2. For static logic, the maximum switching activity is 0.5 when signal probability is 0.5. However, in terms of power dissipation, the same conclusion can not be deduced from switching activity since dynamic logic usually has smaller load capacitance.

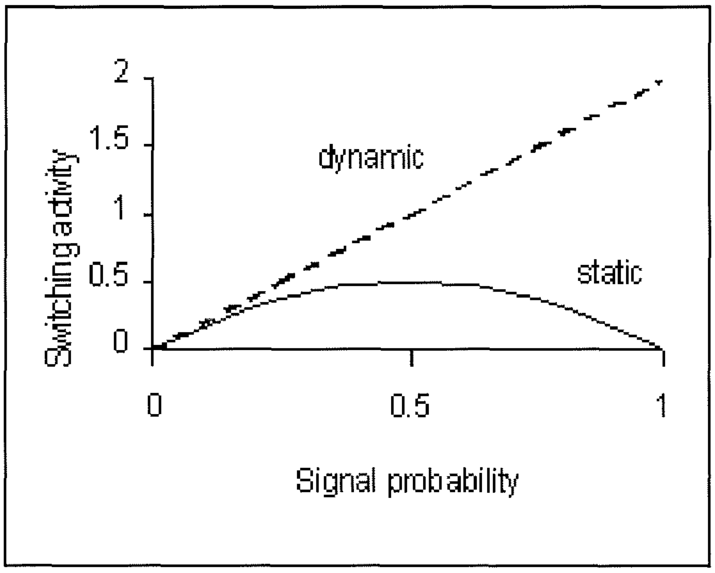


Figure 7.1: Switching activity versus signal probability

Given an n-input XOR gate with primary input signals $I = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and corresponding signal probabilities $\{p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}\}$, based on the above discussion, the procedure for the power estimation of the XOR gate decomposition is as follows.

1. Given the primary input signal probabilities, construct a tree of two input XOR gates.
2. Compute the output probability for each gate.
3. Compute the node switching activity using equations 7.4 or 7.3.
4. Compute the total switching activity using equation 7.1.

Then, the problem for low power XOR gate decomposition can be described as follows.

Low power XOR gate decomposition: Given an n-input *XOR* gate with input signals $I = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and corresponding signal probabilities $\{p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}\}$, two input XOR gates are used to construct a tree $T = (V, E)$ with $p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}$ as its leaves such that equations 7.5 and 7.6 are minimized for dynamic logic and static logic respectively.

For dynamic logic,

$$SA = \sum_{i \in V-I} 2P_{r,i} \tag{7.5}$$

For static logic,

$$SA = \sum_{i \in V-I} 2P_{r,i}(1 - P_{r,i}) \tag{7.6}$$

Where SA is the total switching activity and $P_{r,i}$ is the signal probability of an internal node.

7.3 Previous work

In [131], the authors analyzed some optimal properties and proposed an algorithm to solve the problem of lower power XOR gate decomposition in dynamic logic.

Algorithm 7.1. *Zhou - Wong's algorithm*

==

First, sort the input probabilities which are greater than 0.5.

Then, combine them pairwisely from the largest to the smallest.

Finally, iteratively combine the two signals with the smallest probabilities until there is one signal.

Based on the analysis of XOR operation and assuming that the signals and their complements are available, Naraynana and Liu [74] presented an algorithm as follows.

Algorithm 7.2. *Naraynana - Liu's algorithm*

==

Let k be the number of primary inputs to the XOR tree with probabilities greater than 0.5.

If k is even, replace all k inputs by their respective complements.

If k is odd, replace any $k-1$ inputs by their respective complements.

Then simply use the Huffman algorithms to build the optimal tree.

Take a simple example to show how their algorithms work in Fig. 7.2. The problem has three inputs and corresponding signal probabilities are 0.1, 0.45 and 0.9 respectively. Both algorithms share the same decomposition in Fig. 7.2(b). The total switching activities are 1.984 from equation 7.5. However, if the assumption that both the signals and their complements are available is preserved, the above decomposition is not optimal. Suppose signal x_2 and x_1 are replaced by their complements and it can be proved that the functionality is still preserved. Then, there is another decomposition as shown in Fig. 7.2(c) with a total switching activity of 1.424, which is smaller. It is realistic to assume that both signals and their complements are available like FPGA mapping. Hence, in the chapter, all discussions are based on this assumption.

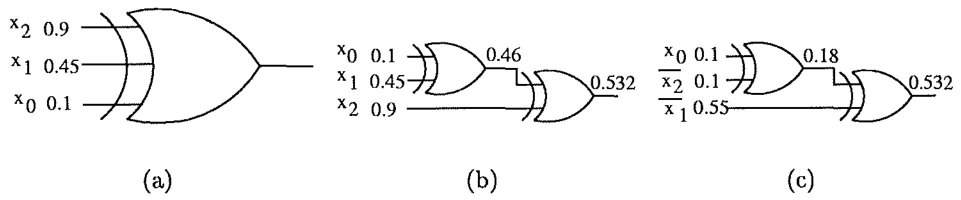


Figure 7.2: (a) Multi input XOR gate; (b) Published decomposition in [131] and [74]; (c) A new decomposition

Actually, both the Naraynana-Liu's algorithm and the new decomposition in Fig. 7.2(c) use the polarity transformation. However, any n-variable Boolean function can be expressed canonically by the Fixed Polarity Reed-Muller (FPRM) form as follows:

$$f(x_{n-1}, x_{n-2}, \cdots, x_0) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i \tag{7.7}$$

The π- terms can be represented as $\pi_i = \dot{x}_{n-1}\dot{x}_{n-2}\cdots\dot{x}_0$, the subscript i can be written as a binary n-tuple $i = (i_{n-1}i_{n-2}\cdots i_0)$, $b_i \in \{0, 1\}$, $\dot{x}$ is a literal which can be x or $\bar{x}$,

$$\dot{x}_j = \begin{cases} 1, \, i_j = 0 \\ \dot{x}_j, \, i_j = 1 \end{cases} \tag{7.8}$$

and $0 \leq j \leq n-1$.

An FPRM form of a Boolean function can be identified by a polarity p, $p = (p_{n-1}p_{n-2}\cdots p_0)$. $p_j \in \{0,1\}$. $p_j = 0$ if x_j is true otherwise $p_j = 1$. For FPRM forms, every variable can only be either true or complemented but not both. Therefore, there are 2^n polarities for an n-variable function.

Definition 7.5. If $p = (p_{n-1}p_{n-2}\cdots p_0)$ and $q = (q_{n-1}q_{n-2}\cdots q_0)$ are the two polarities of an n variable function under FPRM forms. If the number of binary bits that are 1s is even, it is called even polarity, otherwise it is called odd polarity. If all variables appeared in equation 7.7 are in true forms, the polarity of expansion is known as positive polarity or polarity zero.

Obviously, an n variable XOR function is a special case of an n variable FPRM form where the number of π-terms equals to the number of variables and $\pi_i = x_i$ and its polarity is zero. For example, an $n-$variable XOR function under polarity zero can be expressed as

$$f(x_{n-1}, x_{n-2}, \cdots, x_0) = x_0 \oplus x_1 \oplus x_2 \cdots \oplus x_{n-1} \tag{7.9}$$

7.4 Proposed low power XOR gate decomposition

Dynamic logic is used to illustrate the technique. For dynamic logic, according to equation 7.5, minimizing the total switching activity can be achieved by minimizing the total internal signal probabilities.

In terms of polarities, the algorithm proposed in [131] is optimal only under polarity zero. Although the authors in [74] tried to use polarity transform to optimize power dissipation, they only tried one of 2^n polarities, because, for given input signal probabilities the selected polarity is fixed according to their algorithm. For example, for a three input problem shown in Fig. 7.2, the selected polarity is zero. For a five input problem whose input signal probabilities are 0.1, 0.2, 0.55, 0.7 and 0.9, the selected polarity using the algorithm in [74] is 3 or 5. As we know, different polarities have different function forms, which result in different power dissipation as shown in Fig. 7.2. Hence, if 2^n polarities are searched, the best polarity for low power decomposition can be found. However, for reasons that will become apparent in Section 7.4.1, the search space can be reduced to $2^{n-1} - 1$.

7.4.1 Theoretical results

For an n variable function, there are 2^n polarities and correspondingly there are 2^n FPRM forms. FPRM forms can be transformed from one to the other with polarity transformations [114]. From definition 7.5 and starting from positive polarity, all polarity transformations can be categorized into two classes: Odd Polarity Transformation (OPT) and Even Polarity Transformation (EPT).

Lemma 7.1. *Suppose there is an XOR function consisting of n inputs based on equation 7.9. If an OPT is selected, one more $\pi - term$,1, is added while if an EPT is selected, the number of $\pi - terms$ remains the same.*

This is trivial because $\dot{x} \oplus 1 = \bar{\dot{x}}$ and $\dot{x} \oplus \dot{x} = 0$. Take the three inputs in Fig. 7.2 for example. If transformations of polarities 2 (010), odd polarity, and 6 (110), even polarity, are derived from the positive polarity form, the function forms are shown as follows:

$$f(x_2, \bar{x}_1, x_0) = 1 \oplus x_0 \oplus \bar{x}_1 \oplus x_2 \tag{7.10}$$

$$f(\bar{x}_2, \bar{x}_1, x_0) = x_0 \oplus \bar{x}_1 \oplus \bar{x}_2 \tag{7.11}$$

Form Lemma 7.1, there is a following corollary.

Corollary 1. *For any XOR tree T consisting of n inputs, the OPT requires one additional input node compared to the EPT.*

From Lemma 7.1, Corollary 1 is obvious because for a tree of two input XOR gates, the additional input needs one more two-input XOR gate. Hence, EPT can preserve the size of an XOR tree between polarity transformations.

Theorem 7.1. *Given an n variable function, there are one zero polarity, $2^{n-1}-1$ even polarities and 2^{n-1} odd polarities [126].*

This can be proved by induction on n as below.

(1) When n is 1, there are two polarities 0 and 1. Obviously, it satisfies the Theorem 7.1.

(2) Suppose when n is k, $k > 1$, polarity $p' = (p_{k-1}p_{k-2}\cdots p_0)$ and there are one polarity zero, $2^{k-1}-1$ even polarities and 2^{k-1} odd polarities and Theorem 7.1 is true. When n is $k+1$, polarity $p = (p_k p_{k-1}\cdots p_0)$. (a) When p_k is 0, the distribution of polarities is the same as the case $n = k$. Hence, there is one polarity zero, $2^{k-1}-1$ even polarities and 2^{k-1} odd polarities. (b) When p_k is 1, then $p = 2^k + p'$ where "+" is the arithmetic addition. Compared to the p', the polarity distribution changes as follows: Polarity zero$\Rightarrow$odd polarity; $2^{k-1}-1$ even polarities $\Rightarrow 2^{k-1}-1$ odd polarities; 2^{k-1} odd polarities $\Rightarrow$ 2^{k-1} even polarities. Hence, there are 2^{k-1} even polarities and 2^{k-1} odd polarities. From both (a) and (b), Theorem 7.1 is true when n is $k+1$.

Corollary 1 and Theorem 7.1 tell us that considering area constraints, EPT is preferred and the best polarity could be found by searching $2^{n-1}-1$ even polarities.

For any specified polarity, all input signal probabilities can be deduced from the input signal probabilities under polarity zero. However, given a set of input signal probabilities, it must be one of following three cases.

Case 1: At most one input signal probability is greater than 0.5.

Case 2: All input signal probabilities are greater than 0.5.

Case 3: Some are greater than 0.5 and some smaller than 0.5.

For each case, the optimal solution for low power XOR gate decomposition is to find out an optimal combination of input signals with the merging function in equation 7.2 such that the total internal probability is minimized. For Cases 2 and 3, the theorem in [131] is used to combine input signal probabilities greater than 0.5 pairwisely.

Theorem 7.2. *(Zhou and Wong) In an optimal XOR gate decomposition, all inputs of probabilities greater than 0.5 are combined pairwisely from largest to smallest.*

Then, using the above theorem, Cases 2 and 3 can be transformed into Case 1. For Case 1, it is shown that under some conditions a variation of Huffman' s algorithm, which is a little bit different from the original one, can give an optimal solution[74][131]. It is defined as follows.

Definition 7.6. Mini-Huffman algorithm: [107]Start from all the input signals; combine the two signals of minimum probabilities and substitute them with a new signal; continue the process until there remains only one signal.

7.4.2 Proposed algorithm

Based on the above properties, a new algorithm to solve the low power XOR gate decomposition problem is proposed as follows. First, according to input signal probabilities given with a specified polarity, an optimal tree for low power XOR gate is derived based on Theorem 7.2 and Mini-Huffman algorithm, as described in Procedure 7.1 and corresponding pseudo-code in Algorithm 7.3.

Procedure 7.1. *Given the input signal probabilities set for an n variable XOR function with polarity p, carry out Steps 1 to 3:*

Step 1. If the problem belongs to Case 1: At most one input signal probability is greater than 0.5, use mini-Huffman algorithm to combine the signal probabilities and construct a tree and compute the total switching activity;

Step 2. If the problem belongs to Case 2 or Case 3, according to Theorem 7.2, combine the signal probabilities using equation 7.2 until Case 2 or Case 3 is changed into Case 1. Then, go to Step 1;

Step 3. Output the tree and its total switching activity.

Then, other polarities are calculated and their corresponding switching activities are compared to find the best polarity which has the minimum switching activity. However, based on Theorem 7.1, only $2^{n-1} - 1$ even polarities need to be computed. This is shown in Procedure 7.2 and corresponding pseudo-code is given in Algorithm 7.4.

Procedure 7.2. *Given the input signal probabilities set for an n variable XOR function with polarity zero, set bestPolarity, SA_p and OptSA to be 0, respectively. For any p, $0<p\leq 2^n-1$, carry out Steps 1 to 5.*

Step 1. Use Procedure 7.1 to compute switching activity (SA_0) with polarity zero. Let OptSA $= SA_0$;

Step 2. Let p = p+1;

Step 3. If p is OPT, go to Step 2. If p is EPT, use Procedure 7.1 to compute SA_p with polarity p.

Step 4. If SA_p is less than OptSA, let OptSA$=SA_p$ and bestPolarity=p. If $p\leq 2^n-1$,go to Step 2. Otherwise, go to Step 4.

Step 5. Output OptSA and bestPolarity.

Algorithm 7.3. *Pseudo-code of Procedure 7.1*

```
===================================================
Input: a set of signals $X = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and corresponding probabilities
$\{ p'_{r,x_{n-1}}, p'_{r,x_{n-2}}, \cdots, p'_{r,x_0} \}$ under polarity p
Output: a decomposition tree T
Decomp(X)
{
    if |X| ==1
    return $p'_{r,x_{n-1}}$;
    else
    {
      if(Case 1)
        {
          $T = mini\_Huffman(X)$;
          return T;
        }
      if(Case 2 or Case 3)
        {
                $S = \{s_i = combine(x_{n-i}, x_{n-i-1})\}, (x_l \in X_1 \leq X)$; //$X_1$ is a set of primary
input signal probabilities greater than 0.5
                $T = mini\_Huffman(X + S - X_1)$;
                return $(T + S)$;
            }
    }
```

Algorithm 7.4. *Pseudo-code of Procedure 7.2*

```
======================================================
Input: a set of signals $X = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and $\{p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}\}$ under
polarity zero
Output: OptSA and bestPolarity //OptSA-the smallest switching activity for a tree; bestPolarity-
the polarity corresponding
/* Decompose under zero polarity*/
bestPolSearch(X)
{
    if |X|==1
    $OptSA = p_{r,x_{n-1}}$;
    else
    {
      OptSA=Decomp(X)·SA; //SA - switching activity
    }
bestPolarity=0; //bestPolarity - the polarity corresponding OptSA is named bestPolarity
$polarityNo = 1 << n$; //The number of polarities for an n - input variable function is named
polarityNo
for $(p = 1; p < polarityNo; p++)$
  {
    if$(p\%2 == 1)$;
```

```
        else
          {
            //decomp |X| under polarity p
            $SA_p = Decomp(X) \cdot SA$; //$SA_p$- switching activity under polarity p
          }
        if( $SA_p < OptSA$)
          {
            $OptSA = SA_p$;
            bestPolarity = p;
          }
      }
    Output OptSA and bestPolarity;
    }
```

7.5 Experimental results

The proposed algorithm is implemented in C and compiled by the GNU C compiler. The time complexity for the mini-Huffman procedure is $O(nlogn)$[131]. The proposed algorithm has a time complexity of $O(2^n nlogn)$ where n is the number of variables. In order to compare our results with those from other algorithms, Zhou-Wong's algorithm and Narayanan-Liu's algorithm are also implemented. All the three algorithms are tested on a personal computer with PIII 550 CPU and 64M RAM under Linux operating system. Two sets of experiments have been carried out.

First, given a set of input signal probabilities, the best polarity (the lowest switching activity) is found and its corresponding switching activity is computed and compared to that under polarity zero to see the efficiency. Table 7.1 shows the experimental results for input size from 3 to 13 variables. Thirteen input signal probabilities are generated under randomly selected seed number 500, which are 0.98, 0.57, 0.17, 0.70, 0.71, 0.18, 0.03, 0.14, 0.69, 0.70, 0.81, 0.32, 0.63, respectively. For different input sizes, the input signal probabilities are chosen from left to right in the above list. For example, for three inputs, the signal probabilities are 0.98, 0.57 and 0.17 while for four inputs they are 0.98, 0.57, 0.17, 0.70, and so on. In Table 7.1, Column 2 from left shows the best polarity and its corresponding switching activity SA_p, Column 3 shows the switching activity SA_0 decomposed under polarity zero and last column shows the improvement rate which is defined as follows:

$$Improv = \frac{SA_0 - SA_p}{SA_p}\% \tag{7.12}$$

Table 7.1: Best polarity and switching activity for low power decomposition

INs	Best Decomp		Polarity Zero	Improv
	Best Polarity	SA_p	SA_0	(%)
3	3	0.6388	0.8884	39.07
4	9	1.0742	1.1990	11.62
5	27	1.4895	1.6299	9.43
6	27	1.8560	1.9693	6.10
7	3	1.9399	2.1771	12.23
8	3	2.2620	2.5049	10.74
9	257	2.7286	2.9525	8.21
10	3	3.1781	3.4198	7.61
11	305	3.5494	3.6889	3.93
12	305	4.0205	4.1610	3.49
13	2355	4.4925	4.6424	3.34

Decomposed with the best polarity, it can be seen that the improvement could be as high as 39% compared to the result with polarity zero for this randomly selected instance.

Second, for each input size from 3 to 13 variables, 100 sets of input probabilities are randomly generated and then run on each of the above algorithms. For each instance X, let $Opt(X)$ represents the optimal solution, which has the lowest switching activity from the three algorithms. The following parameter defined in equation 7.13 is used to measure the performance of solution $S(X)$.

$$R = \frac{S(X) - Opt(X)}{Opt(X)}\% \qquad (7.13)$$

From the definition in 7.13, obviously, the higher the rate R is, the worse the performance of the algorithm will be. For each algorithm, among all 100 problems of the same input size, the number of non-optimal solutions, the maximum ratio and the average ratio are computed. They are represented by $NonOpt$, MR and AR, respectively. MR is to measure the worst performance among all 100 problems of the same input size while AR is to measure the average poor performance. The average CPU time is also reported in Table 7.2.

Table 7.2: Experimental results for the proposed algorithm and published algorithms

INs	Narayanan-Liu [74]			Zhou-Wong [131]			Proposed	
	Non	MR	AR	Non	MR	AR	Non	time
	Opt	(%)	(%)	Opt	(%)	(%)	Opt	$10^{-2}s$
3	41	46.04	7.79	50	69.77	10.87	0	~0
4	40	44.41	8.11	72	40.58	8.88	0	0.1
5	44	35.75	8.57	89	28.39	8.28	0	0.3
6	53	34.43	9.08	92	28.18	7.42	0	0.8
7	51	25.04	6.76	92	24.10	5.59	0	1.0
8	45	27.43	6.31	95	20.01	4.91	0	1.9
9	48	22.50	5.90	98	14.47	4.38	0	4.2
10	58	18.31	6.80	98	7.25	4.00	0	9.7
11	56	15.17	6.09	99	5.28	3.63	0	21.8
12	54	13.37	5.88	99	4.79	3.56	0	48.6
13	47	11.77	5.08	100	4.06	3.27	0	108.0

From the results reported in Table 7.2, the number of non-optimal solutions for the proposed algorithm is always zero in comparison with Narayanan-Liu's and Zhou-Wong's algorithms. In other words, the switching activity from the proposed algorithm is always smaller than that from Narayanan-Liu's or Zhou-Wong's algorithm. It can be seen that the average poor performance is decreased with the increase of input size for Zhou-Wong's algorithm but this does not happen for Narayanan-Liu's algorithm. For smaller input size like 3 and 4 variables, Narayanan-Liu's algorithm is better than Zhou-Wong's algorithm. The reason is that in this case the number of polarities is smaller and there is a higher chance for Narayanan-Liu's algorithm to find a good polarity than Zhou-Wong's algorithm. For example, for three inputs: 0.10,0.80,0.90, using Narayanan-Liu's algorithm the polarity is 3 and the switching activity is 0.488 while using Zhou-Wong's algorithm it is 0.568 under polarity zero.

7.6 Summary

Using the polarity transform under the assumption that both signals and their complements are available, a new solution to decompose low power XOR circuits is proposed. The algorithm is implemented in C. The time complexity of the algorithm is $O(2^n nlogn)$. The improvement of switching activity is significant compared to published results. It should be pointed out that the algorithm can be applied to static logic as long as the calculation formula of switching activity in equation 7.5 is substituted by the static logic formula in equation 7.6.

Chapter 8

Power and area optimization of FPRM functions

In the past, the task of VLSI designers has been to explore the area-time implementation space. In recent years, power is being given comparable weight to area and time [28].

Up to now, most researchers have focused on developing optimization techniques for area and power dissipation in AND/OR or NAND & NOR based circuits [91]. However, in certain applications, XOR realizations have advantages over the conventional AND/OR or NAND & NOR logic. XOR circuits are easier to test and may require fewer gates and interconnections [3][87]. Applications of Reed-Muller circuits have so far not become popular due to the following two obstacles.

1. XOR gates used to have slow speed and require large silicon area to realize in comparison with OR gates.

2. The problem of optimization of Reed-Muller functions is difficult although there has been a great deal of research in recent years [64][114].

With the development of new technologies and the advent of various field programmable gate array (FPGA) devices [21], the first obstacle has been solved. In programmable devices, the XOR gate is either easily configured in "universal modules" or directly available. For example, logic blocks can be configured as various two-input gates such as XORs, ANDs and NANDs in the AT6000 FPGA series from ATMEL Corporation while in other FPGAs, both AND and XOR gates are available in the macrocells or logic array blocks [76]. For the second obstacle, more recently, there has been some success in achieving area reduction by employing optimization techniques specifically targeted towards initial AND/XOR representations in the well known Fixed Polarity Reed Muller expansion [64][114][107].

Usually, area synthesis of circuits promises more global power savings. However, power dissipation for a circuit depends on input pattern, which requires more specific low power techniques. For low power optimization of Fixed Polarity Reed-Muller (FPRM) functions, to the best of our knowledge, the only results in the literature are algorithms proposed by Zhou and Wong [131] and Narayanan and Liu [74]. The former is for low power XOR gate decomposition, which is the special case of FPRM functions and hence can not solve low power problem for general FPRM functions. And the latter is for the FPRM functions with the implementation of static logic. However, as will be shown later, the algorithm in [74] is not optimal and can only be applied to some special FPRM functions.

The objective of area and power optimization is twofold. This chapter proposes a frame to estimate power dissipation based on FPRM functions, discusses the effect of polarity on area and power dissipation and introduces an optimal algorithm for area and power minimization of single output FPRM functions. The rest of the paper is organized as follows. In Section 8.1, some definitions and terminology are introduced. Some previous work is reviewed in Section 8.2. Section 8.3 presents a frame of power estimation and a cost function to estimate area and power while an optimisation algorithm is proposed in Section 8.4. Finally, some experimental results are shown in Section 8.5 and summary in Section 8.6.

8.1 Definition and terminology

For convenience, some definitions are restated here.

Any n-variable Boolean function can be expressed canonically by the exclusive SOP form as follows.

$$f(x_{n-1}, x_{n-2}, \cdots, x_0) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i \tag{8.1}$$

where the subscript i can be expressed into a binary form as $i = (i_{n-1} i_{n-2} \cdots i_0)$, $b_i \in \{0, 1\}$, the π-terms can be represented as $\pi_i = \dot{x}_{n-1} \dot{x}_{n-2} \cdots \dot{x}_0$, $\dot{x}$ is a literal which can be x or $\bar{x}$,

$$\dot{x}_j = \begin{cases} 1, & i_j = 0 \\ \dot{x}_j, & i_j = 1 \end{cases} \tag{8.2}$$

In equation 8.2, j is from 0 to $n-1$. The form in equation 8.1 is also known as the positive polarity Reed-Muller (PPRM) form if all the variables are in true forms. PPRM forms can be extended to FPRM forms with any fixed polarity p, $p = (p_{n-1} p_{n-2} \cdots p_0)$, where variables can only be either true or complemented, but not both. If a binary bit of p, p_j , is 0 (or 1) then the corresponding variable is in the true (or complemented) form. Hence, there are 2^n polarities for an n-variable function. Each FPRM form can be identified by its polarity.

Example 8.1. A four variable function under polarity 0 is given: $f(x_3, x_2, x_1, x_0) = x_0 \oplus x_1 x_0 \oplus x_2 x_1 \oplus x_2 x_1 x_0 \oplus x_3 \oplus x_3 x_0 \oplus x_3 x_2 \oplus x_3 x_2 x_0 \oplus x_3 x_2 x_1$.

Then, under polarity 5, it has

$$f(x_3, \bar{x}_2, x_1, \bar{x}_0) = 1 \oplus \bar{x}_0 \oplus x_1 \oplus \oplus \bar{x}_2 x_1 \bar{x}_0 \oplus x_3 x_1 \oplus x_3 \bar{x}_2 \bar{x}_0 \oplus x_3 \bar{x}_2 x_1 \tag{8.3}$$

while under polarity 14, it will be

$$f(\bar{x}_3, \bar{x}_2, \bar{x}_1, x_0) = x_0 \oplus \bar{x}_2 \oplus \bar{x}_2 \bar{x}_1 x_0 \oplus \bar{x}_3 \oplus \bar{x}_3 \bar{x}_1 \oplus x_3 \bar{x}_2 \bar{x}_0 \oplus x_3 \bar{x}_2 x_1 \tag{8.4}$$

The above functions can be implemented using AND gates and XOR gates with suitable technology. XOR gate has defined in Definition 7.4 while two input AND gate is based on the following definition of AND operator.

Definition 8.1. For two-input AND operator: $f(x_1, x_0) = x_1 \cdot x_0$, the probability of $f(x_1, x_0)$ is [89]:

$$P_{r,f} = P_r(p_{r,x_1}, p_{r,x_0}) = p_{r,x_1} \cdot p_{r,x_0} \tag{8.5}$$

Here, p_{r,x_i} is the input signal probability, which was defined in Definition 7.3 while $P_{r,f}$ represents the output signal probability of the corresponding gate.

8.2 Related work

The total SA of a specific function form depends on how to combine the input signal probabilities, which refers to low power gate decomposition. The problem for gate decomposition has been extensively studied [107][74][131].

In [107], AND gate decomposition was discussed. For dynamic logic, it is found that a modified Huffman's algorithm gives optimal solutions. The modified version of Huffman's algorithm is called Mini-Huffman algorithm which was described in Algorithm 7.6.

Take a simple example to explain the algorithm. For three inputs (x_2, x_1, x_0) and corresponding signal probabilities are 0.9, 0.45 and 0.1 respectively, Fig. 8.1(a) shows the above algorithm applied to decompose a three input AND gate into a tree of two-input gates in dynamic logic. The switching activity is 0.171.

In [131], the authors analyzed some optimal properties and proposed an algorithm to solve the problem of lower power XOR gate decomposition in dynamic logic in Algorithm 7.1.

Fig. 8.1(b) shows the decomposition based on algorithm 7.1. The switching activity is 1.984. It can be seen that this algorithm only works for XOR gate decomposition.

Based on the analysis of XOR operation and assuming that both the signals and their complements are available, Naraynana and Liu claimed that they use the selection of polarity vectors to solve low power logic synthesis for XOR based circuits in Algorithm 7.2.

However, this algorithm can not handle the reduction of tree size and only works for some special cases of FPRM functions. For example, consider two five variable problems, which share the same input signal probabilities: $\{p_{r,x_4}, p_{r,x_3}, p_{r,x_2}, p_{r,x_1}, p_{r,x_0}\} = \{0.1, 0.3, 0.7, 0.8, 0.9\}$. Two function forms are shown in equations 8.6 and 8.7:

$$f(x_4, x_3, x_2, x_1, \bar{x}_0) = x_1\bar{x}_0 \oplus x_2\bar{x}_0 \oplus x_4 \oplus x_4x_3\bar{x}_0 \tag{8.6}$$

$$f(x_4, x_3, x_2, x_1, \bar{x}_0) = x_1\bar{x}_0 \oplus x_2\bar{x}_0 \oplus x_4x_1 \oplus x_4x_3\bar{x}_0 \tag{8.7}$$

Obviously, algorithm 7.2 works for the function in equation 8.6. If x_1 and x_2 in equation 8.6 are replaced by their complements, then an alternative representation of the function is:

$$f(x_4, x_3, \bar{x}_2, \bar{x}_1, \bar{x}_0) = \bar{x}_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_0 \oplus x_4 \oplus x_4x_3\bar{x}_0 \tag{8.8}$$

which has the same size and the same logic functionality as in equation 8.6. However, if the same algorithm is applied to the function in equation 8.7, the function will be changed into:

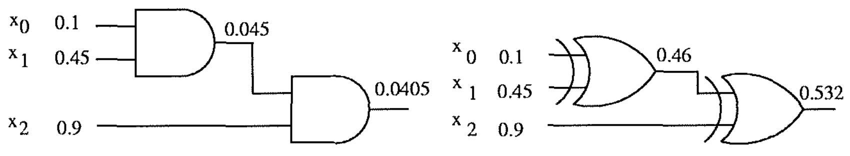


Figure 8.1: (a) Multi input AND gate decomposition in [107]; (b) Multi input XOR gate decomposition in [131]

$$f(x_4, x_3, \bar{x}_2, \bar{x}_1, \bar{x}_0) = \bar{x}_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_0 \oplus x_4\bar{x}_1 \oplus x_4x_3\bar{x}_0 \tag{8.9}$$

which offers different functionality from the function in equation 8.7. This is easily tested by applying the formula $\bar{x} = 1 \oplus x$ to x_2 and x_1 in equation 8.7, which will result in

$$f(x_4, x_3, \bar{x}_2, \bar{x}_1, \bar{x}_0) = \bar{x}_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_0 \oplus x_4 \oplus x_4\bar{x}_1 \oplus x_4x_3\bar{x}_0 \tag{8.10}$$

Obviously, equation 8.10 is different from equation 8.9. Hence, the algorithm does not work well with the function in equation 8.7.

Based on Algorithm 7.2, the algorithm only tried one of 2^n polarities because given input signal probabilities the selected polarity is fixed. For example, for a three input problem shown in Fig. 8.1, the selected polarity is zero while for a five input problem as the above, the selected polarity is 7. Therefore, the selected polarity may not be optimal.

Hence, an FPRM function with multi variables can be implemented using trees of two-input AND gates and a tree of two-input XOR gates. Given the input signal probability set, the switching activity can be computed using the above algorithms.

8.3 Polarity based area and power minimization of FPRM functions

8.3.1 Power estimation

As stated in Section 7.1, all discussions here refer to CMOS technology. Consider dynamic power in CMOS circuit. The power can be modeled as in equation 7.1.

Power dissipation for a circuit is an input pattern involved problem. Any FPRM form can be implemented by AND gates and XOR gates. Before technology mapping, conventionally, all multi input AND gates and XOR gates need to be decomposed into a tree of two input gates, which is called AND & XOR tree. A tree is defined in Definition 7.2.

Furthermore, each primary input signal into a tree is treated as a random variable and its probability is defined as in Definition 7.3.

Power estimation of an AND & XOR tree is similar to that of an XOR tree in Section 7.2. The difference is that there are two kinds of nodes in the AND & XOR tree, namely XOR gate output and AND gate output.

Given an n-input FPRM function with primary input signals $I = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and corresponding signal probabilities $\{p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}\}$, based on the above discussion, the procedure for the power estimation of an FPRM function is as follows.

1. Given an n-variable FPRM function and its primary input signal probability set, construct a tree of two input AND gates and XOR gates;
2. Compute the output probability for each gate using equations 7.2 and 8.5;
3. Compute the node switching activity using equations 7.5 or 7.6;
4. Compute the total power dissipation using equation 7.1.

However, for a given FPRM function with given input signal probability set, the decomposition can have a significant impact on the amount of power dissipation.

8.3.2 Area and power estimation

For an n variable function, there are 2^n polarities and correspondingly there are 2^n FPRM forms, which results in different number of π-terms. If implemented as a tree, then the different number of π-terms will have different tree sizes and result in different power dissipation. Hence, area and power minimization of FPRM functions can be implemented by searching the best polarity among 2^n polarities. This is guaranteed by the following Lemma.

Lemma 8.1. *For any n-variable FPRM function, there are 2^n different forms. Each form can be identified by corresponding polarity. Among 2^n different forms, there exists at least one form, which has the minimal number of $\pi - terms$. For convenience, it is called minimal form.*

An n-variable FPRM function has 2^n different forms. Different forms have different function sizes or different $\pi - terms$. Then, the number of $\pi - terms$ is used to measure the area of an

FPRM form.

For power estimation, it is an input relevant pattern problem. Two input AND gates and XOR gates are used to build a tree to implement an FPRM function. Then, for a given input signal probability set, Mini-Huffman algorithm in Definition 7.6 & Algorithm 7.1 are used to combine the input signal probabilities to calculate the total SA.

Lemma 8.1 also shows that for any n-variable FPRM function, there is one or multiple minimal forms. If the function has one minimal form, the task of area minimization is to find a polarity, which has the minimal form. If the function has multiple minimal forms, then choose any of them because there is no difference among them for area minimization. However, for power dissipation, different minimal forms may have different power dissipation for a specific input signal pattern. Furthermore, the minimal form does not always have the minimal power. The power dissipation under the minimal form may be higher than that under other forms. This can be shown by the following three variable function example.

Example 8.2. A three-variable function, $f(x_2, x_1, x_0)$, with corresponding input signal probabilities $(p_{x_2}, p_{x_1}, p_{x_0})$ = (0.10, 0.90, 0.90) is shown as two-level FPRM format in equation 8.11.

$$f(x_2, x_1, x_0) = x_0 \oplus x_1 \oplus x_1x_0 \oplus x_2x_0 \oplus x_2x_1 \tag{8.11}$$

The function is given under polarity 0. It can be found that the minimal form is under polarity 4 as shown in equation

$$f(\overline{x}_2, x_1, x_0) = x_1x_0 \oplus \overline{x_2}x_0 \oplus \overline{x_2}x_1 \tag{8.12}$$

Also the function form under polarity 5 can be obtained as in equation

$$f(\overline{x}_2, x_1, \overline{x}_0) = x_1 \oplus x_1\overline{x_0} \oplus \overline{x_2} \oplus \overline{x}_2\overline{x_0} \oplus \overline{x_2}x_1 \tag{8.13}$$

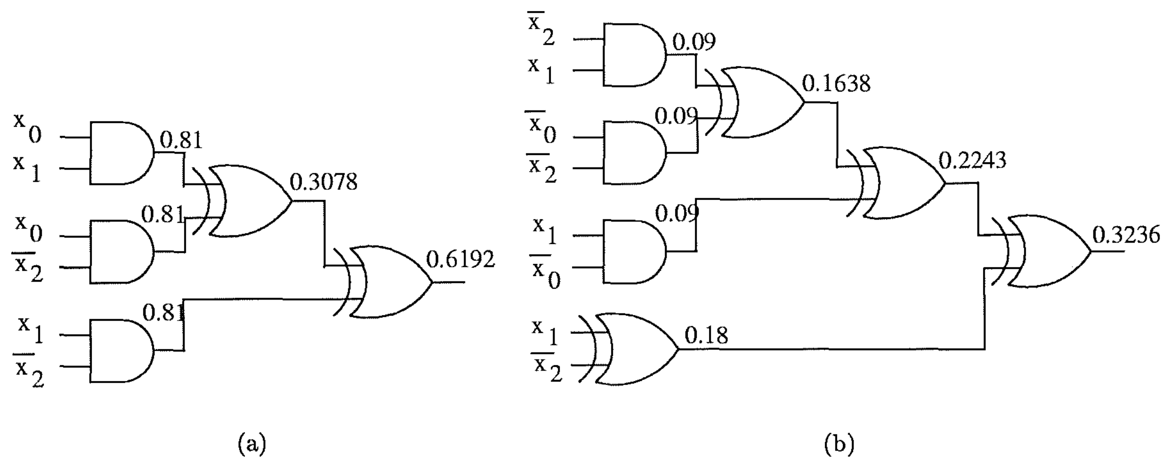


Figure 8.2: Power dissipation with the function forms (a) The minimal form under polarity 4; (b) The function form under polarity 5

Equations 8.12 & 8.13 can be implemented using two-input AND gates and XOR gates as shown in Figure 8.2. The switching activity of Fig. 8.2 (a) is 6.714 while it is 2.323 for Fig. 8.2 (b). For the form under polarity 5, the power reduction is 65.4% compared to that of the minimal form though the penalty of area is two $\pi - terms$. To find a good compromise of area and power dissipation, a cost function is proposed to guide the optimization as in equation 8.14.

$$Cost = \alpha * Area + (1 - \alpha) * SA \tag{8.14}$$

Here, SA stands for the Switching Activity and α is the weight of area and SA and $0 < \alpha < 1$. In order to find the best polarity, which gets area and switching activity minimized, among 2^n polarities, a polarity conversion algorithm is needed.

8.3.3 Polarity conversion

For area minimization, various methods have been developed. They can be classified into two categories [92]:

1. Gray code: Sequentially search 2^n polarities to find the best one which has the minimum number of products. Memory requirement is $O(2^n)$ while the computation time is $O(4^n)$.

2. Extended truth vector: Using an extended truth vector and a weight vector, obtain the number of products for 2^n different expansions simultaneously. Both memory requirement and the computation time are $O(3^n)$.

Any n-variable FPRM function can be uniquely expressed by its on set coefficient. Recently, an exact method was reported in [114] to find the best polarity targeting area minimization based on on-set coefficients by gray code, which is suitable for large functions. Memory requirement and the computation time are $O(M)$ and $O(2^n M)$ respectively, where M is the average number of on-set coefficients. This method is used here to implement function polarity conversion.

For convenience, some definitions are given as follows.

Definition 8.2. Two integers can be expressed by binary n-tuples, $i = \{i_{n-1}i_{n-2}\cdots i_0\}$, $j = \{j_{n-1}j_{n-2}\cdots j_0\}$. If $i_k \geq j_k$ for all k, $0 \leq k \leq n-1$, then i covers j or j is covered by i.

Algorithm 8.1. *Wang's algorithm [116]*

==

Given an on-set Reed-Muller coefficient set R_p for an n-variable Boolean function with polarity p. A coefficient set R_p with any other polarity q can be achieved through the following operations on R_p itself, where $p \wedge q = r$ ($\wedge$ is bitwise operator).

(1) For any coefficient i in the set R_p, if i does not cover r, then i is an element of R_p. Leave i in the set. If i covers r, search the set R_p for the coefficient $(i-r)$. If there is such a coefficient, then delete coefficient $(i-r)$ from the set R_p. Otherwise, if there is not such a coefficient, then add coefficient $(i-r)$ to the set R_p.

(2) The new set obtained in step (1) is the on-set Reed-Muller coefficient set with polarity p.

Using the above polarity conversion algorithm, the optimization implementation of FPRM functions can be described as follows:

Given an n-input PPRM function with primary input signals $I = \{x_{n-1}, x_{n-2}, \cdots, x_0\}$ and corresponding signal probabilities $\{p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots, p_{r,x_0}\}$, two input AND gates and two input XOR gates are used to construct a tree $T = (V, E)$ with $p_{r,x_{n-1}}, p_{r,x_{n-2}}, \cdots$, and p_{r,x_0} as its leaves. Search the best polarity such that both area and equations 8.15 or 8.16 are minimized for dynamic logic and static logic respectively.

For dynamic logic,

$$SA = \sum_{i \in V} 2P_{r,i} \tag{8.15}$$

For static logic,

$$SA = \sum_{i \in V} 2P_{r,i}(1 - P_{r,i}) \tag{8.16}$$

Where SA is the total switching activity and $P_{r,i}$ is the signal probability of an internal node.

8.4 Proposed algorithms

Based on the above discussion, a new algorithms for minimizing area and power dissipation of single output FPRM functions is proposed as follows.

Algorithm 8.2. *Power minimized algorithm*

==

Given an on-set Reed-Muller coefficient set R_p of a single output FPRM function with the input signal probability set under polarity p, let one real variable and one integer variable, SACost, AreaCost, represent the total switching activity and area cost (measured by the number of π-terms) under polarity p, and let one real variable and two integer variables, BestSACost, BestAreaCost and BestPolarity, represent the lowest total switching activity, area cost and the polarity corresponding to the lowest total switching activity, respectively. Then, for any i, $0 < i \leq 2^n - 1$, carry out the following steps:

(1)Generate a polarity q_i in gray code order, that is adjacent with q_{i-1}. Let $r = q_i \wedge q_{i-1}$.

(2) Pass $R_{q_{i-1}}$ and r to Algorithm 8.1 to get the new on-set coefficient set R_{q_i}. Obtain AreaCost from the set R_{q_i}.

(3) If $BestAreaCost < AreaCost$, let $q_i = q_{i-1}$ and $q_i = q_i + 1$, go to Step 1.

(4) If BestAreaCost $\geq$ AreaCost, construct an optimal tree of two input AND gates using Algorithm 7.6 for each π-term in the FPRM function. Compute the switching activity for all AND-trees and store it into SACost and save the primary output signal probabilities of AND-trees into an array called ANDOutProb[]. Taking the signal probabilities in ANDOutProb[] as the primary input signal probabilities, construct an optimal tree of two input XOR gates using Algorithm 7.1. Compute the switching activity of the XOR-tree and add it to SACost. If SACost is less than BestSACost, then BestSACost, BestAreaCost and BestPolarity are substituted by SACost, AreaCost and q_i respectively. Otherwise, go to Step 1.

(5) Output (BestPolarity $\wedge$ p) that is the best polarity with BestSACost and BestAreaCost.

The above algorithm supposes that the area has the higher weight than the power dissipation. If the function has many area minimal representations, the algorithm can find the best polarity, which has the lowest power dissipation.

Example 8.3. A three-variable function, $f(x_2, x_1, x_0)$, with corresponding input signal probabilities (0.10, 0.45, 0.90) is shown as two-level FPRM format in equation 8.17.

$$f(x_2, x_1, x_0) = x_0 \oplus x_1 x_0 \oplus x_2 x_1 x_0 \tag{8.17}$$

Obviously, the function representation is under polarity 0. The on-set Reed-Muller coefficient set R_0 is (1, 3, 7). First, construct an optimal tree of two input AND gates using Algorithm 7.6 for each π-term in the FPRM function and compute the switching activity for all AND-trees and store it into SACost and save the primary out signal probabilities of AND-trees into an array called ANDOutProb[]. For this example, $\pi_1 = x_0$, $\pi_3 = x_1 x_0$, $\pi_7 = x_2 x_1 x_0$ and corresponding AND trees are shown in Fig. 8.3 (a). The switching activity is SACost = 0.9810. The primary out signal probabilities of AND-trees are ANDOutProb[0] = 0.9, ANDOutProb[1] = 0.405 and ANDOutProb[2] = 0.0405.

Then, take the signal probabilities in ANDOutProb[] as the primary input signal probabilities, construct an optimal tree of two input XOR gates using Algorithm 7.1 and compute the switching activity of the XOR-tree and add it to SACost. The XOR tree is shown in Fig. 8.3 (b) and add its switching activity to SACost. Obtain SACost = 2.9460. Let BestSACost be 2.9460, BestAreaCost be 3 and polarity 0 be BestPolarity.

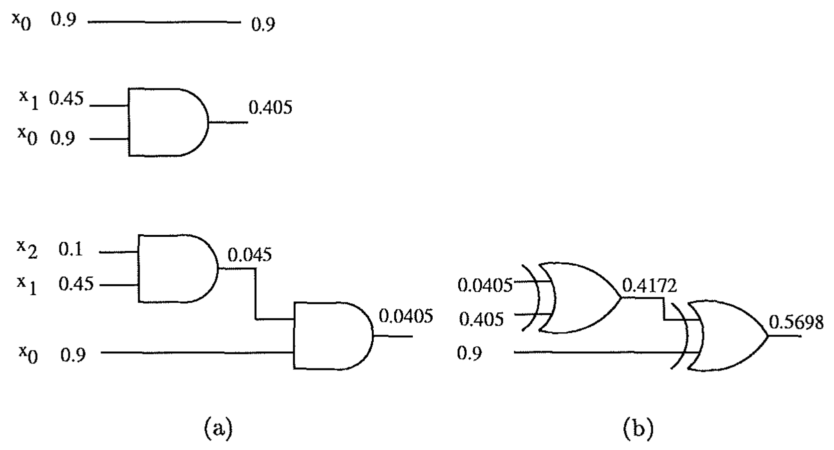


Figure 8.3: Decomposition and AND- trees and XOR tree (a) AND trees; (b) XOR tree

Generate polarity 1 in gray code order, 001. Let $r = 001 \wedge 000 = 1$. Pass $R_0 = (1, 3, 7)$ and $r = 1$ to Algorithm 8.1 to get the new on-set coefficient set R_1 = (0, 1, 2, 3, 6, 7). Because AreaCost is 6, which is greater than BestAreaCost, go to Step 1 and generate gray code for next polarity 2. Repeat this procedure till polarity 7. Finally, find that the BestPolarity is polarity 4, BestSACost is 0.2362 and BestAreaCost is 2.

Algorithm 8.3. *Power and area optimisation algorithm*

==

Given an on-set Reed-Muller coefficient set R_p of a single output FPRM function with the input signal probability set under polarity p, let two real variables and one integer variable, Cost, SACost, AreaCost, represent the cost, total switching activity and area cost (measured by the number of π-terms) under polarity p, and let two real variables and two integer variables, BestCost, BestSACost, BestAreaCost and BestPolarity, represent the best cost, the lowest total switching activity, area cost and the polarity corresponding to the lowest total switching activity, respectively. Then, for any i, $0 < i \leq 2^n - 1$, carry out the following steps:

(1)Generate a polarity q_i in gray code order, that is adjacent with q_{i-1}. Let $r = q_i \wedge q_{i-1}$.

(2) Pass $R_{q_{i-1}}$ and r to Algorithm 8.1 to get the new on-set coefficient set R_{q_i}. Obtain AreaCost from the set R_{q_i}.

(3) Construct an optimal tree of two input AND gates using Algorithm 7.6 for each π-term in the FPRM function. Compute the switching activity for all AND-trees and store it into SACost and save the primary output signal probabilities of AND-trees into an array called ANDOutProb[]. Taking the signal probabilities in ANDOutProb[] as the primary input signal probabilities, construct an optimal tree of two input XOR gates using Algorithm 7.1.

(4) Compute the switching activity of the XOR-tree and add it to SACost and take the number of π-terms as area. Use equation 8.14 to calculate the cost. If Cost is less than BestCost, then BestCost, BestAreaCost and BestPolarity are substituted by Cost, AreaCost and q_i respectively. Otherwise, go to Step 1.

(5) Output (BestPolarity $\wedge$ p) that is the best polarity with BestSACost and BestAreaCost.

8.5 Experimental results

The proposed algorithm has been implemented in C and compiled by the GNU C compiler. The time complexity for mini-Huffman is $O(nlogn)$ [131] while that for polarity conversion is $O(M2^n)$ [114]. Hence, the proposed algorithm has the time complexity of $O(M2^nnlogn)$ where n is the number of variables and M is the average number of on-set coefficients. The algorithm is tested on a personal computer with PIII 550 CPU and 64M RAM under Linux operating system. Ten single output circuits with input size from 8 to 21 from MCNC benchmarks are tested.

The algorithm proposed in [115] is used to convert the test circuits directly from two-level PLA format to two-level FPRM format with polarity 0. Then, the proposed algorithm will work with the two-level FPRM format circuits. Given a set of input signal probabilities, the best polarity is searched and its corresponding switching activity and area are found, where area is measured by the number of π-terms. Twenty-one input signal probabilities are randomly generated, which are 0.95, 0.08, 0.78, 0.25, 0.18, 0.77, 0.75, 0.71, 0.47, 0.07, 0.23, 0.73, 0.28, 0.73, 0.28, 0.73, 0.44, 0.81, 0.20, 0.38, 0.71, respectively. For different input sizes, the input signal probabilities are chosen from the left to the right in the above list. For example, for three inputs (x_2, x_1, x_0), the signal probabilities are 0.78, 0.08, 0.95 while for four inputs (x_3, x_2, x_1, x_0), they are 0.25, 0.78, 0.08, 0.95, and so on.

Two sets of experiments are conducted. The first set is based on Algorithm 8.2. Ten circuits from MCNC benchmark are tested. Table 8.1 shows the experimental results [127]. To see the efficiency of the algorithm, the switching activity and area under the polarity 0 are also listed in the same table for comparison. In Table 8.1, Column 1 shows the circuit name while Column 2 shows the number of inputs (IN#) and the number of product terms (p#) in PLA format; Columns 3 and 4 show the switching activity (SA_0) and area ($Area_0$ the number of π- terms in two-level RM format) under polarity 0 while Columns 5, 6, 7 and 8 show the best polarity, switching activity (SA_b), area ($Area_b$) and CPU time for the best polarity using the proposal algorithm, respectively; Columns 9 and 10 show the improvement percentage of the switching activity and area under the best polarity compared to those under the best polarity given in [115]. The improvement percentage of switching activities is defined as follows:

$$Improvement = \frac{SA_0 - SA_b}{SA_0}\% \tag{8.18}$$

while the one for area is similarly defined in equation 8.19.

$$Improvement = \frac{Area_o - Area_b}{Area_0}\% \tag{8.19}$$

From the results reported in Table 8.1, it is found that the switching activities can be improved significantly for all ten circuits compared to those under polarity 0, which can be as high as 91.9%. The average reduction of switching activities is 61.0% while the average area reduction is 32.0% for all ten circuits. For seven cases of ten circuits, both area and switching activity are reduced.

For each circuit, 10 sets of input probabilities are randomly generated and then run on the above algorithm. For each instance X, let $Opt(X)$ represents the optimal solution under the best polarity and $S(X)$ be the solution with polarity 0. The following parameter defined in equation

8.20 is used to measure the performance of the algorithm.

$$R = \frac{S(X) - Opt(X)}{S(X)}\% \tag{8.20}$$

From the definition in 8.20, obviously, the higher the rate R, the worse the performance of the algorithm. For each circuit, among 10 problems, the maximum ratios and the average improvement ratios of switching activity and area are computed, which are represented by $MaxSAR$, $MaxAreaR$, $AvergAreaR$ and $AvergSAR$, respectively. The results are shown in Table 8.2.

The second experiment is conducted based on Algorithm 8.3, which uses the cost function in equation 8.14 to guide the optimization. To determine the weight α in equation 8.14, an experiment is conducted as follows. An input signal probability set is randomly generated, in

Table 8.1: Best polarity and switching activity for low power decomposition[127]

Name	IN#/p#	Polarity 0		Best Polarity				Improvement (%)	
		SA_0	$Area_0$	polarity	SA_b	$Area_b$	time(s)	SA	Area
Newwill	8/ 8	16.84	57	150	2.10	14	0.21	87.5	75.4
Newtag	8 /8	8.20	21	160	2.18	6	0.14	73.4	71.4
sym10	10/837	128.76	266	966	20.54	557	29.42	84.0	-109.4
xor5	5/5	3.14	5	6	2.52	5	~0	19.7	0
9sym	9/87	136.36	210	450	11.58	173	1.94	91.5	17.6
life	9/512	113.26	184	255	23.12	100	3.82	79.6	45.6
t481	16/481	46.96	41	39323	9.82	19	29.46	76.7	53.6
ryy6	16/122	23.38	80	49152	13.74	64	16.70	41.2	20.0
cm150a	21/17	139.30	163	819200	17.22	81	41.30	87.6	50.3
mux	21/36	77.40	81	786432	40.06	81	40.14	48.1	0
Average improvement (%)								73.2	22.0

Table 8.2: Reduction ratios of area and switching activity

Name	MaxSWR(%)	MaxAreaR(%)	AvergSWR(%)	AvergAreaR(%)
Newill	89.7	75.4	79.2	74.0
Newtag	88.0	93.2	68.0	72.2
Sym10	93.6	-112.0	74.2	-98.4
xor5	19.7	0	8.6	0
9sym	93.2	17.6	83.2	17.6
life	95.4	45.7	79.7	34.8
t481	89.1	68.3	78.9	61.0

which the elements are 0.15, 0.97, 0.48, 0.10, 0.24, 0.16, 0.60, 0.96, 0.78, 0.56, 0.26, 0.74, 0.53, 0.87, 0.37, 0.39 respectively for a sixteen input variable set. Power versus alpha and area versus alpha are conducted. The results are shown in Figs. 8.4 (a) and (b). Fig. 8.4 (a) depicts the normalized power of four benchmarks versus alpha, i.e. the power when $\alpha = 1$ is divided by the power with specific α. Fig. 8.4 (b) depicts the normalized area versus alpha, i.e. the area when $\alpha = 0$ is divided by the area with specific α. For circuits life & 9sym, power and area do not vary with α while for circuits t481 & sym10, power and area varies significant with α. From Figs. 8.4 (a) & (b), it can be seen that setting $\alpha = 0.4$ produces the optimization implementations for area and power.

Table 8.3 shows the results of ten MCNC benchmark circuits. The results of the area minimized algorithm in [116] are also listed in the table for comparison. It can be seen that two circuits have significant power improvement without any area penalty while one circuit has power improvement with small area trade. For the circuits 9sym and cm150a, the proposed algorithm finds the best polarities, which have the minimal area and power dissipation. However, for the circuit ryy6, it means that the polarity which has the minimal area is not the best polarity for power minimization.

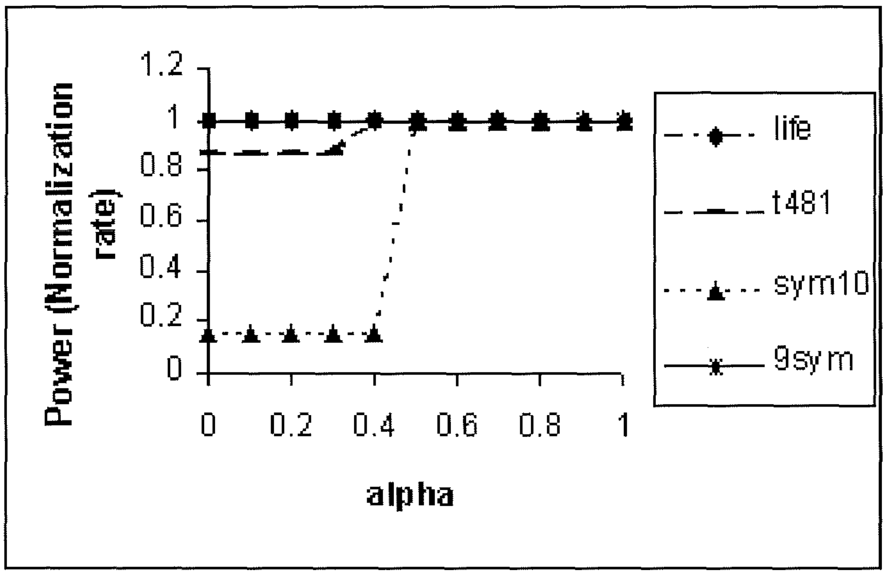
Table 8.3: Power dissipation and area comparison

Name	Inputs	Results in [116]			Proposed in [128]				Reduction	
		P	area	SA	P	area	SA	time (s)	Area (%)	SA (%)
9sym	9	15	173	154.92	195	173	31.1	0.21	0	79.9
ryy6	16	49152	64	38.76	57344	72	19.9	16.83	-12.5	48.7
mux	21	0	81	19.86	0	81	19.86	40.14	0	0
cm150a	21	1	82	47.90	32768	82	25.28	40.35	0	47.2

Table 8.4: Reduction ratios of area and SA

Name	MaxSAR (%)	MaxAreaR (%)	AverageSAR (%)	AverageAreaR (%)
9sym	79.7	0	60.0	0
ryy6	48.7	-12.5	11.7	-3.1
cm150a	78.0	-1.2	29.9	-0.1

Once again, for each circuit, 10 sets of input probabilities are randomly generated and then run on the proposed algorithm and area efficient algorithm in [116] to see the efficiency of the cost function in equation 8.14. Equation 8.20 is used to measure the performance of the proposed algorithm. The results are shown in Table 8.4. It shows that the proposed algorithm has a significant power dissipation improvement with smaller area penalty compared with the results from the area minimized algorithm.



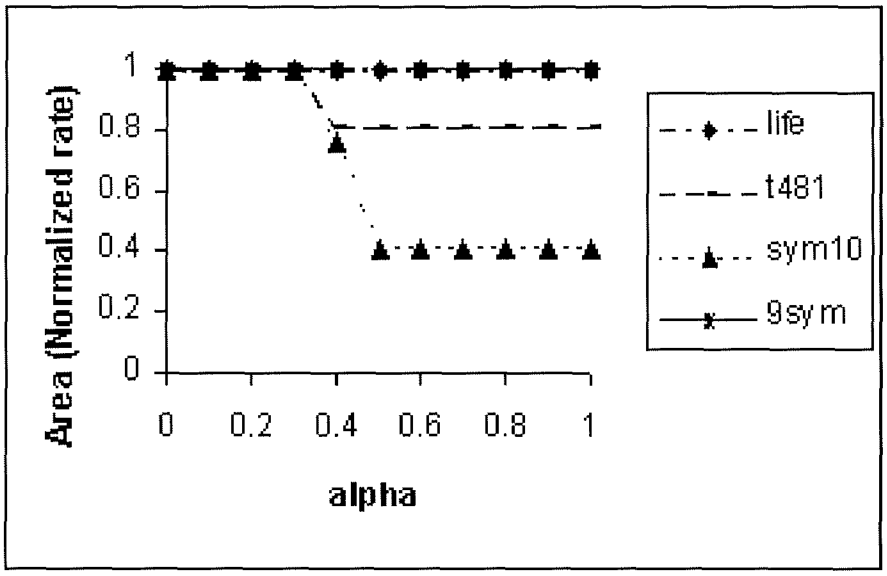


Figure 8.4: Power and area versus alpha (a) Power versus alpha; (b) Area versus alpha

8.6 Summary

A frame to estimate power dissipation of FPRM functions has been presented. Using the polarity conversion under the assumption that both signals and their complements are available, a solution to minimize power dissipation for single output FPRM functions is proposed. The time complexity of the proposed algorithm is $O(M2^{n}nlogn)$. Although the dynamic logic is taken to explain how the algorithm works, it should be pointed out that the algorithm can be applied to static logic as long as the calculation formula of switching activity in equation 7.5 is substituted by the one for static logic in equation 7.6.

Chapter 9

Conclusions and Future work

This project aims to develop low power design techniques for digital logic circuits. The main contributions in this thesis can be summarized as follows.

9.1 Low power state assignment

Using the state assignment to reduce area or power of FSMs is not new. This can be classified as a single object optimization problem. However, using state assignment to optimize power and area is relatively new, which is a multiple object optimization problem and much more difficult than single object optimization. For the optimization problem, the optimization quality of a problem mainly depends on the algorithm used and the cost function developed. An algorithm which has large search space is essential to solve large size problems. Considering this, a genetic algorithm is employed to do this job in Chapter 3. Corresponding operators are developed for this specific application. Switching activity is used to estimate power dissipation of an FSM, which includes the power dissipation both from the register section and combinational section. Based on the number of cubes and switching activity, two cost functions are developed, which can optimize FSMs targeting area or power dissipation or both. For a specific state assignment, the area of the combinational section is calculated by ESPRESSO minimization. The algorithm is implemented in C and the efficiency is tested on MCNC benchmarks. The results show that the proposed algorithm has significant improvement in area and power dissipation compared to those from the state-of-the-art tools.

9.2 Low power flip-flop designs

Pulse sampling flip-flops usually suffer from a long clock chain, which is used to generate narrow pulse, and the long clock chain consumes significant power dissipation compared to the flip-flop cell. In Chapter 4, a clock-gating technique is used to reduce the redundant transition in a clock chain. A novel low power Single Edge-Triggered Flip-Flop with Clock-Gating (CG-SETFF) is proposed. The proposed flip-flop can detect the idle of the input signal to gate the clock chain so that the clock power can be saved. From the simulation result, the power savings can be as high as 86% compared with the published design when the input is in idle conditions.

Clock signal is the most frequent transition signal in a flip-flop except glitches. Traditional flip-flops are single edge-triggered flip-flops, which are sensitive to the rising or falling edge of the clock. Therefore, half of the clock's transitions have nothing to do with the circuit and became redundant. One solution to reduce the redundant transition is to develop double edge-triggered flip-flops. In Chapter 5, two versions of differential CMOS double edge-triggered flip-flops are proposed: Clock Chain Based Double Edge-Triggered Flip-Flop (CCB-DETFF) and Pass Transistor Based Differential CMOS Double Edge-Triggered Flip-Flop (PTB-DETFF). CCB-DETFF has the lower power dissipation than CG-SETFF when input signal probability is greater than 0.38. PTB-DETFF only uses one clocked nMOS transistor. Therefore, it has some advantages to use in the reduced clock swing and voltage scalability.

Reducing the number of signal lines can reduce the total capacitor and hence power dissipation can be reduced. One solution to do that is to use multiple valued signals. In Chapter 6, multiple valued flip-flop is explored and a novel quaternary flip-flop is proposed. The simulated result shows that the proposed flip-flop has a correct function and has lower power dissipation compared to other multiple flip-flop

9.3 Power optimization of FPRM functions

Low power techniques of AND/OR operator based domain have been extensively investigated. However, for AND/XOR operator based domain, the research is still in its infancy.

Before technology mapping, a multiple input XOR gate needs to be decomposed into a tree of two input gates. In Chapter 7, low power XOR gate decomposition is discussed. It is shown that previous algorithms for low power XOR gate are not optimal. Based on polarity searching and some given properties, a novel algorithm is presented and implemented in C. The results show that improved power dissipation is obtained.

In Chapter 8, the above idea is extended to Fixed Polarity Reed-Muller (FPRM) functions. A power estimation frame for FPRM functions is proposed. A cost function, which is linearly combined by the number of π-terms and power dissipation, is proposed to optimize area and power dissipation. Based on polarity conversion, two algorithms are presented. One is to optimize power dissipation and the other is to optimize area and power. The experimental results for MCNC benchmark circuits show that an improvement is obtained compared to the unoptimized circuits or published results.

9.4 Future work

The above work can further be generalized and improved along the following lines.

- Low power state assignment program described in Chapter 3 can be incorporated into other logic minimizers, such as SIS, to improve their performance. The state assignment technique can be extended to solve the multilevel FSMs in which their combinational sections are multilevel. It is also possible to apply this technique to low power FSM decomposition[29].
- The idea of clock-gating technique proposed in Chapter 4 can be applied to low power sequential circuit design[120] and low power finite state machine partitioning[18].
- Multiple-valued clock based flip-flops can be further explored to design flip-flops with simpler structure. Moreover, the design methodology of low power sequential systems based

on this kind of flip-flops needs to be studied.

- Low power XOR gate decomposition algorithm in Chapter 7 can be modified to take the delay into account so that the target circuit is optimized in both power dissipation and delay. If a proper delay model is applied, gliching power can be taken into account. Then, using this logic synthesis technique, the "real" power savings will be more accurate.
- Low power FPRM function algorithm in Chapter 8 is for single output functions, which can be extended to solve multiple output FPRM functions. Further, if some algebraic abstraction and reduction rules are applied, then the above algorithm will be more efficient to optimize area and power.

Publications

The following are papers published, accepted and submitted while at Napier University

- **Y Xia** and A E A Almaini, Differential CMOS edge-triggered flip-flop with clock-gating, Electronics Letters, Vol. 38, No. 1, pp.9-11, 2002
- **Y Xia** and A E A Almaini, Genetic algorithm based state assignment for power and area optimization, IEE Proceedings - Computers and Digital Techniques, Vol. 149, No.4, pp. 128-134, 2002
- **Y Xia** and A E A Almaini, Best polarity for low power XOR gate decomposition, Euromicro Symposium on DSD'2002 Digital System Design, Proceedings of DSD'2002, pp. 53-59 , Germany, Sept., 2002
- P. Wang, X. Wu and **Y Xia**, Design of ternary Schmitt triggers based on its sequential characteristic, IEEE International Symposium on Multiple-valued logic, IEEE Proc. of ISMVL, pp. 156-160, Boston, USA, May 2002
- **Y Xia**, A E A Almaini and Xunwei Wu, Genetic algorithm based finite state machine optimization, Journal of Electronics (Accepted)
- **Y Xia** and A E A Almaini, Power minimisation of FPRM functions based on polarity conversion, Journal of Computer Science Technology (Accepted)
- **Y Xia** and A E A Almaini, Area and power optimisation of FPRM function based circuits, IEEE International Symposium on Circuits and Systems, Bangkok, May, 2003 (accepted)
- **Y Xia** and A E A Almaini, A novel multiple valued flip-flop employing multiple valued clock, (submitted to International Journal of Electronics)

References and Bibliography

[1] Afghahi M. and Yuan J., Double edge-triggered D-flip-flops for high-speed CMOS circuits, IEEE Journal of Solid-state Circutis, Vol. 26, No.8, pp.1168-1170, 1991

[2] Alidian M., Monteiro J., Devadas S., Ghosh A., and Papaefthymiou M., Precomputation-based sequential logic optimization for low power, IEEE Trans. on VLSI Systems, Vol. 2, No. 4, pp. 426-436, 1994

[3] Almaini A. E. A. Electronic logic systems, Third Edition, Prentice Hall, UK, 1994.

[4] Almaini A. E. A. and McKenzie L., Tabular techniques for generating Kronecker expansion, IEE Proc. -Comput. Digit. Tech., Vol. 143, No. 4, pp. 205-212, 1996

[5] Almaini A., Miller J., Thomson P. and Billina S., State assignment of finite state machines using a genetic algorithm, IEE Proc.-Comput. Digit. Tech., Vol. 142, No. 4, pp. 279-286, 1995

[6] Almaini A. E. A. and Ping S., Algorithm for Reed-Muller expansions of Boolean functions and optimization of fixed polarities, The fourth IEEE International Conference on Electronics, Circuits and Systems, Cairo, pp 148-153, Dec., 1997

[7] Almaini A. E. A., Thomson P. and Hanson D., Tabular techniques for Reed-Muller logic, Int. J. Electronics, Vol.70, No. 1, pp. 23-24, 1991

[8] Almaini A. E. A. and Zhuang N., Using genetic algorithms for the variable ordering of Reed-Muller binary decision diagrams, Microelectronic Journal, Vol. 24, No. 4, pp. 471-480, 1995

[9] Almaini A. E. A., Zhuang N. and Bourset F., Minimisation of multiouput Binary Decision Diagrams using hybrid Genetic Algorithms, IEE Electronic Letters, Vol. 31, No. 20, pp. 1722-1723, 1995

[10] Ashar P., Devada S., and Newton A., Sequential logic synthesis, Kluwer, Boston, 1991

[11] Athas W., Svensson L., Koller, Tzartzanis E. and Chou E., A framework for practical low-power digital CMOS systems using adiabatic-switching principles, International Workshop on Low Power Design, pp. 189-194, April, 1994

[12] Bacchetta P., Daldoss L., Sciuto D. and Silvano C., Low-power state assignment techniques for finite state machines, IEEE International Symposium on Circuits and Systems, Gnenva, pp. 28-31, 2000

[13] Bahar R., Frohm E., Gaona C., Hachtel G., Macii E., Pardo A. and Somenzi F., Algebraic decision diagrams and their application, Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Santa Clara, USA, pp. 188-191,1993

[14] Benini L. and Micheli G., Dynamic power management design techniques and CAD tools, Kluwer Academic Publishers, USA, 1998

[15] Benini L. and Micheli G., State assignment for low-power dissipation, IEEE Journal of solid State Circuits, Vol.30, No.3, pp. 258-268, 1995

[16] Benini L. and Micheli G., Automatic synthesis of low-power gated-clock finite-state machines, IEEE Trans. on CAD Design of Integrated Circuits and Systems, Vol. 15, No.6, pp. 630-643, 1996

[17] Benini L. and Micheli G., Synthesis of low-power selectively-clocked systems from high-level specification, AMC Trans. on Design Automation of Electronic Systems, Vol. 5, No.3, 311-321, 2000

[18] Benini L., Micheli G. and Vermeulen F., Finite-state machine partitioning for low power, Proceedings of IEEE International Symposium on Circuits and Systems, Monterey, CA, pp. 5-8, May, 1998

[19] Blair G., Low-power double-edge triggered flip-flop, IEE Electronics Letters, Vol. 33, No. 10, pp. 845-847, 1997

[20] Brace, K., Rudell R., Bryant R., Efficient implementation of a BDD package, Proceedings of the Design Automation Conference, Orlando, USA, pp. 40-45, June, 1990

[21] Brown S., Francis R. and Vranesic Z. G. File-programmable gate array, Kluwer Academic Publisher, Boston, 1992.

[22] Bryaton R., Graph-based algorithms for Boolean function manipulation, IEEE Trans on Comput., Vol. C-35, No.8, pp.79-85, 1986

[23] Brayton R. K., Hachtel, G. D., McMullen C. T. and Sangiovanni-Vincentelli A. L., Logic minimization algorithms for VLSI synthesis, Kluwer Academic Publishers, Boston, 1984

[24] Chandrakasan A. and Brodensen R., Low power digital CMOS design, Kluwer Academic Publishers, Boston, 1995

[25] Chandrakasan A., Sheng S. and Brodersen R., Low power techniques for portable real-time DSP applications, Proceedings of the 5th International Conference on VLSI Design, Bangalore, India, pp. 203-208, Jan. 1992

[26] Chandrakasan A., Sheng S., and Brodersen R., Low-power CMOS digital design, IEEE Journal of Solid-State Circuits, Vol. 27, No.4, pp. 473-484, 1994

[27] Chandrakasan A., Potkonjak M., et al., Optimizing power using transformations, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, No. 1, pp. 12-31, 1995

[28] Chang J. and Pedram M., Energy minimization using multiple supply voltages, Proceedings of International Symposium on Low Power Electronics and Design, Monterey, Calfornia, pp. 157-162, USA, 1996

[29] Chow S., Ho Y. and Hwang T., Low power realization of finite state machines- A decomposition approach, ACM Trans on Design Automation of Electronic Systems, Vol.1, No. 3, 315-340, 1996

[30] Coudert O., Berthet C., Madre J., Verification of sequential machines using Boolean functional vectors, Proceedings of IFIP Intl. Workshop on Applied Formal Methods for Correct VLSI Design, Leuven, Belgium, pp.111-128, Nov. 1989

[31] Current K., Multiple-valued logic memory circuit, International Journal of Electronics, Vol. 78, pp. 547-555, 1995

[32] Dasgupta A. and Ganguly S., Divide & Conquer: A strategy for synthesis of low power finite state machines, Proceedings of IEEE International Conference on Computer Design, Texas, USA, pp740-745, Oct. 1997

[33] Devadas S., Ma H., Newton A. and Sangiovanni-Vinventelli A., MUSTANG: State assignment of finite state machines targeting multilevel logic implementations, IEEE Trans., Comput.-Aided Des, Integr. Circuits Syst., Vol. 7, No.12, pp. 1290-1300,1998

[34] Dickinson A., Denker J., Adiabatic dynamic logic, IEEE Journal of Solid-State Circuits, Vol. 30, No. 3, pp. 311-315, 1995

[35] Drechsler R., Theobald M. and Becker B., Fast OFDD-based minimization of fixed polarity Reed-Muller expressions, IEEE Trans. Computers, Vol. 45, No. 11, pp 1294-1299, 1996

[36] Du X., Hatchel G., Lin B. and Newton A., MUSE: A MUltilevel Symbolic Encoding Algorithm for State Assignment, IEEE Transactions on Computer Aided Design, Vol. 10, pp. 28-38, 1991

[37] Ellis S., Power management in notebook counters, Proceedings of the Personal Computer Design Conference, PP. 749-754, 1991

[38] Forrest J., ODE: Output direct state machine encoding, Proceedings of European Design Automation Conference with EURO-VHDL'95 on EURO-DAC, Brighton, England, pp.600-605, Dec. 1995

[39] Forth R., Molitor P., An efficient heuristic for state encoding minimizing the BDD representations of the transition relations of finite sate machines, Proceedings of the IEEE/ACM

Asia and South Pacific Design Automation Conference, Yokohama, Japan, pp. 61-66, Jan. 2000

[40] Gago A. Escano R. and Hidalgo J., Reduce implementation of D-type DET flip-flops, IEEE Journal of Solid-State Circuits, Vol.28, No. 3, pp.400-402, 1993

[41] Geiger M. and Muller-Wipefurth T., FSM decomposition revisited: algebraic structure theory applied to MCNC benchmark FSMs, Proceedings of the Design Automation Conference, pp. 182-185, 1992

[42] Ghosh A., Devadas S., Keutzer K. and White J., Estimation of average switching activity in combinational and sequential circuits, ACM/IEEE Design Automation Conference, California, USA, pp. 253-259, June 1992

[43] Hachtel G., Hermida M., Pardo A., Poncino M., and Somenzi F., Re-encoding sequential circuits to reduce power dissipation, Proceedings of the Intl Conference on Computer-Aided Design, San Jose, USA, pp. 70-73, Nov. 1994

[44] Hachtel G., Macii E., et al., Symbolic algorithms to calculate steady-state probabilities of a finite state machine, Proceedings of IEEE European Design and Test Conference, Paris, pp. 214-218, Feb. 1994

[45] Harris E., Depp S., Pence W., Kirkpatrick S., Sri-jayantha M and Troutman R., Technology directions for portable computers, Proceedings of the IEEE, Vol. 83, No.4, pp. 636-658, 1995

[46] Hartmanis J., and Stearns H., Algebraic structure theory of sequential machines, Prentice-Hall, New Jersey, 1966

[47] Holland J., Adaptation in natural and artificial system, University of Michigan Press, Ann Arbor, MI, 1975

[48] Hong S., Pard S., Hwang S. and Kyung C., State assignment in finite state machines for minimal switching power consumption, IEE Electronics Letters, Vol. 30, No. 8, pp. 627-629, 1994

[49] Hossain R., Wronaski L. and Albicki A., Low power design using double edge triggered flip-flops, IEEE Trans. VLSI Syst., Vol. 2, No.2, pp. 261-265, 1994

[50] Hopcroft J., Ullman J., Introduction to automata theory, languages, and computation. Addison-Wesley, New York, 1979

[51] http://www.intle.com/kits/quickrefyr.htm

[52] Huang M., Kwok R., Chan S., Simplified and accurate power -analysis method for deep-submicron ASIC designs, IEE Proc. -Circuits Devices Syst., Vol. 147, No. 3, pp. 175-182, 2000

[53] Johnson M. and Roy K., Datapath scheduling with multiple supply voltages and level converters, ACM Trans. on Design Automation Electronic Systems, Vol. 2, No. 3, pp. 227-248, 1997

[54] Kawaguchi H., Skurai T., A reduced clock-swing flip-flop (RCSFF) for 63% power reduction, Journal of Solid-State Circuits, Vol. 33, No. 5, pp. 807-811, 1998

[55] Keonjian E., Micropower Electronics, Pergamon Press, London, New York, 1964

[56] Liew B., et al., Electromigration interconnect lifetime under AC and pulse DC stress, International Reliability Physics Symp., pp. 215-219, 1989

[57] Lin B. and Newton A., Synthesis of multiple level logic from symbolic high-level description languages, Proceedings of International Conference on Very Large Scale Integration, Federal Republic of Germany, pp. 187-196, Aug., 1989

[58] Lu S. and Ercegovac M., A Novel CMOS implementation of double-edge-triggered flip-flops, IEEE Journal of Solid-state Circuits, Vol. 25, No.4, pp. 1008-1010, 1990

[59] Kim J., Jang Y. and Park H., CMOS sense amplifier-based flip-flop with two N-C^2MOS output latches, IEE Electronics Letters, Vol. 36, No. 6, pp. 498-499, 2000

[60] Macii E. and Pncino M., Design techniques and tools for low-power digital systems, Euro-Training Course, Italy, Nov. 2000

[61] Mangione-Simith B., Low power communication protocols: Paging and beyond, Symposium on Low Power Electronics, San Jose, USA, pp.8-11, Nov. 1995

[62] Marculescu D., Marculescr R. and Pedram M., Information theoretic measures for power analysis, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.15, No.6, pp. 599-598,1996

[63] Marculescu D., Marculescr R. and Pedram M., Theoretical bounds for switching activity analysis in finite-state machines, IEEE Trans. on VLSI systems, Vol.8, No. 3, pp. 335-339, 2000

[64] Mckenzie L., Logic synthesis and optimization using Reed-Muller expansions, PhD thesis, Napier University, UK, 1995

[65] Meng T., Gordon B., et al., Portable video-on-demand in wireless communication, Proceedings of the IEEE, Vol. 83, No. 4, pp. 659-680, 1995

[66] Meindl J., Micropower Circuits, J. Wiley and Sons, New York, 1969

[67] Micheli G., Brayton R. and Sangiovanni-Vincentelli A., Optimal state assignment for finite state machines, IEEE Trans. on CAD, Vol. CAD-4, No.3, pp.269-283, 1985

[68] Mishra S., Rofail M. and Yeo, K., Design of high performance double edge-triggered flip-flops, IEE Proc. -Circuits Devices Syst., Vol. 147, No. 5, pp. 283-2, 2000

[69] Molhi S. and Chatterjee P., I-V microsystem-scaling on schedule for personal communications, IEEE Circuits and Devices, pp. 13-17, 1994

[70] Monteiro J. and Devadas S., Retiming sequential circuits for low power, Proceedings of the Int'l Conference on Computer-Aided Design, California, USA, pp. 398-402, 1993

[71] Monteiro J. and Oliveira A., Finite state machine decomposition for low power, Proceedings of the 35th Annual Conference on Design Automation Conference, San Francisco, USA, pp. 758-763, June 1998

[72] Mosisiads Y. and Bouras I., Differential CMOS edge-triggered flip-flop based on clock racing, Electronics Letters, Vol 36, No. 12, pp. 1012-1013, 2000

[73] Najm F., Goel. S. and Hajj I., Power estimation in sequential circuits, ACM/IEEE Design Automation Conference, San Francisco, USA, pp.635-640, June 1995

[74] Narayanan U. and Liu C. L. Low power logic synthesis for XOR based circuits, International Conference on Computer-Aided-Desigm, San Jose, USA, pp. 570-574, Nov. 1997

[75] Nemani M. and Najm N., Towards a high-level power estimation capability, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, Vol.15, No. 6, pp. 588-598, 1996

[76] Oldfield J., Dorf R., Field programmable gate arrays, John Wiley & Sons, Inc., New York, 1995

[77] Olson E. and Kang S., Low-power state assignment for finite state machines search, Proceedings of IWLPD'94: ACM/IEEE International Workshop on Low Power Design, Napa Valley, pp. 63-68,1994

[78] Panda P., Dutt N., Reducing address bus transitions for low power memory mapping, IEEE Europena Design and Test Conference, Paris, pp. 63-67, March 1996

[79] Parrilla L., Ortega J. and Lloris A., Nondeterministic AND-EXOR minimisation by using rewrite rules and simulated annealing, IEE Proc., Comput. Digit. Tech., Vol. 146, No. 1, pp.1-8, 1999

[80] Pedram M, Wu Q and Wu X, A new design of double edge triggered flip-flops, Proceedings of the Asia and South Pacific Design Automation Conference, Yokohama, Japan, pp. 417-421, Feb. 1998

[81] Perry D., VHDL, 3rd edition, McGraw-Hill Inc., New York, 1998

[82] Prosser F., Wu X., and Chen X., 1988, Ternary CMOS flip-flops and their applications, IEE Proceedings, Vol. 135E, pp. 256-272, 1988

[83] Rabaey J. and Pedram M., Low power design, Kluwer Academic Publishers, USA, 1996

[84] Raghunathan A., Dey S. and Jha N., Wakabayashi K., Power management techniques of control-flow intensive design, ACM/IEEE Design Automation Conference, California, USA, pp.429-434, 1997

[85] Raghunathan A. and Kja N., Behavioral synthesis for low power, Proceedings of the International Conference on Computer Design, Cambridge, USA, pp. 318-322, Oct. 1994

[86] Rashid M., SPICE for circuits and electronics using PSpice, Prentice Hall, New Jersey, 1995

[87] Reddy S., Easily testable realization for logic functions, IEEE Transaction on Computers, No.11, pp. 1183-1188, 1972

[88] Roy K. and Prasad S., Syclop: Synthesis of CMOS logic for low power application, Proceedings of IEEE International Conference on Computer Design, Cambridge, USA, pp. 464-467, 1992

[89] Roy K. and Prasad S., Low-power CMOS VLSI circuit design, John Wiley & Sons, Inc., New York, 2000

[90] Roy S., Banerjee P. and Sarrafzadeh M., Partitioning sequential circuits for low power, Proceedings of the IEEE International Conference on VLSI Design, India, pp. 212-217, Jan. 1998

[91] Panda R. and Najm F. Technology decomposition for low-power synthesis, IEEE Custom Integrated Circuits Conference, Santa Clara, CA, pp. 627-630, May 1995.

[92] Sasao T. and Fujita M. Representations of discrete functions, Kluwer Academic Publishers, USA, 1996

[93] Semiconductor Industry Association, Workshop Working Group Reports, Irving, TX, 22-23, 1992

[94] Sentovich E. M., Singh K. J., Lavagno L. Moon C., Murgai R., Saldanha A., Savoj J., Stephan P. R., Brayton R. K., and Sangiovanni-Vincentelli A., SIS: A system for sequential circuit synthesis, Electronics Research Laboratory, Memorandum No. UCB/ERL M92/41, UC, Berkeley, May, 1992

[95] Shahidi G., et al., $0.1\mu m$ CMOS devices, Proceddings of the Third Great Lakes Symposium on VLSI Technology, pp.67, 1993

[96] Singh D., Prospects for low power microprocessor design, International Workshop on Low Power Design, Napa, Cal., p. 1, 1994

[97] Song N. and Perdowski M., New fast approach to approximate ESOP minimization for incompletely specified multi-output functions, Proc. Reed-Muller'97 Conference, Oxford Univ., UK, pp 67-72, 1997

[98] Stan M., Burleson W., Bus-invert coding for low-power I/O, IEEE Trans on VLSI Systems, Vol. 3, No. 1, pp.49-58, 1995

[99] Stojanovic V. and Oklobdzija V. G. , Comparative analysis of master-slave latches and flip-flops for high-performance and low-power system, IEEE Journal of Solid-State Circuits, Vol. 34, No. 4, pp. 536-548, 1999

[100] Strollo A. and Caro D., Low power flip-flop with clock gating on master and slave latches, Electronics Letters, Vol. 36, No. 4, pp. 294-295, 2000

[101] Strollo A, Napoli E and Caro D, Low-power flip-flops with reliable clock gating, Microelectronics Journal, Vol. 32, pp.21-28, 2001

[102] Strollo E. Napoli E. and Cimino C., Low power double edge-triggered flip-flop using one latch, Electronics Letters, Vol. 35, No. 3, pp. 187-188, 1999

[103] Thomas D., The Verilog Hardware Description Language, Fourth Edition, Kluwer Academic Publishers, Lowell, MA, 1998

[104] Tiwari V., Ashar P. and Malik S., Technology mapping for low power, Proceeding of the Design Automation Conference, Dallas, Texas, pp. 74-79, June 1993

[105] Tran A. adn Wang J., Decomposition method for minimisation of Reed-Muller polynomials in mixed-polarity, IEE Proc-E, Vol. 140, No.1 pp. 65-68, 1993

[106] Trivedi K., Probability and statistics with reliability, queueing, and computer science applications, Prentice-Hall, USA, 1982

[107] Tsai C. and Marek-Dadowska M., Multilevel logic synthesis for arithmetic functions, Design Automation Conference, Las Vegas, pp. 68-73, June 1996

[108] Tsui C., Pedram M. and Despain A., Low power state assignment targeting two- and multilevel logic implementations, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No.12, pp. 1281-1291, 1998

[109] Tyagi A., Entropic Bounds on FSM switching, IEEE Transactions on VLSI Systmes, Vol.5, No.4, 456-464, 1997

[110] Unger S., Double-edge-triggered flip-flops, IEEE Tans. on Computers, Vol. C-30, No.6, pp.447-451, 1981

[111] Villa T., Sangiovanni-Vincentelli A., NOVA: State assignment of finite state machines for optimal two-level logic implementations, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 9, No.9, pp. 905-924, 1990

[112] Vranesic, Z., Multiple-valued logic: an introduction and overview, IEEE Transactions on Computers, Vol. C-26, pp. 1181-1182, 1977

[113] Wang K., Wang W., Hwang T., Wu A. and Lin Y., State assignment for power and area minimization, Proceedings of IEEE International Conference on Computer design, Cambridge, USA, pp. 250-254, Oct. 1994

[114] Wang L., Automated synthesis and optimization of multilevel logic circuits, PhD thesis, Napier University, UK, 2000

[115] Wang L. and Almaini A. E. A., Fast conversion algorithm for very large Boolean functions, Electronics Letters, Vol. 36, No. 16, pp. 370-1371, 2000

[116] Wang L. and Almaini A. E. A., Exact minimisation of large multiple output FPRM functions, IEE Proc.-Comput. Digit. Tech., Vol. 149, No. 5, pp. 203-212, 2002

[117] Wang, S. and Horng, M., State assignment of finite state machines for low power applications, Electronics Letters, Vol. 32, No. 25, pp. 2323-2324, 1996

[118] Weste N., and Eshragian K., Principles of CMOS VLSI design: a systems perspective, Second Edition, Addison-Wesley Publishing Company, Santa Clara, 1992

[119] Woodford C., Solving linear and non-linear equations, Ellis Horwood Limited, England, 1992

[120] Wu Q., Pedram M and Wu X, Clock-gating and its application to low power design of sequential circuits, IEEE Trans. on Circuits and Systems, Part 1, Vol.47, No. 3, pp. 415-420, 2000

[121] Wu X., and Chen X., Quaternary CMOS circuits based on transmission function theory, Chinese Science (Series A), No. 5, pp. 528-536, 1989

[122] Wu X., Shen J., and Chen X., CMOS multivalued flip-flops based on new presetting scheme and transmission function theory, Proceedings of International Workshop on Spectral Technique, Beijing, pp. 74-77, 1994

[123] Wu X., Chen B. and Pedram M., Power estimation in binary CMOS circuits based on multiple-valued logic, Journal of Multiple Valued Logic, Gordon and Breach Publishing Group, Vol. 7, No. 3-4, pp. 195-211, 2001

[124] Xia Y. and Almaini A. E. A., Genetic algorithm based state assignment for power and area optimisation, IEE Proc.-Comput. Digit. Tech., Vol. 149, No.4 , 128-133, 2002

[125] Xia Y. and Almaini A. E. A., Differential CMOS edge-triggered flip-flop with clock-gating, Electronics Letters, Vol. 38, No.1, pp. 9-11, 2002

[126] Xia Y. and Almaini A. E. A. , Best polarity for low power XOR gate decomposition, Proceedings of Euromicro Symposium on Digital System Design, Dortmund, Germany, pp. 53-59, Sept. 2002

[127] Xia Y., Almaini A.E.A. and Wu X., Power minimization of FPRM function on polarity conversion, Journal of Computer Science Technology (Accepted)

[128] Xia Y., Almaini A.E.A., Area and power optimization of FPRM function based circuits, IEEE International Symposium on Circuits and Systems, Bangkok, Thailand, May 25-28, 2003 (Accepted)

[129] Yang S., Logic synthesis and optimization benchmarks user guide, Microelectronics Center of North Carolina, 1991

[130] Yim J. and Kyung C., Datapath layout optimization using genetic algorithm and simulated annealing, IEE Proc.-Comput. Digit. Tech., Vol. 145, No. 2, pp.135-141, 1998

[131] Zhou H. and Wong D. F., Optimal low power XOR gate decomposition, Design Automation Conference, Bergen, Norway, pp.104-107, Aug. 2000

[132] Zhuang N. and Wu H., Novel ternary JKL flip-flops, Electronics Letters, Vol. 26, pp. 1145-1146, 1990

[133] Zukeran, C., et al, 1985, Design of new low-power quaternary CMOS logic circuits based on multiple ion implants, IEEE International Symposium on Multiple-Valued Logic, Kingston, pp. 84-90, 1985

Disk Containing the Software

The attached floppy disk contains the programs developed in the previous chapters.

- GA based FSM low power state assignment: Shell script file and C source files
- GA based FSM power and area optimisation: Shell script file and C source files
- Low power XOR gate decomposition: C source files
- FPRM function power optimisation: FPRM function power minimisation; FPRM function power and area optimisation