

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

利用與資料相依之延遲改善運算單元之能量效率

Improving Energy Efficiency of Functional Units by
Exploiting its Data-Dependent Latency



研究生：林彥呈

指導教授：劉志尉

中華民國九十七年十一月



利用與資料相依之延遲改善運算單元之能量效率

**Improving Energy Efficiency of Functional Units by Exploiting its
Data-Dependent Latency**

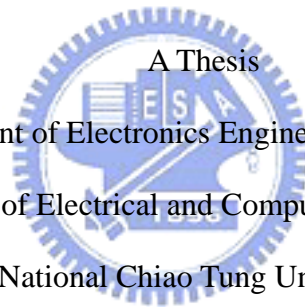
研 究 生：林彥呈

Student: Yen-Cheng Lin

指 導 教 授：劉志尉 博士

Advisor: Dr. Chih-Wei Liu

國 立 交 通 大 學
電 子 工 程 學 系 電 子 研 究 所 班
碩 士 論 文



Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University

In partial Fulfillment of the Requirements for the Degree of
Master of Science

in

Electronics Engineering

November 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 七 年 十 一 月



利用與資料相依之延遲改善運算單元之能量效率

研究生：林彥呈

指導教授：劉志尉 博士

國立交通大學

電子工程學系 電子研究所

摘要

隨著可攜式裝置的需求持續增加，以及多媒體和通訊應用所需要的運算能力也越來越強，因此降低能量消耗已經變成電路設計中主要的考量因素。在一般同步數位電路中，為了確保運算都能正確無誤，合成電路的時序限制(timing constraint)都會根據所希望的操作時脈來設定，但是當操作時脈拉高時，電路所消耗的能量會隨著時序限制的變緊而劇烈增加。在本篇論文中，我們提出一個改善運算單元(functional units)能量效率的方法，而這個方法主要是利用資料相依延遲(data-dependent latency)的特性來放鬆合成電路時的時序限制，如此可以有效降低運算單元的能量消耗而且不需要降低運算單元的操作時脈，這樣的方式雖然會造成一些資料的運算錯誤以及修復錯誤所花費的效能代價，但是在運算單元裡的最長路徑通常只會被少數的特定資料所感應(sensitize)，大部分的資料所需要的運算時間都小於操作時脈，所以只需要很小的效能代價。因此，我們設計一個偵測單元(detection logic)來偵測這些會造成運算錯誤的少數的資料，並且額外多花費一個週期的運算時間來修復錯誤，此外，我們也提出一個系統化的方法來設定運算單元合成時的時序限制以及調整資料運算錯誤的發生機率，讓我們可以利用最小的效能代價來節省最大的能量消耗。在我們的模擬中，我們利用所提出的方法來改善一個 8-bit 乘法器的能量效率，並且利用隨機資料(random pattern)，還有色彩空間轉換(color space transform)和有限脈衝響應(FIR)等測試程式來分析，在製程環境是 UMC 90nm CMOS cell library 下，和傳統設計方法相比可以有效降低 10%~29%的能量消耗，而所花費的效能代價是非常小的甚至是可以忽略的(<1%)。



Improving Energy Efficiency of Functional Units by Exploiting its Data-Dependent Latency

Student: Yen-Cheng Lin

Advisor: Dr. Chih-Wei Liu

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

ABSTRACT

For the increasing demand of portable devices and high computing power requirement for the multimedia and communication applications, the energy reduction now has become a major issue in the circuit design. In the synchronous digital circuits, synthesis timing constraint is based on the desired clock period for function correctness, but the circuit energy increases drastically when the synthesis timing constraint and clock frequency approach peak value (peak speed). In this thesis, we propose a design method for improving energy efficiency of functional unit. It exploits data-dependent latency to relax synthesis timing constraint so that the energy consumption can be reduced effectively and the desired clock period will not be degraded, but it will cause some computation errors and spend performance penalty for correcting the errors. Critical paths of the circuits are usually sensitized by very specific data sequences, and computation time of most data sequences is smaller than clock period. Hence the performance penalty may be very small. Because of this property we generate the detection logic to detect the specific data sequences that will cause computation errors and spend one-cycle latency penalty to correct the errors. Besides, we also propose a design flow which systematically determines the synthesis timing constraint and fine tunes the error rates, therefore we can trade the minimum performance penalty for the maximum energy reduction. In our simulations, we use the proposed technique in an 8-bit multiplier to improve the energy efficiency with the UMC 90nm CMOS cell library. The benchmarks consist of random patterns, color space transform and FIR. Compared with conventional synthesis strategy, the proposed method can reduce 10% ~ 29% of energy consumption and the performance penalty is negligible (<1%).



誌 謝

研究生涯轉眼即逝，兩年來受到許多人幫助及鼓勵，才能順利完成碩士學業，在此致上最深的感激。

感謝劉志尉教授的照顧。使我在專業知識及研究態度上更臻成熟。此外，特別感謝任建葳教授、周景揚教授及周世傑教授，謝謝你們在百忙之中，撥冗參與論文口試，並對我的研究給予寶貴的意見，讓此篇論文更加完備充實。

感謝林泰吉學長在研究上的指導，而且時常在我遇到困難時伸出援手，讓我適時得到幫助，並培養我做人做事上應有的態度。另外感謝 Hardware team 的成員，歐士豪學長、呂進德同學以及甘禮源學弟平時在研究和生活的諸多幫忙和協助。

感謝實驗室學長，感謝陳信凱學長、郭羽庭學長、林禮圳學長、鄧翔升學長、卓志宏學長、劉士賢學長、陳慶至學長、王炳勛學長、卓毅學長、林佑昆學長以及張彥中學長，感謝你們平時對於我提供的諸多協助和意見。

感謝實驗室同學及學弟妹，感謝顏于凱同學、洪正堉同學、張巍瀚同學以及李岳泰同學，我們一同經歷了許多的努力和奮鬥，這回憶我永生難忘。感謝張國強學弟、莊明勳學弟、葉世賢學弟、吳聲昀學弟、張雅婷學妹、蔡安綺學妹，謝謝你們在我研究生活上的一切幫忙。

最後，感謝我最親愛的家人。爸、媽、姊以及芝瑄，感謝你們一路上的支持及鼓勵，沒有你們就沒有今日的我，我愛你們。

謹將此篇論文獻給所有曾支持我、協助我的人，衷心的感謝並祝福你們。

彥呈

謹誌於 新竹

2008 冬



CONTENTS

1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement and Related Works	3
1.3 Thesis Organization	5
2 Low-Power & Energy-Efficient Design Methods	7
2.1 CMOS Power/Energy Dissipation	8
2.1.1 Static Power Dissipation	8
2.1.2 Dynamic Power Dissipation.....	11
2.2 Static Approaches	14
2.2.1 Supply Voltage	15
2.2.2 Switching Activity and Capacitance	17
2.3 Dynamic Approaches	20
2.3.1 Supply Voltage	20
2.3.2 Switching Activity and Capacitance	21
2.4 Adaptive Approaches	24
2.4.1 Supply Voltage	25
2.4.2 Switching Activity and Capacitance	25
3 Proposed Energy-Efficient Design	29
3.1 Delay of CMOS Circuits	29
3.2 Variable Latency Design	34
3.2.1 Template of Variable Latency Design	34
3.2.2 Detection Logic.....	37
3.3 Design Flow	48
3.3.1 Characterization	50
3.3.2 Overhead Estimation.....	53
3.3.3 Overhead Reduction.....	56
4 Experimental Results	61
4.1 Simulation Results of Energy-Efficient Design	61
4.2 Comparison of Energy Reduction Techniques	63
5 Conclusions	71
Reference	73



LIST OF FIGURES

FIGURE 1-1 POWER DENSITY TREND	2
FIGURE 1-2 ENERGY V.S TIMING CONSTRAINT	3
FIGURE 1-3 EXPLOITING PARALLELISM.....	4
FIGURE 2-1 CMOS INVERTER MODEL.....	9
FIGURE 2-2 MODEL DESCRIBING PARASITIC DIODES PRESENT IN A CMOS INVERTER.....	10
FIGURE 2-3 MODEL DESCRIBING PARASITIC DIODES PRESENT IN A CMOS INVERTER.....	12
FIGURE 2-4 SWITCHING POWER IN A CMOS INVERTER.....	13
FIGURE 2-5 PARALLEL AND PIPELINED DATAPATH.....	16
FIGURE 2-6 EXAMPLE OF OPERATOR REORDERING	18
FIGURE 2-7 GATE RESTRUCTURING	19
FIGURE 2-8 ARCHITECTURE OF THE DVFS SYSTEM.....	21
FIGURE 2-9 CLOCK-GATED D FLIP-FLOP	22
FIGURE 2-10 PRE-COMPUTATION LOGIC	23
FIGURE 2-11 COMPUTATIONAL KERNEL	24
FIGURE 2-12 ARCHITECTURE OF THE AVS SYSTEM.....	25
FIGURE 2-13 EXAMPLE OF BIT SWAPPING	26
FIGURE 2-14 PATH DELAY DISTRIBUTION	27
FIGURE 2-15 CONCEPTUAL CIRCUIT OF PROPOSED DESIGN	28
FIGURE 3-1 PATH DELAY DISTRIBUTION (8-BIT MULTIPLIER).....	31
FIGURE 3-2 ENERGY CURVE OF 8-BIT MULTIPLIER	32
FIGURE 3-3 PATH DELAY DISTRIBUTION (8-BIT MULTIPLIER).....	33
FIGURE 3-4 TEMPLATE OF VARIABLE LATENCY DESIGN.....	35
FIGURE 3-5 DETECTION LOGIC.....	36
FIGURE 3-6 TIMING DIAGRAM OF THE VARIABLE LATENCY DESIGN	36
FIGURE 3-7 DETECTION LOGIC GENERATION	38
FIGURE 3-8 EXACT PATH SENSITIZATION CRITERION	42
FIGURE 3-9 DELAY-DEPENDENT FALSE PATH.....	42
FIGURE 3-10 VIABLE PATH SENSITIZATION CRITERION	43
FIGURE 3-11 AN EXAMPLE OF GIVEN CRITICAL PATH.....	44
FIGURE 3-12 MODIFIED VIABLE PATH SENSITIZATION CRITERION.....	45
FIGURE 3-13 FLOWCHART OF FAULT FUNCTION.....	46
FIGURE 3-14 TWO VIOLATING PATHS.....	47
FIGURE 3-15 DESIGN FLOW OF ENERGY-EFFICIENT FUNCTIONAL UNIT	49
FIGURE 3-16 CHARACTERIZATION RESULTS	52

FIGURE 3-17 CHARACTERIZATION RESULTS (SUB-SET CELL LIBRARY)54

FIGURE 3-18 OVERHEAD ESTIMATION (CLOCK PERIOD IS 1.7NS).....55

FIGURE 3-19 OVERHEAD ESTIMATION (CLOCK PERIOD IS 2NS).....56

FIGURE 3-20 RE-SYNTHESIS FLOW58

FIGURE 3-21 BEFORE OVERHEAD REDUCTION (RE-SYNTHESIS).....59

FIGURE 3-22 AFTER OVERHEAD REDUCTION (RE-SYNTHESIS).....60

FIGURE 4-1 BASELINE DSP MODEL.....64

FIGURE 4-2 DSP MODEL WITH PARALLEL MULTIPLIER DESIGN65

FIGURE 4-3 DSP MODEL WITH PIPELINE MULTIPLIER DESIGN65

FIGURE 4-4 DSP MODEL WITH PROPOSED MULTIPLIER DESIGN.....66

FIGURE 4-5 AREA IS NORMALIZED BY THE RESULT OF SINGLE CYCLE DESIGN67

FIGURE 4-6 ASSEMBLY CODE OF CHROMA68

FIGURE 4-7 NORMALIZED BY THE RESULT OF SINGLE CYCLE DESIGN69

FIGURE 4-8 ENERGY-DELAY VALUE NORMALIZED BY SINGLE CYCLE DESIGN.....70



LIST OF TABLES

TABLE 3-1 CHARACTERIZE MULTIPLIER FOR DIFFERENT CLOCK PERIODS	53
TABLE 4-1 ESTIMATED AREA.....	62
TABLE 4-2 ESTIMATED ENERGY PER MULTIPLICATION.....	63
TABLE 4-3 ESTIMATED TOTAL AREA.....	66
TABLE 4-4 SIMULATION RESULTS.....	69





1 INTRODUCTION

1.1 Motivation



As number of transistor is doubled every technology generation, chips grow in functionality and switching frequencies [1]. The millions of parasitical capacitances charging and discharging at an ever-increasing rate have led to a soaring amount of power dissipation. For desktop computers, the high power densities reduce chip reliability and life expectancy [2], shown in Figure 1-1. To keep the system stable, the cooling system is necessary and thus induces additional cost.

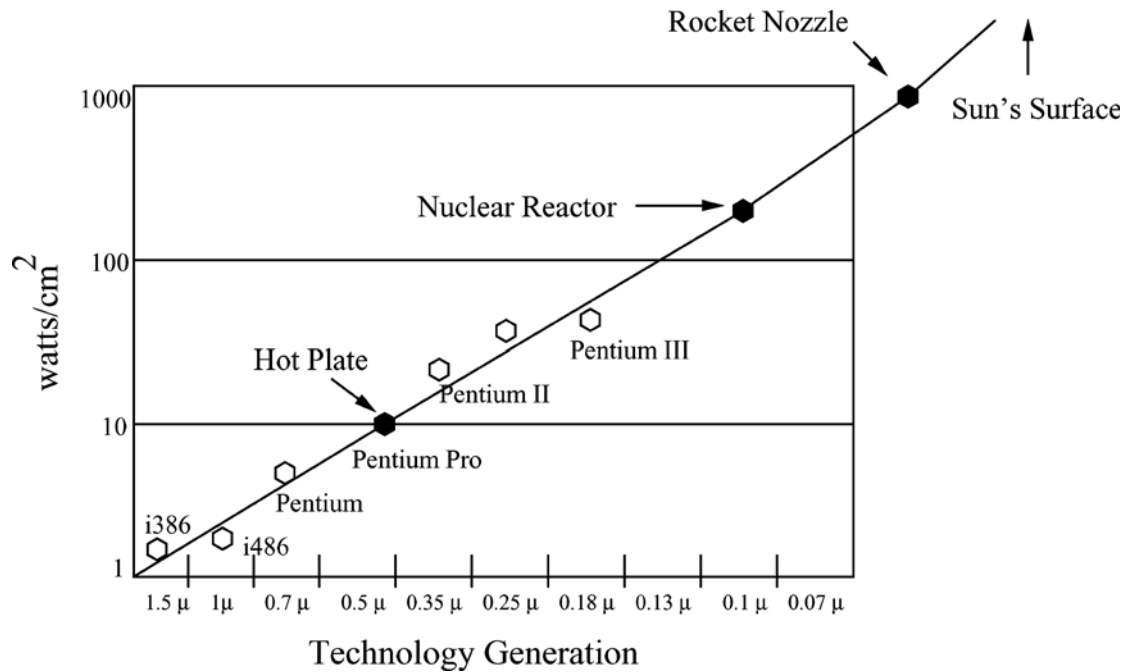


Figure 1-1 Power density trend [2]

With today's growing popularity of portable devices such as cell phones, PDAs, and laptops, low energy consumption is also one of important issues in VLSI designs. The duration of operation time is limited by the battery capacity in microamp-hour or watt-hour. Devices that operate with high energy consumption can only be used for short duration of time. The duration of operating time can be lengthened by using a battery with higher capacity. Unfortunately, projected improvements in the capacity of batteries (5-10% CAGR [3]) are much slower than what is needed to support the increasing complexity, functionality and performance of the systems they power [4]. The need to improve battery life time has driven the research and development of low power and energy-efficient design techniques for electronic circuits and systems. The low power and energy-efficient circuit design methods are discussed in Chapter 2.

Generally, the DSP processor is the key component of portable system, and it always dominates the performance of the system and consumes large amount of

energy from battery. The functional units in the DSP processor are used to implement multimedia processing, and they always dominate the clock frequency and determine the performance of the processing. However, the demand for high-performance portable systems incorporating multimedia capabilities has elevated the design for low-energy to the forefront of design requirement in order to maintain reliability and provide longer hours of operation.

1.2 Problem Statement and Related Works

High performance and low energy always represent contradictory design requirements. In our experiments, we found the relationship between energy and timing constraint, and it is shown in Figure 1-2.

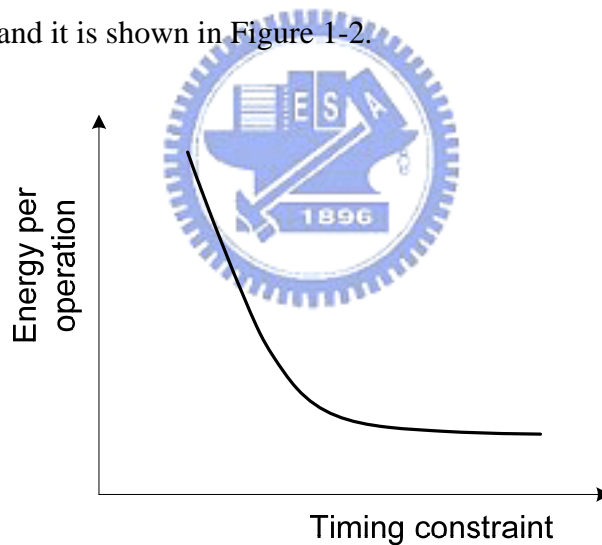


Figure 1-2 Energy v.s timing constraint

Tightening the timing constraint of the circuit will induce the increasing of the circuit energy consumption. Especially when the timing constraint approaches the peak value (the minimum value), the circuit energy increases drastically. The reason for the soared energy is to use the cells with high driving ability and low threshold voltage, improve the circuit structure and use high supply voltage [5]. Hence,

relaxing timing constraint can reduce the circuit energy effectively, but it makes clock frequency degradation.

Exploiting parallelism is one of the effective methods to compensate for the loss in clock frequency [6]. Figure 1-3 (a) shows the original circuit with sample rate $1/T$, and the timing constraint of the multiplier is T . Figure 1-3 (b) shows the multiplier with the timing constraint of the multiplier is relaxed to $2T$, and therefore the energy consumption is reduced. Although the energy is reduced, the clock frequency is degraded to $1/2T$ for functional correctness and the sample rate is $1/2T$. Exploiting parallelism can compensate the clock frequency degradation and gain the energy reduction, shown in Figure 1-3 (c) and (d).

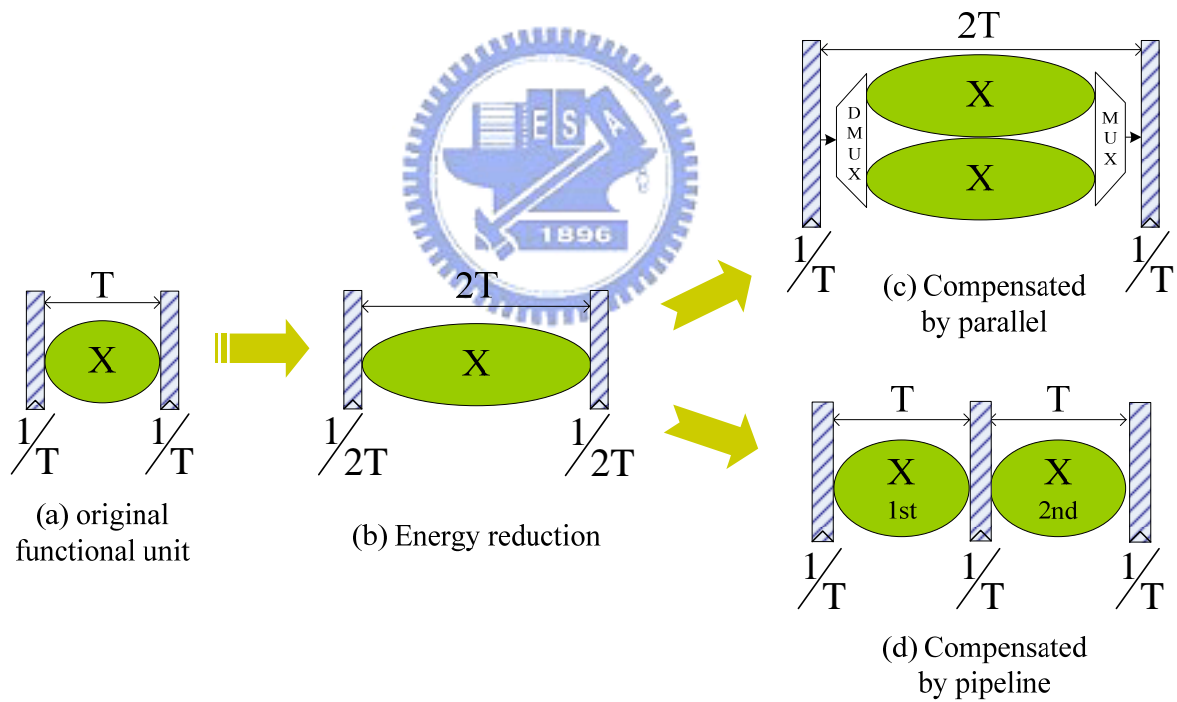


Figure 1-3 Exploiting parallelism

Parallel design (Figure 1-3 (c)) uses two low energy multipliers (Figure 1-3 (b)) to interleave computation, hence both the sample rate and clock frequency are $1/T$. The average energy per operation is almost the same as Figure 1-3 (b), but it needs

mux and de-mux to control the computation. Pipeline design (Figure 1-3 (d)) inserts registers into the low energy multiplier (Figure 1-3 (b)) to compensate the clock frequency and sample rate. The overhead is pipeline registers. The main drawback of these two designs is data latency. Therefore, they require high data parallelism to hide latency. If the instruction level parallelism in a program is inefficient and limited, the design will incur serious data hazards and many stall cycles.

In the thesis, we exploit the circuit data-dependent delay [7][8][9] to achieve energy reduction. In many designs, the critical paths may be activated infrequently. Hence we propose a method to relax the synthesis timing constraint for energy reduction, and it doesn't make clock frequency degradation. Because of this property we generate the detection logic to detect the specific data sequences that will cause computation errors and spend one-cycle latency penalty to correct the errors. Besides, we also propose a design flow which systematically determines the synthesis timing constraint and fine tunes the error rates, therefore we can trade the minimum performance penalty for the maximum energy reduction.

1.3 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 introduces several techniques of energy reduction the techniques are classified into categories. The proposed design method is also described. Chapter 3 addresses the data-dependent delay and the template of the variable latency design. The variable latency design can tolerate the computation errors by two-cycle latency, and the detection logic that is included in the variable latency design is responsible to detect the computation errors. Moreover, the design flow for the proposed energy-efficient circuit is also introduced in the Chapter 3. Chapter 4 shows simulation results consist of area,

energy and performance. The proposed technique is compared with parallel design and pipeline design. Finally, Chapter 5 concludes this thesis and describes the future works.



2 LOW-POWER & ENERGY-EFFICIENT DESIGN METHODS



Power dissipation and energy consumption are critical factors in the design of any system-on-chip. For battery-powered applications, these are extremely important because they govern battery lifetime and users always value products that run longer on a battery change. Power dissipation of digital CMOS circuits consists of static power dissipation and dynamic power dissipation. Static power dissipation is defined as the power that is dissipated without any switching in the circuit. Leakage current is the major source of static power dissipation. On the other hand, the dynamic power dissipation is due to the signal transitions in the circuit. Dynamic power consists of two parts: the first (switching power) is caused by charging and

discharging load capacitance of gates, and another (short circuit power) is due to the conduction path between supply voltage and ground appears during signal transition.

Static power consumption has been traditionally ignored since it has been negligible; however it is becoming more significant with the downward-scaling of transistor dimensions. Therefore opportunities for significant power reduction are available in both static and dynamic power.

This chapter focuses on methods for dynamic power (dynamic energy) reduction mainly. The methods of dynamic power (energy) reduction are classified to three approaches (static approaches \ dynamic approaches \ adaptive approaches), they would be discussed in detail below.

2.1 CMOS Power/Energy Dissipation

Power dissipation in CMOS circuits can be divided into three main components: short-circuit power, switching power, and leakage power. Although the terms “power” and “energy” have different definitions, both serve to achieve the same objective [2]. Power is defined as the average power that is supplied to a circuit from the power supply and is measured in “watts”. Meanwhile, the term energy refers to the total amount of power dissipation over a period of time. Energy is measured in “joules”. In fact, energy can be expressed in terms of the power-delay product that is shown in equation (2-1), which is the product of power consumption and execution time.

$$\text{Energy} = \text{Power} \times \text{Time} \quad (2-1)$$

2.1.1 Static Power Dissipation

Ideally, the CMOS circuits dissipate no static power since there is no direct path from V_{DD} to ground in the steady state. In Figure 2-1 shows a CMOS inverter model, for a complementary CMOS circuit, if V_{in} is “0” that N-MOS will be turned off while the P-MOS is turned on. The output voltage (V_{out}) will be “1”. On the opposite, if V_{in} is “1” that N-MOS will be turned on while the P-MOS is turned off. The output voltage (V_{out}) will be “0”. From this scenario, the CMOS circuit has no direct path from V_{DD} to ground, therefore it will not induce the static power dissipation. However, the scenario of ideal CMOS circuit cannot be realized in practice since the MOS transistor is not a perfect switch. There are some small static dissipation due to reverse bias leakage current (junction reverse bias current) between the diffusion regions and the substrate [10]. In addition, “sub-threshold conduction current” and “gate-induced drain leakage” can contribute to the static power dissipation.

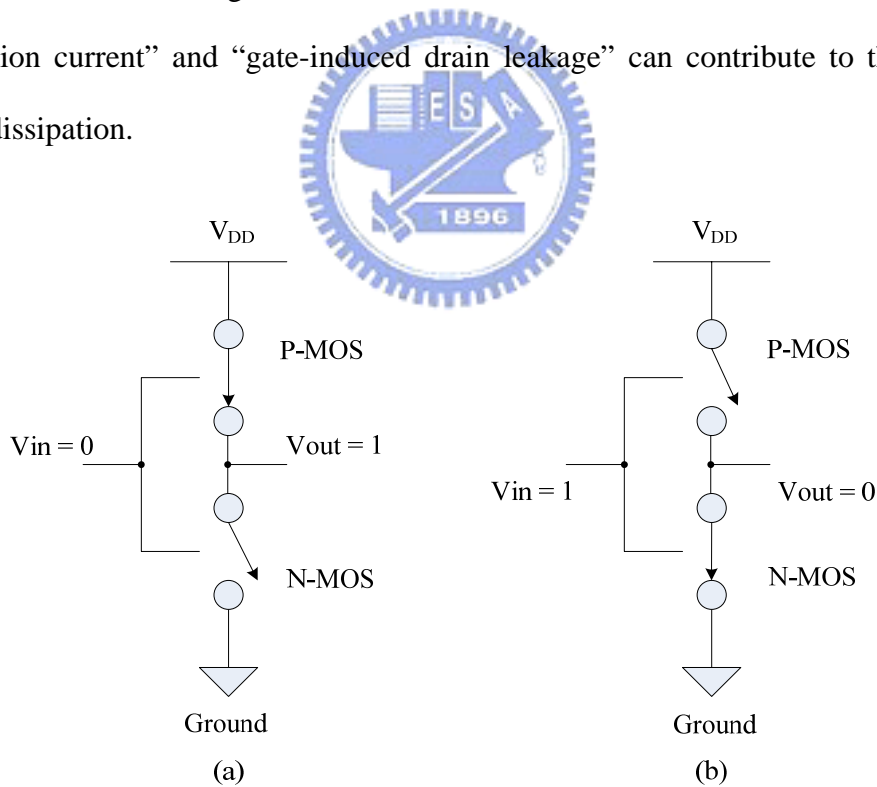


Figure 2-1 CMOS inverter model

The reverse bias leakage current is due to the parasitic diodes existing in CMOS transistor. To give a comprehensive explanation, Figure 2-2 depicts the parasitic

diodes in a CMOS inverter. Consider when V_i equals ground, the NMOS is turned off, and the PMOS is turned on. Thus V_o is driven to high, and parasitic diode made of n+ diffusion and p-substrate is reversely biased. That is, there will be a diode reverse saturation current drawn from supply to ground.

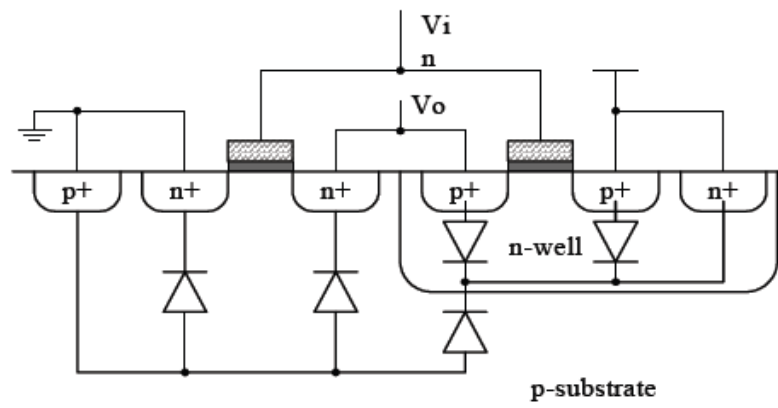


Figure 2-2 Model describing parasitic diodes present in a CMOS inverter

The sub-threshold conduction current is between source and drain when supply voltage is below threshold voltage. The gate-induced drain leakage current arises in the high electric field under gate and drain overlap region causing a thinner depletion region of drain to well junction.

The static power dissipation equals the product of device leakage current and supply voltage. Equation (2-2) represents the static power dissipation, where “ $I_{leakage}$ ” is a sum of all leakage currents. The static power is independent of signal switching.

$$\text{Power}_{\text{static}} = V_{DD} \times I_{\text{leakage}} \quad (2-2)$$

The leakage current is related to the threshold voltage. Threshold voltage will affect the leakage current exponentially. Higher threshold voltage will result smaller

leakage current and smaller static power dissipation. But the high-threshold transistor takes longer time to complete a transition. Therefore, Dual threshold voltage [11][12] is a scheme of reducing leakage current by assigning some high-threshold voltage transistors in the non-critical paths, and using low-threshold transistors in the critical paths.

2.1.2 Dynamic Power Dissipation

The dynamic power dissipation consists of two parts: one is due to short-circuit current when both pull-up and pull-down transistors are momentarily on at the same time, another is due to switching current from charging and discharging parasitic capacitance of the CMOS circuits.

- **Short Circuit Power Dissipation**

The short circuit power dissipation is dependent on signal switching. During the output transfers from logic 1 to logic 0 or from logic 0 to logic 1, there exists a discharging path from supply voltage to ground for a short period. This is because of the rising time and falling time of PMOS or NMOS are not ideal zero. Taking CMOS inverter as an example, if the rising and falling time of input waveforms are not zero, when $V_{tn} < V_{in} < V_{DD} - |V_{tp}|$ holds for the input voltage, there will be a conductive path open between V_{DD} and ground because both the NMOS and PMOS devices are turned-on (where V_{tn} and V_{tp} are threshold voltages of NMOS and PMOS). The short circuit current is from V_{DD} to ground, as described in Figure 2-3.

On a low-to-high transition at the input, the NMOS will start to conduct when V_{in} is equal to V_{tn} , and the PMOS will stop conducting when V_{in} is equal to $\{V_{DD} - |V_{tp}|\}$. In this inverter example [6], the short circuit power is given by equation (2-3).

$$P_{\text{short-circuit}} = V_{DD} \times I_{\text{mean}} = \frac{\beta}{12} (V_{DD} - 2V_t)^3 \frac{t}{T_{clk}} \quad (2-3)$$

The “t” is the rising time or the falling time of the input signal, and $V_t = V_{tn} = |V_{tp}|$. Also, the effective transistor strengths are equal for the NMOS and PMOS; let $\beta = \beta_n W_n = \beta_p W_p$.

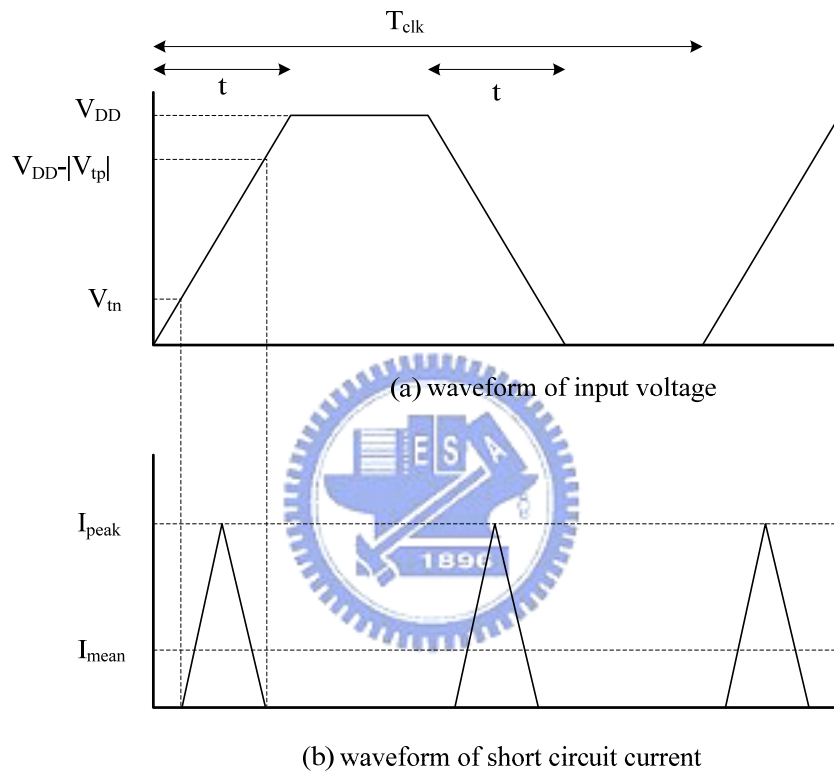


Figure 2-3 Model describing parasitic diodes present in a CMOS inverter

By the equation (2-3), short circuit current is significant when the rising or falling time at the input of a gate is much longer period of time, which means more significant short circuit dissipation. Thus to minimize the short circuit dissipation, it is desirable that the short-circuit dissipation is minimized by making the output rising or falling time larger than the input rising or falling times [6][13]. Careful design is required to keep this component of power dissipation small enough to be ignored [14].

- **Switching Power Dissipation**

The other part of dynamic power dissipation is due to signal switching of the nodes in the circuit. The power is dissipated when the circuit capacitance is charged to V_{DD} throughout the pull-up network (PMOS) and discharged to ground throughout the pull-down network (NMOS). Figure 2-4 describes the switching power in a CMOS inverter. The equation (2-4) is calculating the energy that needed to charge the circuit capacitance and the equation (2-5) is calculating the energy that will be discharged while pull-down network is turned on [6].

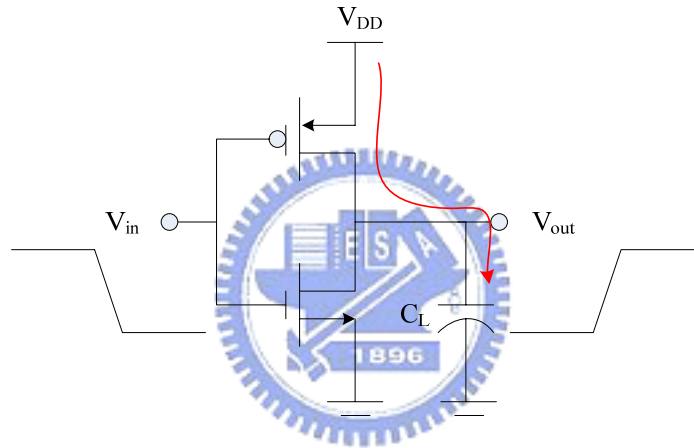


Figure 2-4 Switching power in a CMOS inverter

$$E_{V_{DD}} = \int_0^{\infty} i_{V_{DD}}(t) V_{DD} dt = V_{DD} \int_0^{\infty} C_L \frac{dV_{out}}{dt} dt = C_L V_{DD} \int_0^{V_{DD}} dV_{out} = C_L V_{DD}^2 \quad (2-4)$$

$$E_{C_L} = \int_0^{\infty} i_{V_{DD}}(t) V_{out} dt = \int_0^{\infty} C_L \frac{dV_{out}}{dt} V_{out} dt = C_L \int_0^{V_{DD}} V_{out} dV_{out} = \frac{C_L V_{DD}^2}{2} \quad (2-5)$$

It means that the capacitance only sustained half-the-energy that charged. Obviously, for each switching cycle (consisting of an $L \rightarrow H$ and an $H \rightarrow L$ transition) takes a fixed amount of energy, $C_L V_{DD}^2$.

In order to compute the power consumption, we have to take into account how

often the device is switched. If the gate is switched on and off “ f ” times per second, the power consumption is given by equation (2-6), where “ α ” is the switching activity factor which represents the probability of the switching from 0 to 1.

$$P_{\text{switching}} = \alpha C_L V_{DD}^2 f \quad (2-6)$$

So far we know that power dissipation is composed of static power dissipation and dynamic power dissipation. The total power consumption of the CMOS is the sum of its three components. The dynamic power dissipation (switching power dissipation) is the major source of total power dissipation, when the signal is switching [6]. From equation (2-1), we can derive the dynamic energy consumption (per transition) that is shown in equation (2-7).

$$\text{Energy} = \text{Power} \times \text{Time} \approx \alpha C_L V_{DD}^2 f \times \text{Time} = \alpha C_L V_{DD}^2 \quad (2-7)$$

The dynamic energy consumption is related to the supply voltage, switching activity and switching capacitance. Reducing energy consumption is independent of clock frequency. It is related to the supply voltage and circuit capacitance (switching activity and switching capacitance). If it is possible, using lowest voltage and smallest amount of capacitance will result the design with minimum energy dissipation, but it will slow down the path delay. On the other hand, increasing the supply voltage or gate size (capacitance) will improve the circuit delay, but it will increase the circuit energy.

Here I classify the methods of power and energy reduction into three categories: static approaches, dynamic approaches and adaptive approaches.

2.2 Static Approaches

This category is said that the circuit is optimized at design time and it is inflexible at run time. I'll introduce the techniques of energy reduction by reducing supply voltage and switch activity/capacitance respectively.

2.2.1 Supply Voltage

- **Algorithmic transformation**

The choice of algorithm is the most highly leveraged decision in meeting the power constraints. Transformations are changes of the computational structure in a manner that the input/output behavior is preserved. The use of transformations makes it possible to explore a number of alternative architectures and to choose which result in the lowest power. The key approach is reducing the supply voltage by minimizing the number of operations and exploitation of concurrency.

At algorithm level, minimizing the number of operations and exploitation of concurrency can increase the throughput such that the supply voltage can be reduced to meet the requirement. The example is a first order IIR filter shown in [15] that it is applying loop unrolling and algebraic transformations to exploit data concurrency. We also can design a FIR filter with polyphase decomposition to minimize the number of operations [16].

- **Parallelism & pipelining**

At the architecture level parallelism and pipelining are also the effective way to increase the circuit throughput and frequency such that the supply voltage can be reduced. Although it would increase the circuit capacitance (area), supply voltage is square proportioned to energy consumption such that energy can be reduced.

Show parallelism and pipelining examples in Figure 2-5 (b) and (c) respectively [5]. Figure 2-5 (a) is the original structure. Equation (2-8), equation

(2-9) and equation (2-10) represent the power dissipation of original datapath, parallel datapath and pipelining datapath respectively.

$$P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}} \quad (2-8)$$

$$P_{\text{par}} = (2.15C_{\text{ref}})(0.58V_{\text{ref}})^2 \left(\frac{f_{\text{ref}}}{2}\right) \approx 0.36P_{\text{ref}} \quad (2-9)$$

$$P_{\text{pipe}} = (1.15C_{\text{ref}})(0.58V_{\text{ref}})^2 f_{\text{ref}} \approx 0.39P_{\text{ref}} \quad (2-10)$$

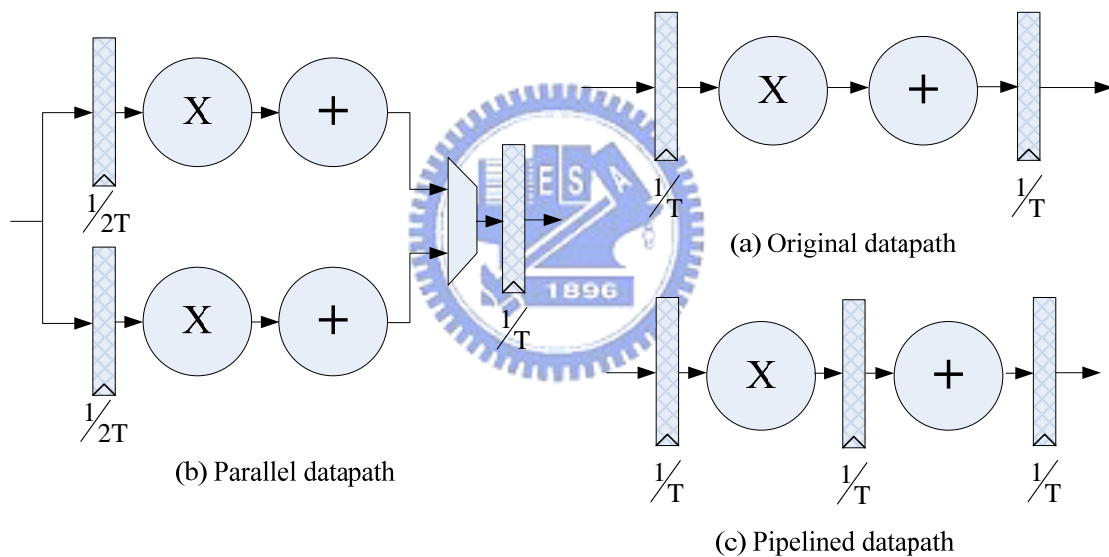


Figure 2-5 Parallel and pipelined datapath

- **Dual supply voltage**

The alternative approach for optimizing supply voltage is to selectively decrease the supply voltage on some of the gates based on the path delay. The critical paths are supplied by higher supply voltage, and the non-critical paths can be supplied by lower supply voltage. Using dual supply voltage in different parts of a circuit may reduce the energy consumption of a design at a rather small cost in terms of algorithmic

and/or architectural modifications [17][18][19][20].

Using dual supply voltage on the same circuit requires the use of level converters at the boundaries of the various modules (a level converter is needed between the output of a gate supplied by a low V_{DD} and the input of a gate supplied by a high V_{DD}).

2.2.2 Switching Activity and Capacitance

● Operation substitution & operator reordering

The switching activity and switching capacitance can be reduced by optimizing the ordering of operations and using operation substitution in a design. To illustrate this, consider the problem of multiplying a signal with a constant coefficient, which is a very common operation in signal processing applications.

Multiplications with constant coefficients are often optimized by decomposing the multiplication into shift-add operations and using the canonical sign digit representation. Thus the circuit area (capacitance) can be reduced. Consider the example in which a multiplication with a constant is decomposed into $IN + IN \gg 7 + IN \gg 8$, shown in Figure 2-6.

In the Figure 2-6 (b) (obtained by applying associativity and commutativity), the two small number $IN \gg 7$ and $IN \gg 8$ are summed in the first adder and the output is added to IN in the second adder. In this case, the output of the first adder has a small amplitude (since we are adding 2 scaled number of the same sign) and therefore lower switching activity. The second implementation switched 30% less capacitance than the first implementation [15].

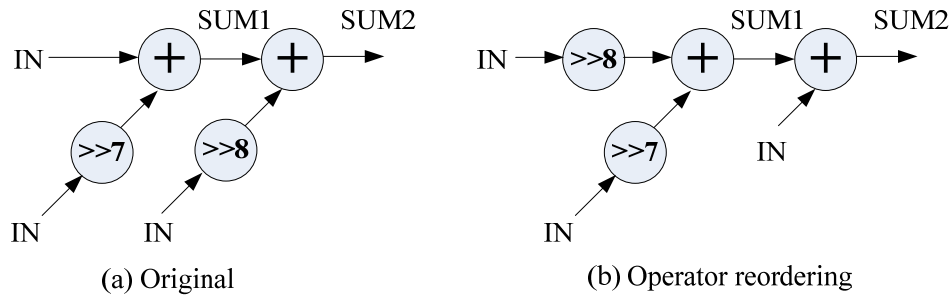


Figure 2-6 Example of operator reordering

- **Data representation & bus encoding**

In most signal processing applications, two's complement is typically chosen to represent numbers since arithmetic operations (addition and subtraction) are easy to perform. One of the problems with two's complement representation is sign-extension, which causes the msb sign-bits to switch when a signal transitions from positive to negative or vice-versa (for example, going from -1 to 0 will result in all of the bits toggling). Therefore using a two's complement representation can result in significant switching activity when the signals being processed switch frequently around zero and when they do not utilize the entire bit-width (i.e., the dynamic range is much smaller than the maximum possible value determined from the bit-width) since a lot of the msb bits will perform sign-extension.

Minimizing the switching in the msbs can use a sign-magnitude representation, in which only one bit is allocated for the sign and the rest for the magnitude [6][15]. In this case, if the dynamic range of a signal does not span the entire bit width, only one bit will toggle when the signal switches sign, as opposed to the two's complement representation where due to sign extension several of the bits will switch.

For the bus encoding, we also can use the gray code to substitute for binary code such that the signal transitions of the program and the data memory address busses

can be reduced. For sequentially access, the average toggling of binary and gray code are 2 and 1 respectively [6][15].

- **Logic reordering (circuit optimization)**

There are many ways to build a circuit out of logic gates. One decision that affects power consumption (glitch activity) is how to arrange the gates [6][15]. For example, consider two implementations of a four-input AND gate shown in Figure 2-7, a chain implementation (a), and a tree implementation (b).

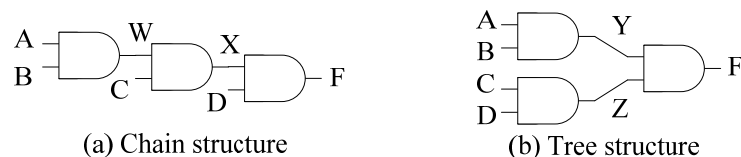


Figure 2-7 Gate restructuring

There is a issue of glitches or spurious transitions that occur when a gate does not receive all of its inputs at the same time. These glitches are more common in chain implementations where signals can travel along different paths having widely varying delays. One solution to reduce glitches is to change the topology so that the different paths in the circuit have similar delays. This solution, known as path balancing often transforms chain implementations into tree implementations.

- **Gate sizing**

Gate sizing is an effective method for circuit power-reduction, because the major power dissipation is consumed inside the block rather than in driving the external load capacitance. Reduce gate size can reduce circuit capacitance such that the circuit power consumption can be reduced. Applying this technique in [5][21] usually associate with each gate a tolerable delay which varies depending on how close that gate is to critical path. Then, we can try to scale each gate to be as small as

possible without violating its tolerable delay. The main objective of transistor sizing is to downsize the gate off the critical path to save power.

2.3 Dynamic Approaches

This category is said that the supply voltage and switch activity/capacitance can be adjusted dynamically for different applications and throughput requirement. It is more flexible than the static approaches. I'll introduce the techniques of energy reduction by reducing supply voltage and switch activity/capacitance respectively.

2.3.1 Supply Voltage

- **Dynamic voltage and frequency scaling (DVFS)**

The gap between high performance and low power can be bridged through the use of dynamic voltage scaling, where periods of low processor utilization are exploited by lowering the clock frequency to the minimum required level, allowing corresponding reduction in the supply voltage [22][23].

Figure 2-8 shows the overall architecture of a DVFS system. The performance manager uses a software interface to predict performance requirements. Once performance requirement for the next task is determined, the performance manager sets the voltage and frequency just necessary to accomplish the task. The target frequency is sent to the phase-locked loop (PLL) to accomplish frequency scaling. Based on the target voltage, the voltage regulator scales supply voltage to meet performance target.

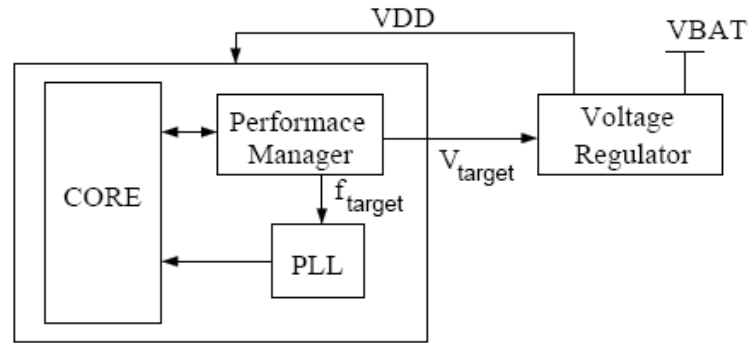


Figure 2-8 Architecture of the DVFS system

A robust system should be able to meet the deadlines at any voltage, process and temperature condition. The conventional approach performs voltage scaling that it uses a target operating voltage for each required operating frequency. To guarantee a robust operation, the frequency-voltage relationship is determined via chip characterization at worst case conditions. This technique is utilized in open-loop dynamic voltage and frequency scaling system where the frequency-voltage relationship is stored in a look-up table. Since such LUT (look up table) is pre-loaded with voltage-frequency points, DVFS systems are not able to adapt to process variations or environmental conditions.

2.3.2 Switching Activity and Capacitance

- **Clock gating & operand isolation**

Clock gating is a common method for reducing the unnecessary signal transitions. In [24], it proposes a technique to automatically synthesize gated clocks for finite-state machines to reduce power dissipation. The following graph (Figure 2-9) is a gated-clock D flip-flop.

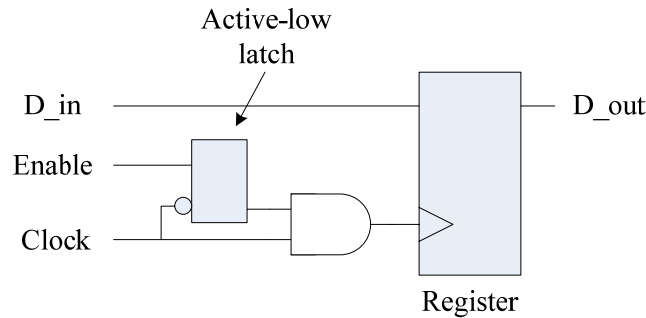


Figure 2-9 Clock-gated D flip-flop

There will be an additional signal named “Enable”. For a D flip-flop without gated-clock, the input will be passed to output at the rising edge of clock. The input of gated D flip-flop will only be passed to output at the rising edge of clock if the enable signal is “1”.

We can control the enable signal dynamically according to the different requirements. It reduces the signal transitions of register and combinational circuit. If the inputs of a circuit are gated, the inputs are the same with the ones in the previous cycle. And all the nodes in circuit remain unchanged. If the circuit is without gated-clock input registers, there might be some glitches in this cycle which consumes power also.

Hence, we also can insert latches (flip-flops) at the inputs of the functional units. If the output of the functional units is not necessary, the input data can be isolated using latches (flip-flops).

- **Pre-computation logic**

It relies on the idea of duplicating part of the logic with the purpose of pre-computing the circuit output values one clock cycle before they are required, and it uses these values to reduce the total amount of switching in the circuit during the next clock cycle.

In [25][26], they present an algorithm to synthesize pre-computation logic for

the complete input-disabling architecture. The pre-computation logic is a function of all of the input variables. It is shown in Figure 2-10, the complete input-disabling architecture can reduce power dissipation for a larger class of sequential circuits.

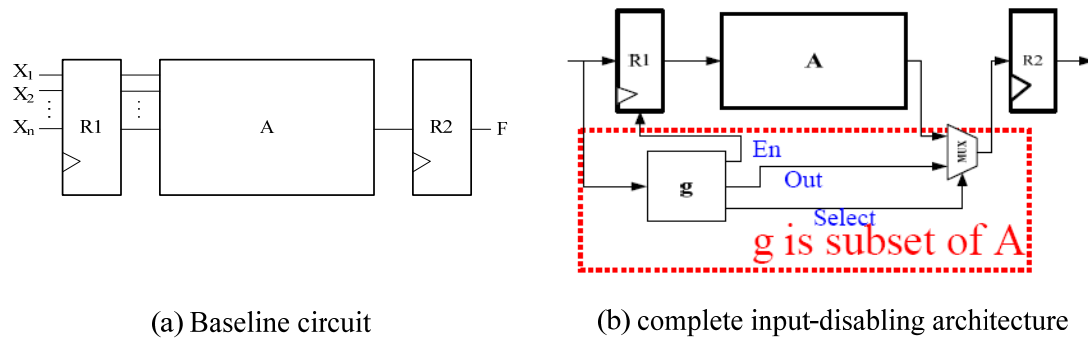
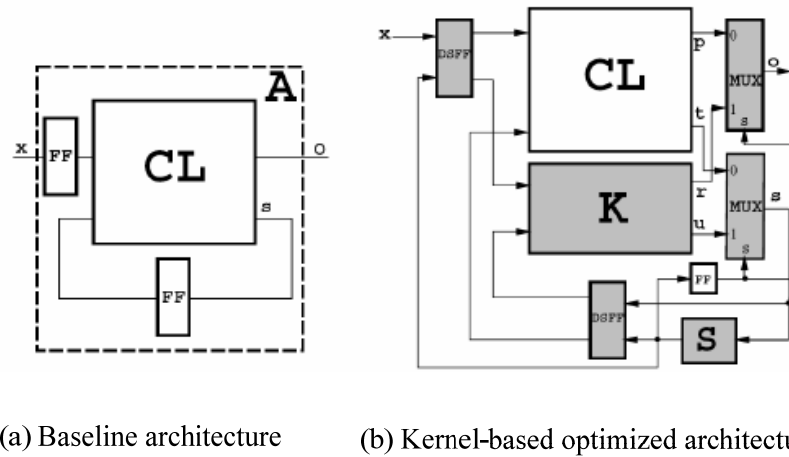


Figure 2-10 Pre-computation logic

- **Computation kernel**

It also duplicates a part of the original circuit. The sub-set logic is smaller and faster such that it dissipates less power. At the most time, the sub-set logic can accomplish the circuit operation, and the original circuit is turned off.

Figure 2-11 (a) shows an example with the standard topology. The paradigm for improving its quality with respect to a given cost function is based on the architecture shown in Figure 2-11 (b). The architecture consists of the combinational portion of the original circuit (block CL), the computational kernel (block K), the selector function (block S), the double state flip-flops (DSFF), and the output multiplexers (MUX).



(a) Baseline architecture (b) Kernel-based optimized architecture

Figure 2-11 Computational kernel [27]

In [27] that presents a power optimization technique by exploiting the concept of computational kernel of a sequential circuit, which is a highly simplified logic block that imitates the steady-state behavior of the original specification. This block is smaller, faster, and less power consuming than the circuit from which it is extracted and can replace the original network for a large fraction of the operation time.

In [28] that presents a low power adder for SIMD data path. By exploiting the difference length in the critical path for the types of operations (e.g., 4x8/2x16/1x32), energy-efficient SIMD adders can be developed. Indeed, 8-bit adders have smaller gates and energy consumption. Hence, 4x8-bit operations on an 8-bit ripple adder consume 1.8 times less compared 1x32-bit operation on a 32-bit adder. To alleviate the power dissipation, it combines four 8-bit energy optimized adders and one 32-bit adder to support SIMD.

2.4 Adaptive Approaches

This category is said that the supply voltage and switch activity/capacitance can be adjusted adaptively. It is also more flexible than the static approaches. Compared

with dynamic approaches, it can adapt to environmental conditions or data correlations. I'll introduce the techniques of energy reduction by reducing supply voltage and switch activity/capacitance respectively.

2.4.1 Supply Voltage

- **Adaptive voltage scaling (AVS)**

It is a one method of dynamic voltage scaling. It can adaptively scale the supply voltage by monitoring the actual silicon speed [23][29]. Therefore, worst case characterization is no longer required.

The actual performance is monitored using on-chip structures. The frequency of the ring oscillator is sampled using a counter as shown in Figure 2-12. The frequency count is then compared to the frequency required by the system and the difference is filtered using the system's filter. It has to be built in the ring oscillator to accommodate for all types of gates and all conditions. A better approach is to use a critical path replica as shown in Figure 2-12.

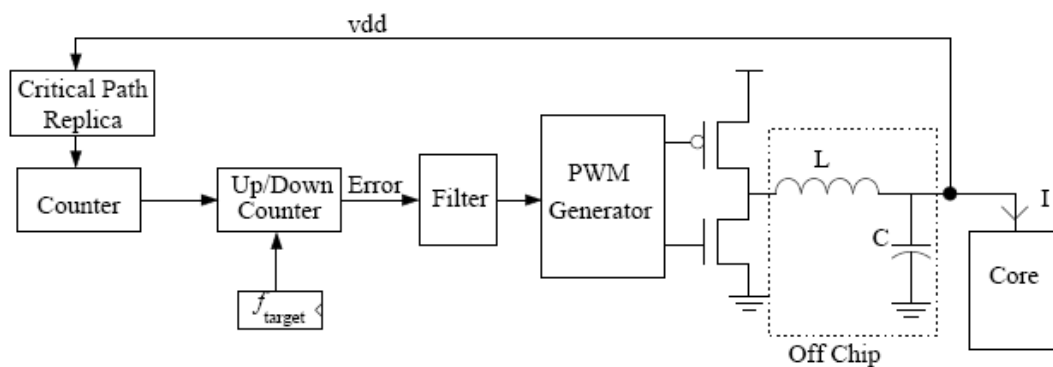


Figure 2-12 Architecture of the AVS system

2.4.2 Switching Activity and Capacitance

- **Bit swapping**

The most effective method to reduce the number of transitions in functional units is increasing the correlation of input data. The bit-swapping method is to change the input bit of functional unit according to the previous input bit status such that the number of signal transitions can be minimized [6].

Shown an example in Figure 2-13, the exclusive-OR gate is a selection logic that it manages the bit swapping. Previous data of in1 is 4'b0011 and in2 is 4'b1100, and the next data of in1 is 4'b0100 and in2 is 4'b1011. After bit swapping, the next data of in1 is swapped as 4'b0011 and in2 is swapped as 4'b1100.

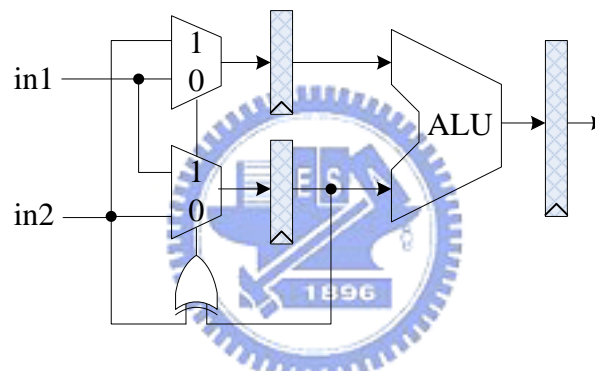


Figure 2-13 Example of bit swapping

- **Guarded evaluation**

Guarded evaluation is based on placing some guarded logic, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power managed. When the block must execute some useful computation in a clock cycle, the enable signal makes the latches transparent. Otherwise, the latches retain their previous states and block any transition within the logic block.

In [30], it proposes a technique which is called partially guarded computation. The technique disables a part of a circuit based on the dynamic range of input

operands. They divide a circuit into two parts – MSP and LSP – and allow only the LSP computation when the range of input operands is covered by the range of the LSP. Therefore, it can reduce unnecessary signal transitions.

- **Proposed energy-efficient design**

Circuit delay is strongly data dependent, and only exhibits its critical path delay for very specific data sequences [7][8][9]. Proposed design is exploiting data-dependent delay to reduce circuit energy. Shown in Figure 2-14 (a) is an example that it depicts a path delay distribution of original circuit. The x-axis represents the path delay, and the y-axis represents the number of patterns.

In this example, we assume that it is a normal distribution. Noted the distribution, delay time of most patterns is smaller than the critical path delay (clock period), and only few patterns can activate the critical path. We can attempt to optimize the common case for energy reduction based on the clock period, rather than to optimize the worst-case (critical paths) based on the clock period, shown in Figure 2-14 (b). Therefore path delay of some paths (critical paths) may be longer than the clock period, but the circuit energy can be reduced effectively. As long as we can tolerate these critical paths, we can gain the energy reduction.

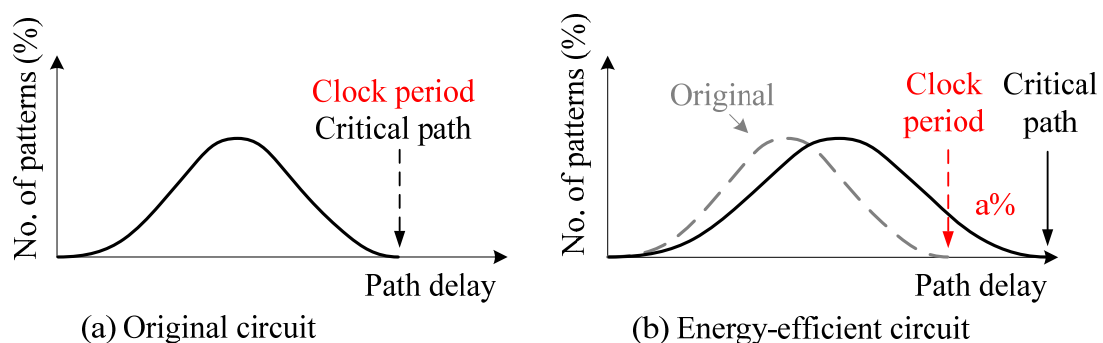


Figure 2-14 Path delay distribution

Shown in Figure 2-14 (b), there are a% of total input patterns that can not

accomplish a computation within a clock period and may cause to computing errors. In order to tolerate the errors, all patterns that will incur computing errors are operated two clock cycles (one-cycle latency penalty). Hence we generate a “detection logic” that is responsible for the error detection, and the circuit is augmented with the “detection logic”.

Shown in Figure 2-15, the input pattern of the detection logic is the same as the functional unit, and the output of detection logic is a 1-bit “wait” signal. If the “wait” signal is asserted, the input patterns would be latched one more cycle and output data is not available.

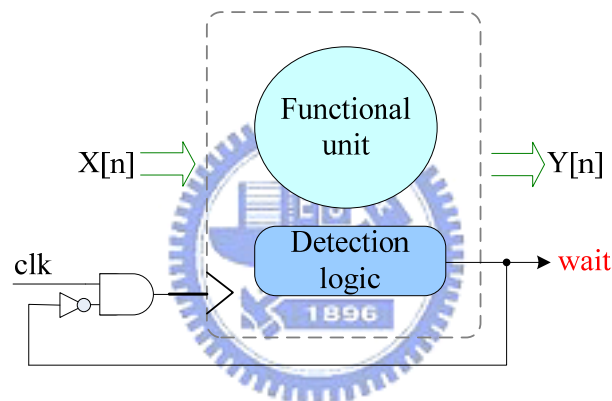


Figure 2-15 Conceptual circuit of proposed design

From this scenario, although the circuit energy can be reduced, the performance may be degraded also. In order to reduce the performance penalty, the detection logic needs to exactly detect the computation errors. We can also reduce the number of violating paths to reduce performance penalty, but that also influences the effect upon the energy reduction. It has to trade-off between energy and performance. This part is the main problem I want to solve.

3 PROPOSED ENERGY-EFFICIENT DESIGN



Energy consumption has become a critical issue in modern VLSI designs. For the circuit energy reduction, we propose a method that trades small performance penalty for large energy reduction. In this chapter, I will introduce our proposed energy-efficient design that it consists of the CMOS circuit delay, the template of variable latency design and proposed design flow.

3.1 Delay of CMOS Circuits

In the synchronous circuit design, traditional strategies for circuit optimization are based on worst case (critical path). For the given clock period, the critical paths of the circuit must be optimized to meet it, but that usually spends much energy effort to accomplish. The energy effort consists of gate size, structure and voltage.

In [7][8][9], we observe that the circuit delay is strongly data dependent, and only exhibits its critical path delay for very specific data sequences. Because, CMOS circuit delay is equal to the elapsed time of charging and discharging the circuit capacitances [31]. The computation time of each input pattern is based on the original status of the circuit capacitance. The same input patterns with different status of circuit capacitances will activate different paths such that the computation times are different.

Hence, estimating the circuit delay or path delay requires a two-pattern sequence — the first pattern initializes the circuit while the second pattern causes and propagates the desired transition [32][33].

To observe the delay of CMOS circuits, we synthesized a 8-bit unsigned carry-save-array multiplier using the UMC 90nm CMOS cell library. After the gate level synthesis, we used the 10,000 random pattern sequences for gate-level simulation. Figure 3-1 shows the path delay distribution of the 8-bit carry-save-array multiplier. The x-axis represents the delay time of the data computation (path delay), and the y-axis represents the number of patterns.

The green line represents the path delay distribution of the multiplier. The clock period is 1.6ns, so the critical path of the multiplier can not be larger than the clock period. The path delay distribution is similar to the normal distribution, and the probability of sensitizing the critical paths is very low.

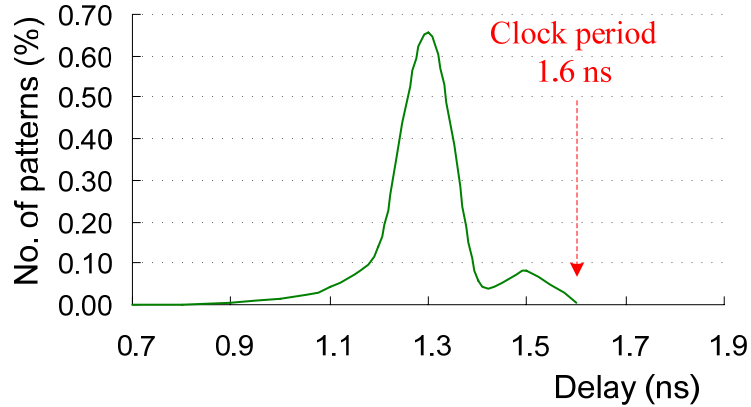


Figure 3-1 Path delay distribution (8-bit multiplier)

For the 1.6ns clock period, the conventional design method is directly synthesizing the circuit with 1.6ns timing constraint. From the path delay distribution we found that the delay time of most patterns are smaller than 1.4ns even. From this scenario, the circuit energy can be optimized for common case, rather than the few critical cases. In other words, we can relax the synthesis timing constraint for energy reduction and tolerate the few critical cases.

Then we observe the relationship between the circuit energy and synthesis timing constraint. We use the UMC 90nm CMOS cell library and 10,000 random patterns to estimate the energy consumption (average energy consumption per operation). Figure 3-2 shows the energy curve of 8-bit carry-save-array multiplier with different synthesis timing constraints. The x-axis represents the synthesis timing constraint (circuit delay), and the y-axis represents the energy per multiplication.

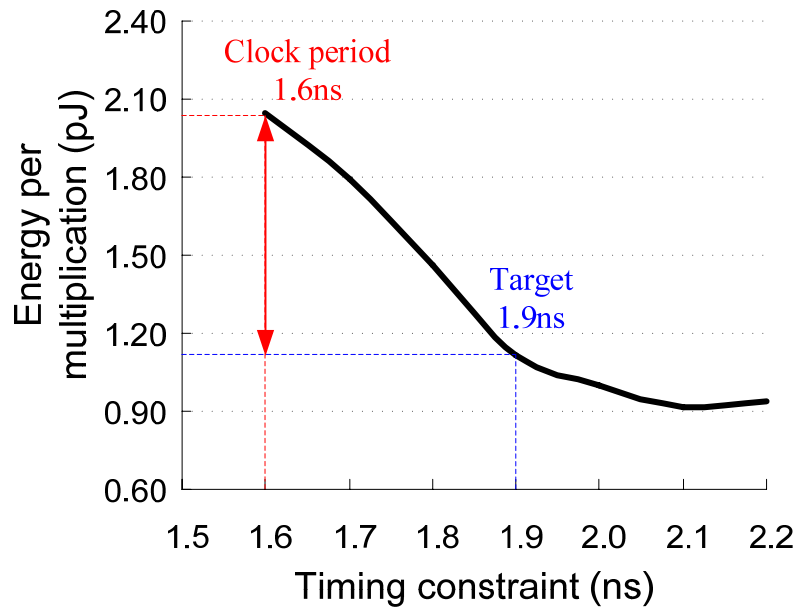


Figure 3-2 Energy curve of 8-bit multiplier

Tightening the timing constraint of the multiplier will induce the increasing of the energy per multiplication. Especially when the timing constraint approaches the peak value (1.6ns), the energy consumption increases drastically. Even if we synthesize the circuit with power optimization constraint, the circuit energy decreases also as the synthesis timing constraint relaxes.

Optimizing the circuit delay needs to spend large energy effort. From the path delay distribution, we found that the energy effort is spent on the few circuit critical paths. The energy effort consists of optimizing the circuit structure and upping the gate sizes, and it makes the circuit delay (critical path delay) to be reduced. Optimizing the circuit structure or upping the gate sizes usually causes the circuit capacitance to be increased, and therefore the circuit energy is increased.

Multimedia systems are desired not only for low-energy consumption but also for high speed (high performance). Although relaxing timing constraint is an effective method for energy reduction. In Figure 3-2, the timing constraint is relaxed from 1.6ns to 1.9ns will lead the energy consumption to be reduced about 45%, but

it indicates that the clock frequency (performance) is degraded directly.

Hence, exploiting the data-dependent delay of circuit can not only avoid clock frequency degradation but also gain the energy reduction. For instance, from the above multiplier, if the operating clock period is 1.6ns, the synthesis timing constraint can be relaxed to 1.9ns, and therefore the energy can be reduced about 45%. Then we observe the path delay distribution of the multiplier with 1.9ns critical path, it is shown in Figure 3-3.

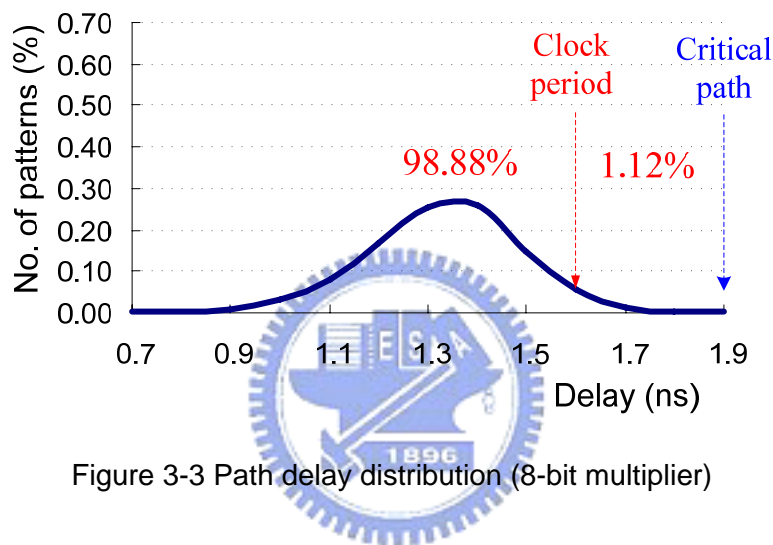


Figure 3-3 Path delay distribution (8-bit multiplier)

The delay time of most patterns (98.88% of pattern) is less than 1.6ns (clock period), and only 1.12% of pattern that delay time is greater than 1.6ns. The delay time (computation time) of few input patterns will exceed 1.6ns, and these patterns may incur computing errors. The possible computing errors can be detected and it can be corrected by two-cycle operation. This implies that a one-cycle latency penalty. The detection and correction will be discussed in detail later.

From Figure 3-2 and Figure 3-3, only 1.12% of pattern that the delay time is greater than 1.6ns, the probability of spending one-cycle latency penalty is 1.12%, so the performance penalty is very light and negligible. If few errors (one-cycle latency penalty) can be tolerated by the multiplier design, the energy per

multiplication can be reduced about 45%.

In order to detect the computing errors and accommodate the one-cycle latency penalty, we proposed a variable latency design that it can be simply integrated into other systems.

3.2 Variable Latency Design

3.2.1 Template of Variable Latency Design

The proposed variable latency design can accommodate the additional one-cycle latency penalty. In other words, the latency of the functional unit can adapt to the input patterns, most patterns only need one-cycle latency and few patterns need two-cycle latency.

The template of variable latency design is shown in Figure 3-4. We assume the functional unit has the input and out registers, and it is augmented with the detection logic. In normal situation, the functional unit has only one-cycle latency, and the detection logic does not influence the functional unit. When the computation time of input patterns exceeds a clock period, the input patterns need to be operated two clock cycles to avoid the computing error. The detection logic is responsible to detect the input pattern that the computation time exceeds a clock period and control the latency of the functional unit.



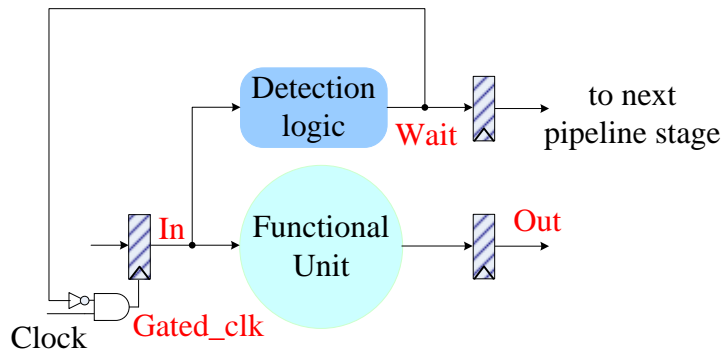


Figure 3-4 Template of variable latency design

Detection logic shown in Figure 3-4 is responsible to detect the input patterns that will result in computing errors. The input pattern of the detection logic is the same as the input pattern of the functional unit. If the detection logic detects a computing error will occur, the output signal “wait” will be asserted. The wait signal will propagate to the next stage and make the output data of the functional unit invalid. At the same time, the wait signal will control the flip-flops of the previous stage to latch all patterns one more cycle. The behavior is like the stall cycle in the processor. In other words, if the wait signal is asserted, the functional unit needs a stall cycle.

The detection logic consists of a fault function and additional flip flops, and it is shown in Figure 3-5. The additional flip flops are used to latch the previous input pattern, because the circuit delay is data-dependent [32][33]. The propagation time of input patterns is based on the original signal status of the each gate in the functional unit [31].

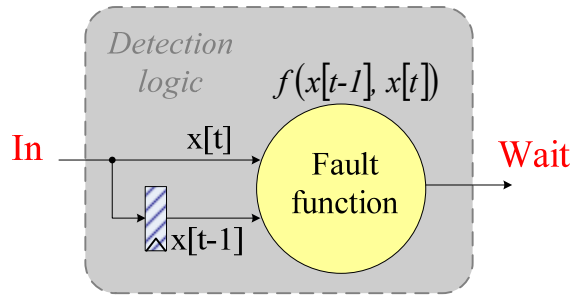


Figure 3-5 Detection logic

The previous input pattern is latched by the additional flip flops, and hence the original signal status of each gate in the functional unit can be estimated. Hence, the signal propagation time of the functional unit can be estimated accurately. If the maximum delay at the certain input bit is guaranteed to meet the clock period, the additional flip flop of the certain bit does not need.

The fault function is a function of all of the input variables, $f(x[t-1], x[t])$. If the input pattern satisfies the function, the input pattern spends a one-cycle latency penalty (two-cycle operation). I will introduce about how to derive the fault function in detail later.

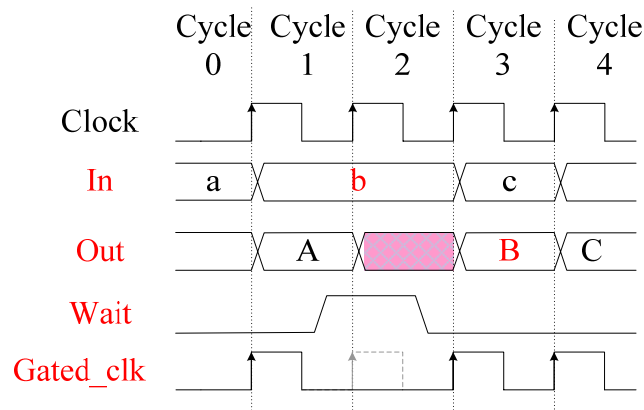


Figure 3-6 timing diagram of the variable latency design

Timing diagram in Figure 3-6 shows an example for the variable latency design. Each signal in Figure 3-6 corresponds to it in Figure 3-4. We show an example that

the input pattern “b” of functional unit in cycle 1 is a violating data (the computation time of input data exceeds a clock period). In this case, the detection logic detects that a computing error will occur, hence the wait signal is asserted at cycle 1. That represents the output data is invalid, input data will be latched one more cycle and the clock will be gated.

At cycle 2, the input pattern “b” needs to be latched one more cycle. In other words, it incurs one-cycle latency penalty. The output data “B” that corresponds to the input data “b” can not be available at cycle 2. The clock is gated at cycle 2, therefore the input data is still “b”, and output data at cycle 2 is invalid.

At cycle 3, the correct output data “B” is available, and the circuit is restored to normal in the subsequent cycle.

Next section, the detection logic generation will be introduced.

3.2.2 Detection Logic



The detection logic generation is shown in Figure 3-7. The input file contains a netlist file of functional unit and the given clock period. After receiving the input files, three steps need to be executed in sequence. Then the detection logic can be generated.

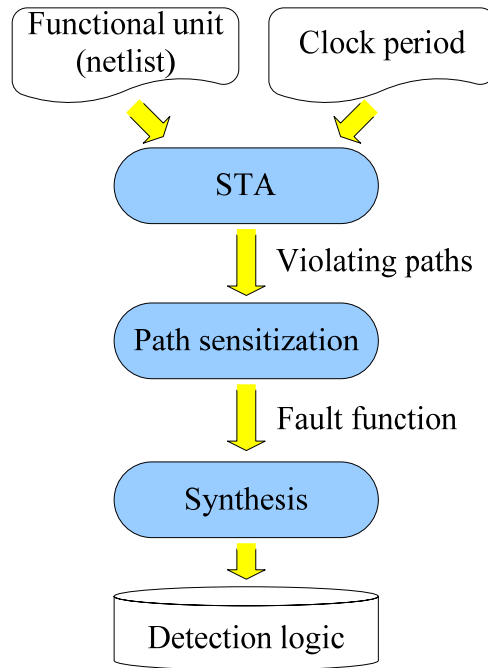


Figure 3-7 Detection logic generation

First step is to use the static timing analysis (STA) analyzing the path delay of functional unit. Static timing analysis (STA) is a method of validating the timing performance of a design by checking all possible paths for timing violations [34]. It checks for violations of timing constraints inside the design and at the input/output interface.

In our experiments, we use the Synopsys Prime-Time tool to perform static timing analysis (STA), and the timing constraints is the given clock period. It can report all paths that path delay exceeds one clock period, and these paths are called “violating paths”. Then, we can perform path sensitization to find all patterns that would sensitize the violating paths.

Second step is to analyze the violating paths based on the path sensitization criterion. After the path sensitization algorithm is accomplished, all input patterns that the propagation time from input to output is larger than one clock period can be found. Such input patterns we called “violating patterns”. After all violating patterns

are found, the “fault function” can be derived. The fault function is a function of all of the input variables, $f(x[t-1], x[t])$, and it contains all violating patterns. All violating patterns must be included in the ON-set of the fault function. If the fault function is satisfied by the input patterns, the input patterns need to be operated two cycles. We will discuss the path sensitization and fault function in detail later.

The final step is to synthesize the fault function, and therefore the detection logic can be obtained. If the complexity of fault function is very great, some don't care patterns can be added to the fault function such that the complexity of fault function can be simplified. In our experiment, we directly synthesize the fault function (PLA format) by Synopsys Design Compiler. The synthesis timing constraint is one clock period, and we need to guarantee the critical path of detection logic is smaller than one clock period.

Before introducing the path sensitization and fault function, some definition and notations have to be defined. This will help the explanation of the method of deriving the fault function.



● Definition and notations

A combinational circuit is bounded by primary inputs and primary outputs and it is composed of simple gates (i.e., AND, NAND, OR, NOR, and NOT gates). The delay of gate G is denoted by $d(G)$.

Definition 1 (path)

A path $P = (I, G_1, G_2, \dots, G_m, O)$ in a combinational circuit is an sequence of primary input (I), gates (G_i), and primary output (O). The primary input (I) connects to gate G_1 , output of gate G_i connects to input of gate G_{i+1} , where gate G_i , $1 \leq i \leq m - 1$, and output of gate G_m connects to primary output (O). The delay of path P is the sum of the delays of all the gates, and is denoted by $d(P)$.

Definition 2 (on-input and side-input)

Let $P = (I, G_1, G_2, \dots, G_m, O)$ be a path. Primary input I is an on-input of gate G_1 , and output of gate G_i that connects to gate G_{i+1} is an on-input of gate G_{i+1} , where gate G_i , $1 \leq i \leq m - 1$. Other inputs of gate G_i are defined as side-inputs, where gate G_i , $1 \leq i \leq m$.

Definition 3 (controlling value)

A logic value is the controlling value to a gate if and only if the logic value at an input to the gate independently determines the value at the output of the gate. The controlling value to gate G is denoted by $c(G)$. For examples, $c(G)$ is logic 0 if G is an AND gate or a NAND gate, and $c(G)$ is logic 1 if G is an OR gate or a NOR gate.

Definition 4 (non-controlling value)

The non-controlling value to gate G , denoted by $nc(G)$, it is the complementary value of $c(G)$. For examples, $nc(G)$ is logic 1 if G is an AND gate or a NAND gate, and $nc(G)$ is logic 0 if G is an OR gate or a NOR gate.

For a NOT gate which has single input, both logic 0 and logic 1 are considered to be its controlling values.

Definition 5 (input vector)

An input vector v is a vector of logic values at all the primary inputs. Each logic value is either logic 0 or logic 1.

Definition 6 (stable value and stable time)

Let v be an input vector applied to the circuit under analysis. The logic values stabilized at the end of the output of gate G are called the stable values at G under v . When the end of the output of G becomes stable, the time is called the stable time at G under v .

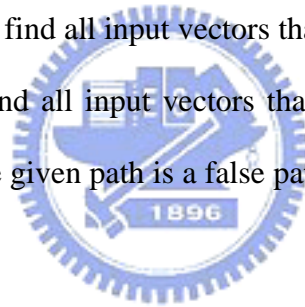
Definition 7 (sensitizable)

A path is sensitizable if there is at least one input vector to activate it. A path is false if there is no input vector to activate it. The critical paths are the longest sensitizable paths.

● **Path sensitization**

Delay of the circuit is equal to the delay of the longest sensitizable path. A path is sensitizable if it can be activated by at least one input vector. Therefore, determining the sensitizability of a path is equivalent to determine the existence of input vectors which activate the path [35]. Thus it will be very helpful to develop a criterion which is capable of computing the set of input vectors that activate the path. Now, we will focus on how to find all input vectors that activate a given path.

The exact criterion can find all input vectors that can activate a given path. If that finds no input vectors, the given path is a false path. I briefly introduce the exact criterion.



Exact Path Sensitization Criterion

The path P is a exact sensitizable path if there is at least one primary input such that for each on-input of path P and for each side-input of path P hold either one of the following conditions (shown in Figure 3-8):

- 1) *The on-input is the earliest controlling input, otherwise all side-inputs are non-controlling inputs*

- 2) *The on-input is the latest non-controlling input, given all its side-inputs is also non-controlling inputs*

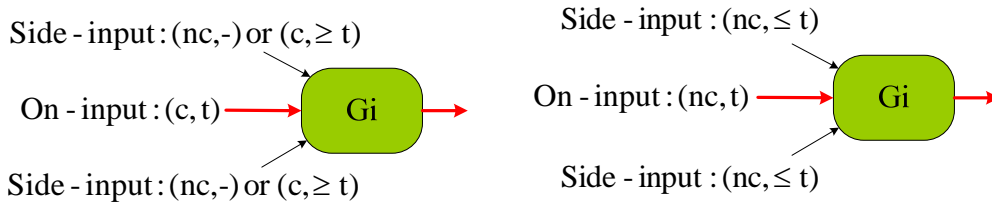


Figure 3-8 Exact path sensitization criterion

Figure 3-9 shows an example of a delay-dependent false path in carry-look-ahead logic. Let the delay of each gate be 1 time unit. The highlighted path $P=(x, C, D, E, F, G, O)$ is false for a rising transition at input x . From the exact path sensitization criterion, we found that the side-input of gate “F” is the earliest controlling input such that the shorter direct path $P_s=(x, F, G, O)$ determines the longest true path. Therefore the path P can not be sensitized.

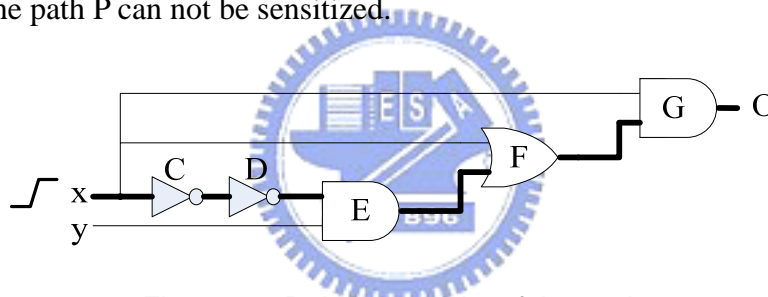


Figure 3-9 Delay-dependent false path

The exact path sensitization criterion is the general path sensitization criterion, regardless of the longer path or the shorter path. The violating paths are the most critical paths, so we can use some criteria that they are proposed only for dealing with the critical paths. These criteria are only applied to the critical paths, and they are less restricted than the exact criterion.

Our purpose is to analyze the violating paths based on the viable path sensitization criterion, and violating paths are the most critical paths in the circuit. The viable criterion and the loose criterion achieve the same estimation of the critical paths [35]. The loose criterion and the exact criterion achieve the same

estimation of the critical paths [35]. So, the viable criterion and the exact criterion can also achieve the same estimation of the critical paths. The viable criterion is easier to implement. Hence we use this criterion to analyze the critical path, and I briefly introduce the viable criterion.

Viable Path Sensitization Criterion

According to McGeer and Brayton, a path P is viable [36] if there is at least one primary input such that for each on-input of path P and for each side-input of path P hold either one of the following conditions (shown in Figure 3-10) :

- 1) *All side-inputs are non-controlling inputs*
- 2) *If any side-input is controlling input, the stable time of side-input must be later than on-input*

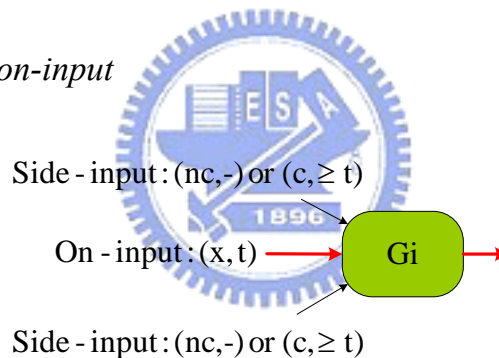


Figure 3-10 Viable path sensitization criterion

All the input patterns that activate the given critical paths can be found based on the viable path sensitization criterion, and the stable time of side-input is given by the static timing analysis. We show an example in Figure 3-11, the critical path $P=(A, G1, G2, G3, O)$ is given by the static timing analysis, and it has a rising transition. Use the viable path sensitization criterion to find all patterns that can sensitize the given critical path.

Based on the viable path sensitization criterion, we set non-controlling values on the side-input of all gates on the critical path. Because the path is critical path, the

stable time of all side-inputs is earlier than on-input. In other words, we needn't consider the controlling values on the side-inputs of all gates. Therefore, the non-controlling value of AND gate and NAND gate is logic 1, the non-controlling value of OR gate is logic 0. The values are backtracked to the primary inputs, and the input patterns $(A, B, C, D) = (x, 1, 0, 1)$ are obtained, where "x" represents don't care.

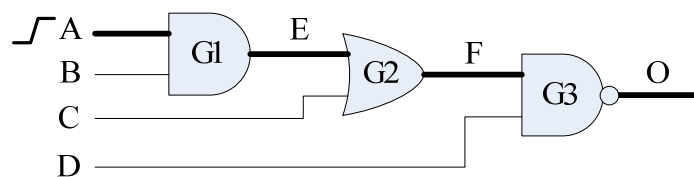


Figure 3-11 An example of given critical path

The above path sensitization criteria use pattern-independent timing, and they only consider a single pattern to sensitize the path. The results are usually very pessimistic [34]. Indeed circuit delay is pattern dependent, and it is caused by the signal transition and propagation [32][33]. Therefore, if we want to find the violating patterns exactly, we need to use pattern-dependent timing to analyze the path [37].

In other words, we need to consider the previous patterns. Thus we need to consider two-pattern sequence, where the first pattern initializes the circuit and the second pattern causes transition. Therefore the viable path sensitization criterion can be modified to consider the on-input transition (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$) [38][39]. We have a brief explanation below.

Modified Viable Path Sensitization Criterion

A path P is sensitizable if there is at least one input sequence (two-pattern sequence) such that for each on-input of path P and for each side-input of path P

hold either one of the following conditions (also shown in Figure 3-12) :

- 1) *All side-inputs are non-controlling inputs, and on-input (primary input) has an event (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$).*
- 2) *If any side-input is controlling input, the stable time of side-input must be later than on-input, and on-input (primary input) has an event (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$).*

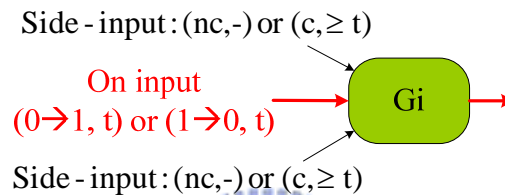


Figure 3-12 Modified viable path sensitization criterion

We analyze the example shown in Figure 3-11 again based on the modified viable path sensitization criterion and the critical path that is reported by static timing analysis (Prime Time). Because the transition on the primary input of the critical path is considered, the given critical path is sensitized by the two-pattern sequence. The non-controlling values are backtracked to the primary inputs and the transition on the primary inputs is considered, therefore all the pattern sequences $(ABCD[t-1], ABCD[t]) = (0xxx, 1101)$ that can sensitize the given critical path are obtained, where “x” represents don’t care.

The result is more precise than it that is analyzed by the viable criterion, but the complexity of the result is greater than it that is analyzed by the viable criterion.

- **Fault function**

Fault function $f(x[n-1], x[n])$ contains all and only those input patterns

(violating patterns) that the propagation time from the inputs to the outputs is longer than one clock period. All violating patterns can be found by analyzing the violating paths based on the modified viable path sensitization criterion.

We use a flowchart shown in Figure 3-13 to explain how we derive the fault function. The input file “violating paths” is reported by performing the static timing analysis. The timing information about stable time of each signal is also based on the result of performing the static timing analysis.

Therefore, we only analyze the logic value of all gates, instead of stable time of all signals of all gates. The stable time of all signals of all gates on the most critical path is the latest. When all of the violating paths are obtained, three recursive steps have to be taken iteratively.

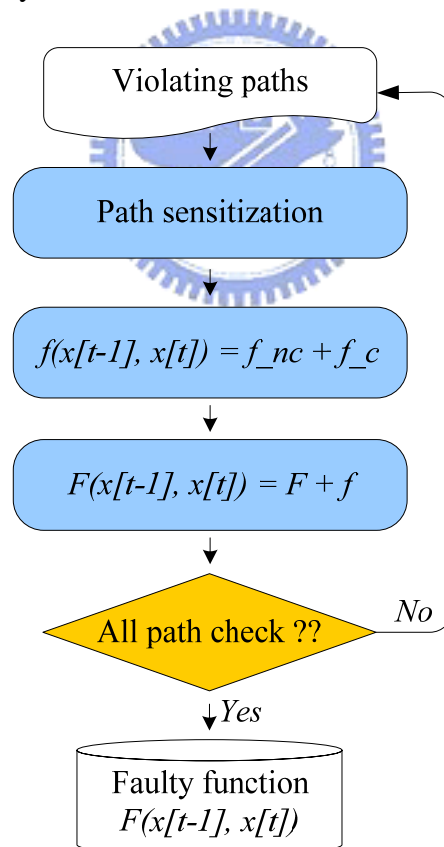


Figure 3-13 Flowchart of fault function

First step is to analyze the violating paths based on the modified viable path

sensitization criterion. The violating paths are in path-delay order, the longest path is analyzed first and only one path is analyzed at a time. After one path is analyzed, two sub-functions are obtained. One sub-function “ $f_{nc}(x[t-1], x[t])$ ” is derived based on all side-inputs of the path are non-controlling values, and another sub-function “ $f_c(x[t-1], x[t])$ ” is based on any side-input of the path has a controlling value.

Second step is to combine two sub-functions to form a complete function “ $f(x[t-1], x[t])$ ” of the path. The pattern sequence satisfies the function is represented that the pattern sequence sensitizes the path. Finally, we accumulate the function of each path until all violating paths have been analyzed, and the fault function is obtained “ $F(x[t-1], x[t])$ ”.

● **Example**

In Figure 3-14, we show an example with two violating paths. The longest path P1 = (A, G1, G2, O) has a rising transition at input A. Another path P2 = (C, G3, G2, O) has a falling transition at input C.

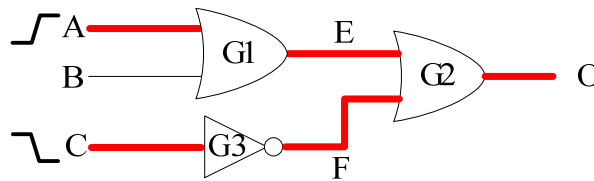


Figure 3-14 Two violating paths

For the first path P1 = (A, G1, G2, O)

Step 1 :

$$f_{nc}(ABC[t-1], ABC[t]) = A \uparrow \cdot \bar{B} \cdot \bar{F} = A \uparrow \cdot \bar{B} \cdot C = (0xx, 101)$$

$$f_c(ABC[t-1], ABC[t]) = \varphi$$

Step 2 :

$$f(ABC[t-1], ABC[t]) = f_{nc} + f_c = (0xx, 101)$$

Step 3 :

$$F(ABC[t-1], ABC[t]) = F + f = (0xx, 101)$$

For the second path P2 = (C, G3, G2, O)

Step 1 :

$$f_{nc}(ABC[t-1], ABC[t]) = C \downarrow \cdot \bar{E} = C \downarrow \cdot \bar{A} \cdot \bar{B} = (xx1, 000)$$

$$f_c(ABC[t-1], ABC[t]) = C \downarrow \cdot E = C \downarrow \cdot A \uparrow \cdot \bar{B} = (0x1, 100)$$

Step 2 :

$$f(ABC[t-1], ABC[t]) = f_{nc} + f_c = (xx1, 000) + (0x1, 100)$$

Step 3 :

$$F(ABC[t-1], ABC[t]) = F + f = (0xx, 101) + (xx1, 000) + (0x1, 100)$$

We obtain the final result after the two paths are analyzed.

3.3 Design Flow

The functional units are the main blocks for the multimedia applications and portable devices. High speed and low energy consumption are both the requirements for the embedded systems. In this section, I will introduce a design flow that can help us to systematically design the most energy-efficient functional units. The energy-efficient functional unit represents that it is the one that consumed the least energy among all configurations that deliver the same performance [40], and the energy-delay product is usually used to be a metric. Smaller energy-delay values imply a lower energy solution at the same level of performance — a more energy-efficient design. The proposed energy-efficient functional units can be operated at the desired clock period, and it trades the minimum performance penalty for the maximum energy reduction.

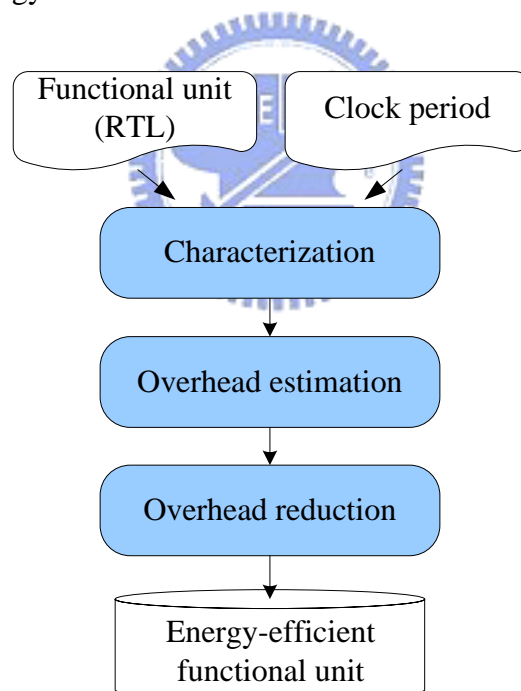


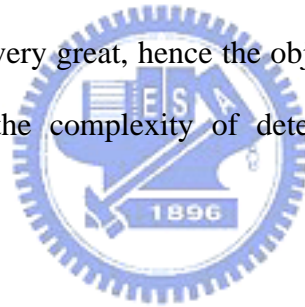
Figure 3-15 Design flow of energy-efficient functional unit

Figure 3-15 illustrates the design flow for energy-efficient design, and it consists of three steps that need to be executed in sequence. The flow is based on the cell-based design flow, and it can easily obtain the most energy-efficient result.

First step is to characterize the functional unit to know the relationship between circuit energy per operation, error rate, and timing constraints. The error rates of the functional unit with different timing constraints are estimated at the given clock period. The energy or error rate can be estimated by using the random patterns or a real application. In our experiments, we use 10,000 random patterns to characterize the functional units.

Second step is overhead estimation. The overhead is the energy consumption of detection logic. The most energy-efficient design is consuming the minimum energy that includes both functional unit energy and detection logic energy, and the performance is negligible.

Final step is overhead reduction. The area and energy of detection logic or the performance penalty may be very great, hence the objective of this step is to reduce these overhead. It reduces the complexity of detection logic and performance penalty.



3.3.1 Characterization

Characterization is the first step of our proposed design flow. When the clock period is given, the functional unit is characterized to know the relationship between circuit energy, error rate and timing constraints. The objective of the characterization is to know how many space that the circuit energy can be reduced, and the corresponding performance penalty.

The minimum value of the timing constraint is the value of clock period, and the maximum value of the timing constraint is the value of the double clock period. The timing constraint must be smaller than the double clock period, because the operation with computation error only has a one-cycle latency penalty (two-cycle

operation). If the timing constraint is larger than the double clock period, the operation with computation error may need a two-cycle latency penalty (three-cycle operation) to correct it. In our variable latency design, we only can accommodate a one-cycle latency penalty.

Multiplier is one of the most energy-hungry functional units in datapath of DSP. Here we will use a multiplier as an example to show the characterization results. The structure of the multiplier is 8-bit unsigned carry-save-array multiplier. The characterization in this example is cell based using UMC 90nm CMOS cell library with Synopsys Design Compiler (Version V2007-03) as synthesizer. The CAD tools used to measure power and error rate are Synopsys PrimePower (Version V2006.06) and Cadence Verilog-XL respectively.

The peak operating frequency of the multiplier is 1.6ns, hence the 1.6ns is assumed as the given clock period. Figure 3-16 shows the results of the characterization. The x-axis represents the timing constraint, the left y-axis represents the energy per multiplication, and right y-axis represents the error rate. The blue line represents the circuit energy that the circuits are synthesized with different timing constraints, and the green line represents the error rate that the circuits are operated at the 1.6ns clock period with different timing constraints.

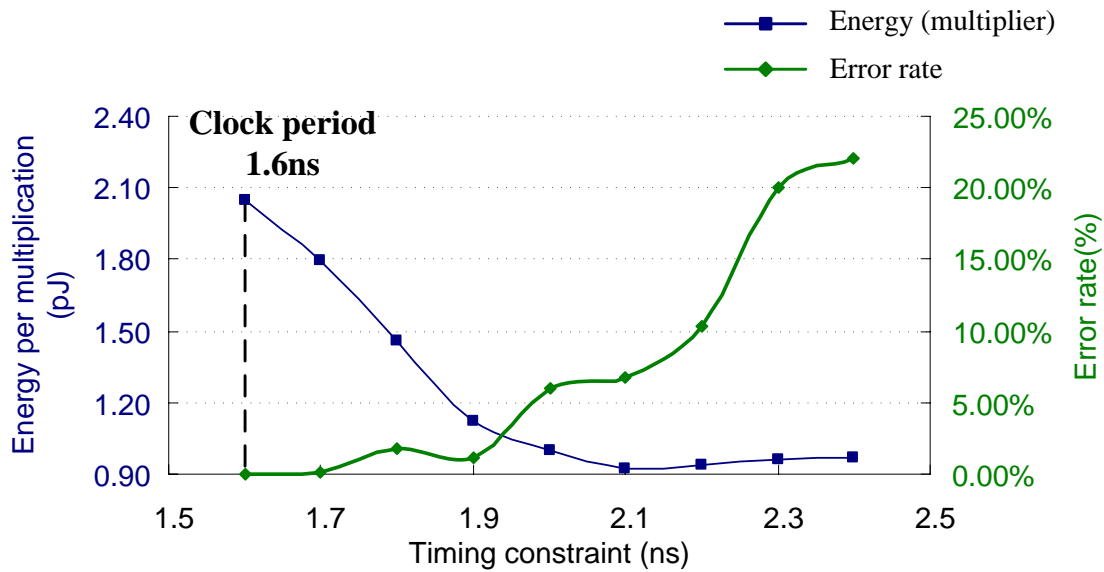


Figure 3-16 Characterization results

The energy represents average energy consumption per multiplication, and it is estimated at gate-level. The energy of the multiplier is decreased rapidly as the timing constraint is relaxed from 1.6ns.

The error rate represents the probability of the multiplier can't complete the operation within the given clock period. It is estimated using gate-level simulation with 10,000 random pattern sequences operated at the 1.6ns clock period. For instance, the circuit with 2.1ns timing constraint would complete 93.28% of all operations without errors, and it would save about 55% energy consumption.

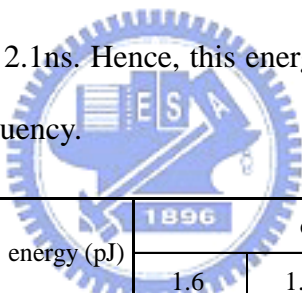
The range of timing constraints is from 1.6ns to 2.4ns, because the energy per multiplication is almost the same when the timing constraint is larger than 2.1ns but the error rate keeps increasing.

The concept of our proposed energy-efficient design is trading light performance penalty for large energy reduction. Hence, from the energy curve shown in Figure 3-16, we are only interested to the multipliers that with timing constraint from 1.6ns to 2.1ns, because the energy per multiplication is almost the same when the timing

constraint is larger than 2.1ns.

From the error rate curve shown in Figure 3-16, we are only interested to the multipliers that with timing constraint from 1.6ns to 1.9ns, because the error rate in this region is very small. So, we can only estimate the energy overhead (detection logic) in this region.

Before we perform the step 2 — overhead estimation, we show the complete characterization results in Table 3-1. The column 1 shows the timing constraints (critical path delay) of the multiplier. The area and energy of the multiplier are shown in column 2 and column 3 respectively. The column 4-9 in Table 3-1 represent the error rates of multiplier operating at different clock periods. The clock period is only shown from 1.6ns to 2.1ns, because the energy can't be saved when the clock period is larger than 2.1ns. Hence, this energy reduction technique is only suitable for the high clock frequency.



timing constraint (ns)	area (um ²)	energy (pJ)	clock period (ns) & error rate (%)					
			1.6	1.7	1.8	1.9	2.0	2.1
1.6	3,809	2.0464	0.00%					
1.7	3,447	1.7935	0.14%	0.00%				
1.8	2,982	1.4582	1.81%	0.06%	0.00%			
1.9	2,484	1.1191	1.12%	0.07%	0.00%	0.00%		
2.0	2,319	0.9984	6.03%	0.96%	0.04%	0.00%	0.00%	
2.1	2,193	0.9192	6.72%	1.08%	0.10%	0.02%	0.00%	0.00%
2.2	2,183	0.9392	10.38%	2.39%	0.30%	0.04%	0.00%	0.00%
2.3	2,161	0.9644	20.02%	7.99%	1.78%	0.12%	0.01%	0.00%
2.4	2,161	0.9696	22.10%	9.04%	2.37%	0.31%	0.06%	0.01%

Table 3-1 Characterize multiplier for different clock periods

3.3.2 Overhead Estimation

Overhead estimation is the second step of our proposed design flow. The overhead is the energy consumption of the detection logic. After the circuit characterization, we generate the detection logic and estimate its energy consumption (energy per operation). The energy consumption of detection logics is estimated at gate level and operated at the given clock period.

For simplicity, we reduce the cell library space. The space of UMC 90nm CMOS cell library is restricted to 2-input gates except XOR gate. If all gates are 2-input gates in the netlist file, we can easily to program the detection logic generation and analyze the violating paths of functional units.

Hence, we re-characterize the multiplier using the sub set of cell library. The characterization results are shown in Figure 3-17.

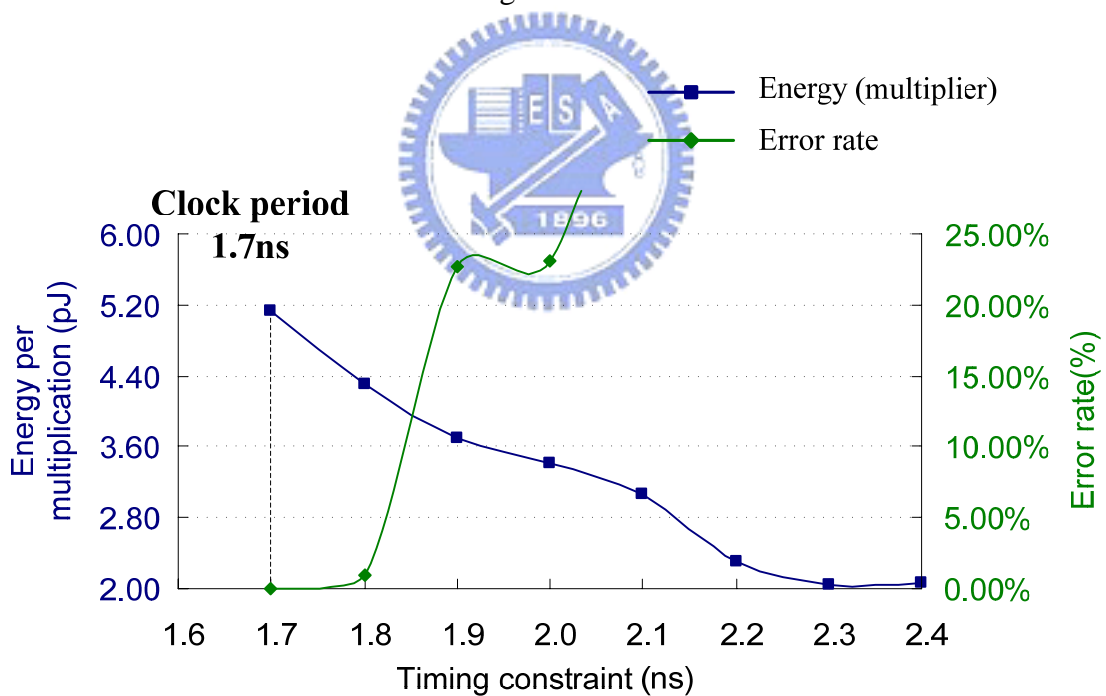


Figure 3-17 Characterization results (sub-set cell library)

If we assume the clock period is 1.7ns, and the timing constraint of the multiplier is from 1.7ns to 2.4ns. Compared with using full cell library (Figure 3-16), the energy by using the reduced cell library is larger (Figure 3-17). Although restricting

the cell library would influence the circuit energy, the increase trend of energy is also very drastic. It still exist a space for energy reduction also. For instance, the circuit energy can be reduced about 60%, when the timing constraint is relaxed from 1.7ns to 2.3ns. The error rate is estimated with 1.7ns clock period and it is very great when the timing constraint is larger than 1.8ns.

Next, we generate the detection logic and estimate the energy consumption. The result is shown in Figure 3-18. The pink line represents the total energy consumption that consists of multiplier and detection logic. The difference between the pink line and blue line represents the energy consumption of detection logic.

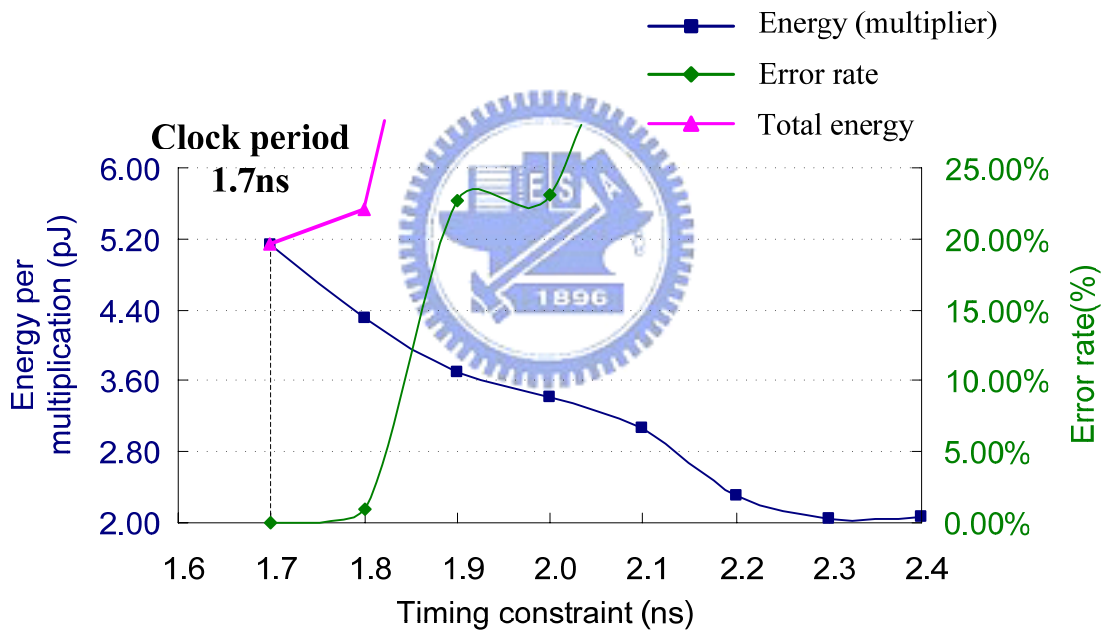


Figure 3-18 Overhead estimation (clock period is 1.7ns)

When the timing constraint is larger than 1.8ns the error rate is very great such that the performance penalty and complexity of detection logic are both very great also. Because we only focus on light performance penalty, the error rate in Figure 3-18 needs to be reduced. The final step of our proposed design flow is overhead reduction. I will introduce it in detail later.

If the clock period is 2ns, we re-characterize the error rate and re-estimate the overhead. The result is shown in Figure 3-19. From it we can find the error rate is very small and the total energy (multiplier + detection logic) is similar to a convex function. Hence, the structure with minimum energy in this convex curve is the most energy-efficient. The multiplier with 2.3ns timing constraint consumes 2.83pJ (multiplier + detection logic), and it is operated at 2ns clock period only with 0.83% performance penalty. The energy-delay value is the minimum value with the 2.3ns timing constraint, and it can be called the most energy-efficient design.

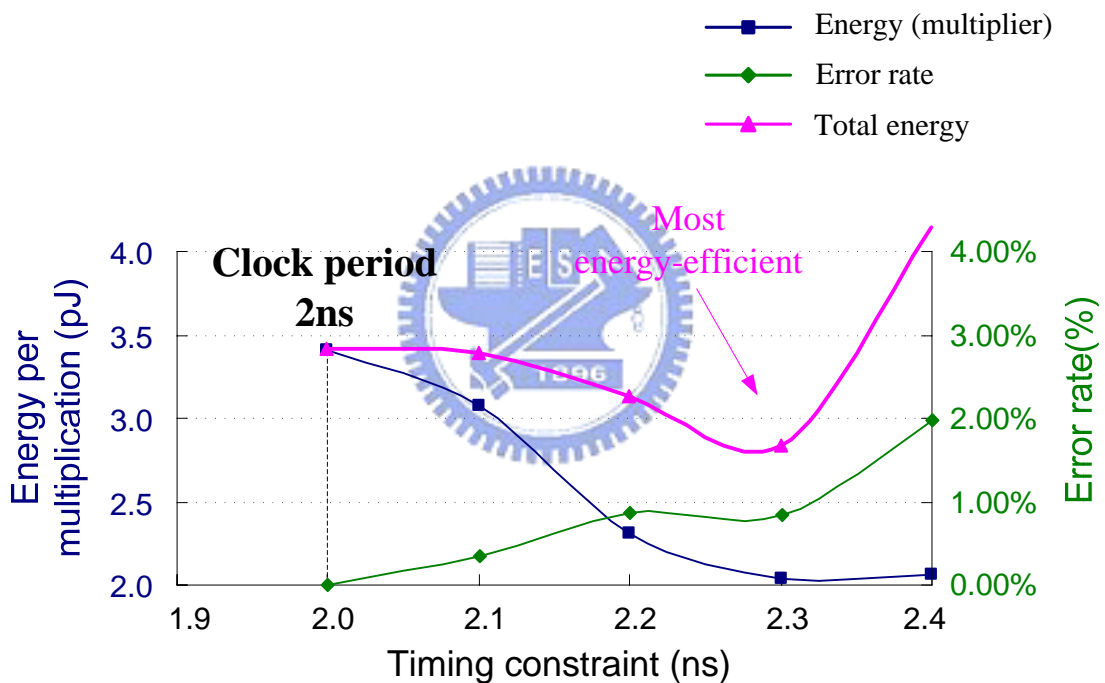


Figure 3-19 Overhead estimation (clock period is 2ns)

3.3.3 Overhead Reduction

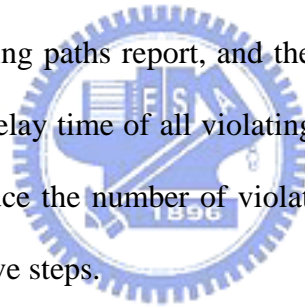
Overhead reduction is the final step of our proposed design flow. The objective of this step is to reduce performance penalty and energy overhead of the detection logic.

Because the increase of error rate will cause the increase of performance penalty

and the complexity of detection logic, reducing the error rate is an effective method to achieve overhead reduction. In order to reduce the error rate, we can reduce the number of violating paths such that the probability of sensitizing the violating paths can be reduced.

The number of violating paths can be reduced by upping the gate size of the violating paths such that the path delay of some violating paths can meet clock period [21][41], but it may increase the area and energy of the original functional unit. Here, we re-synthesize the functional unit to adjust the number of violating paths, and estimate the energy consumption of the functional unit and detection logic. Hence we can find the most energy-efficient functional units.

Figure 3-20 shows the re-synthesis flow. The input files are the netlist file of the functional unit and the violating paths report, and the functional unit is operated at the given clock period. The delay time of all violating paths is larger than the given clock period. In order to reduce the number of violating paths we need to perform one pre-step and three recursive steps.



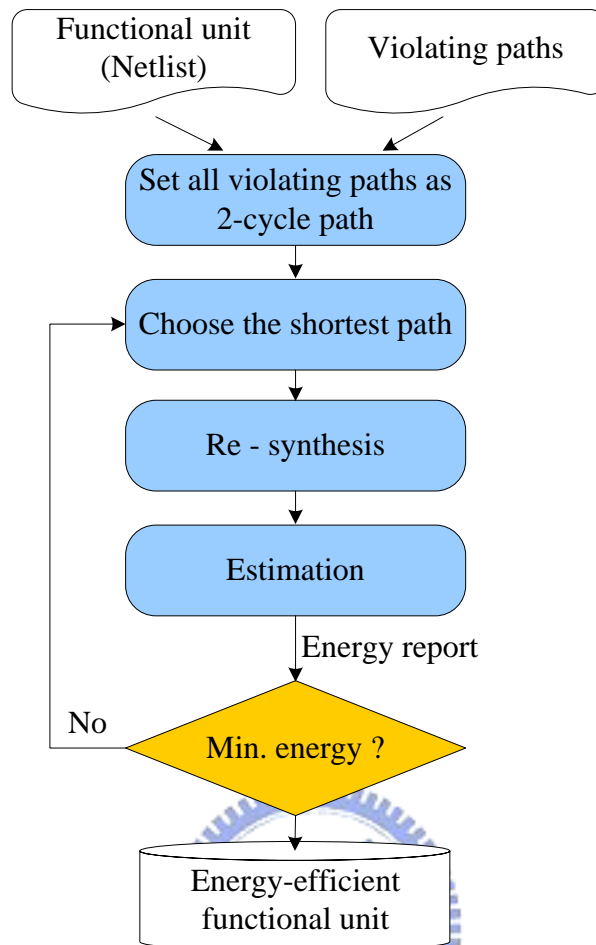


Figure 3-20 Re-synthesis flow

The pre-step is to set all violating paths as 2-cycle path. The violating paths represent that they can not complete computation in one clock cycle. In other words, they need two clock cycles. Because the number of the violating path may be very great, we only consider the input ports and output ports of the functional unit.

Next we select the “shortest path” of all violating paths as one-cycle path. Then we re-synthesize the functional unit such that path delay of the “shortest path” will be optimized. In other words, the path delay of the “shortest path” will meet one clock period such that the number of violating path will be decreased. Hence, the performance penalty and complexity of detection logic are also reduced, but the area and energy of function unit may be increased. After re-synthesis, we need to

estimate the energy consumption of the functional unit and the detection logic. The above steps need to be executed iteratively until the minimum energy design with negligible performance penalty is found.

The re-synthesis flow is a heuristic solution, because we only consider the input port and output port of the violating path instead of all gates on the path. Although the re-synthesis can't adjust the violating paths precisely, it can find a not bad solution and very fast.

Here we choose a case that shown in Figure 3-18 to perform the re-synthesis flow. The scale of the error rate (y-axis) is rearranged and the result (before re-synthesis) is shown in Figure 3-21.

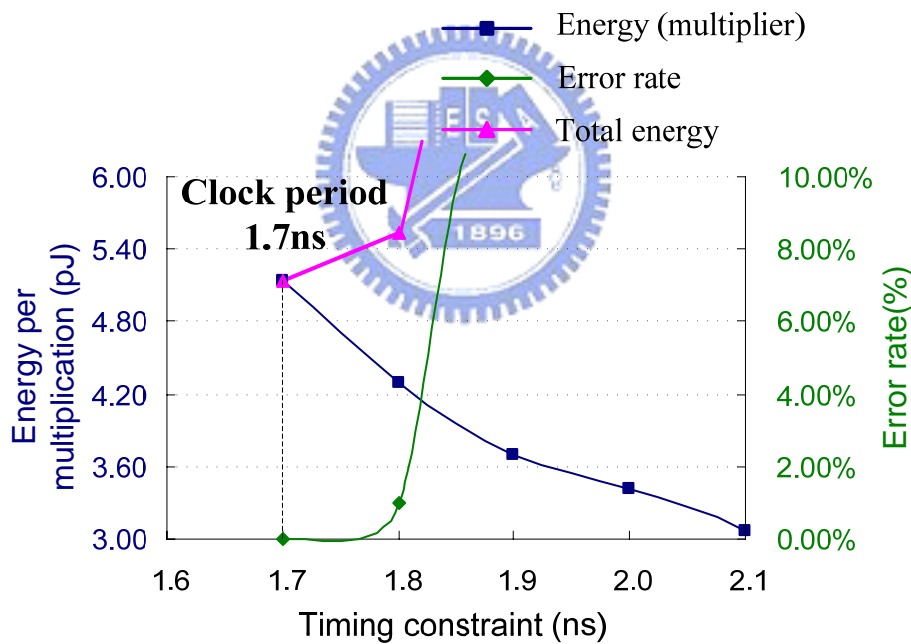


Figure 3-21 Before overhead reduction (re-synthesis)

Then perform the re-synthesis flow, the results is shown in Figure 3-22. The error rate can be reduced effectively by the re-synthesis method, but the multiplier energy is also increased. We also can find a convex curve of total energy that it has a minimum energy value with negligible error rate.

The multiplier with 1.8ns timing constraint consumes 4.56pJ (multiplier + detection logic), and it is operated at 1.7ns clock period only with 0.13% performance penalty.

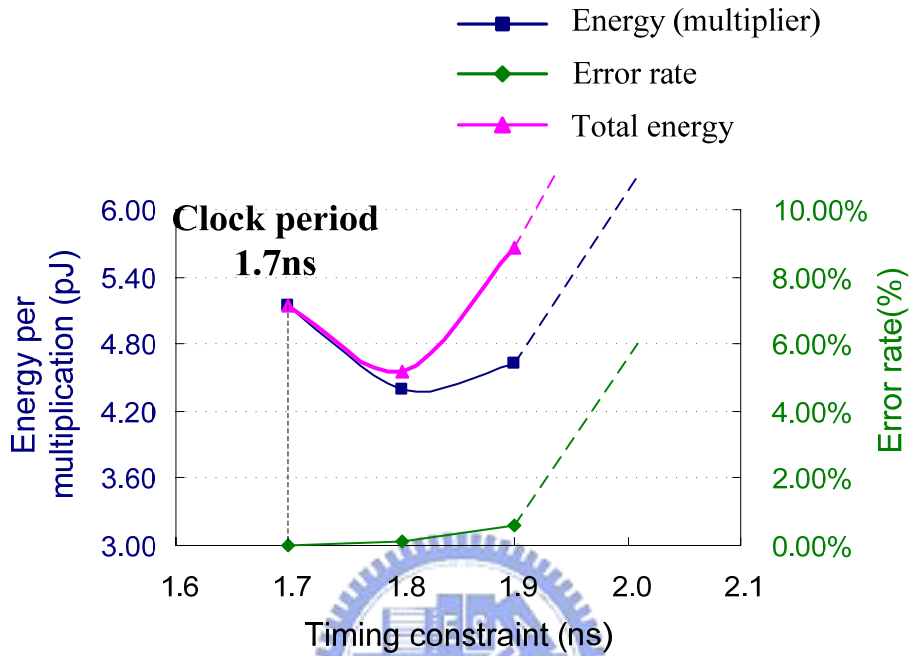
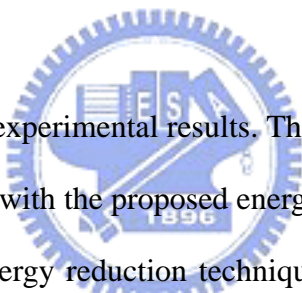


Figure 3-22 After overhead reduction (re-synthesis)

4 EXPERIMENTAL RESULTS



In chapter 4 we show two experimental results. The first is to evaluate the energy and area of the functional unit with the proposed energy-efficient design. The second is to compare the different energy reduction techniques for functional unit of DSP core. These techniques that consist of parallel design, pipeline design and proposed design are evaluated from area, energy, and performance penalty.

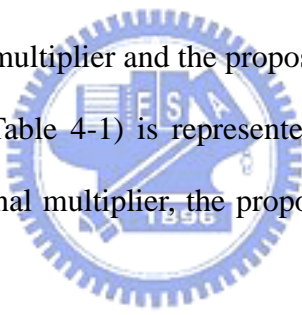
4.1 Simulation Results of Energy-Efficient Design

Most digital signal processor systems incorporate a multiplier to implement algorithms such as convolution and filtering. In many DSP algorithms, the multiplier lies in the critical path delay and ultimately determines the performance of the algorithm. However, the demand for high-performance portable systems incorporating multimedia capabilities has elevated the design for low-energy to the forefront of design requirement in order to maintain reliability and provide longer

hours of operation.

Hence, the objective of this experiment is to find the most energy-efficient multiplier that is operated at the peak clock frequency by our proposed design flow. The multiplier is 8 bit and the structure of the multiplier is Booth-encoded Wallace-tree. It is one of the fastest multipliers from Synopsys DesignWare IP. The most energy-efficient multiplier represents that it is the one that consumed the least energy among all configurations that deliver almost the same performance [40].

In this experiment, the proposed design flow is performed, and the multiplier is synthesized using the Synopsys Design Compiler (Version V2007-03) with the sub-set UMC 90nm CMOS cell library. The timing constraint is 1.2ns, because it is the peak value of the multiplier can achieve. The Table 4-1 summarizes the area of the conventional single-cycle multiplier and the proposed energy-efficient multiplier. The overhead in column 3 (Table 4-1) is represented the area of detection logic. Compared with the conventional multiplier, the proposed multiplier can save about 30% of total area.



	Area (um ²)		
	Multiplier	Overhead	Total
Conventional	6816	0	6816
Proposed	3982	782	4764
Improvement			30%

Table 4-1 Estimated area

After the gate level synthesis, we used the Synopsys PrimePower (Version V2006.06) to estimate the energy consumption (energy per multiplication) with back-annotated timing and parasitic information. The multiplier works at 833 MHz and computes 10,000 random patterns. Table 4-2 shows the energy per

multiplication of the conventional multiplier and the proposed multiplier. The overhead in column 3 (Table 4-2) is represented the energy consumption of detection logic. Compared with the conventional multiplier, the proposed multiplier can save about 21% of total energy.

	Energy (pJ)		
	Multiplier	Overhead	Total
Conventional	3.236	0.000	3.236
Proposed	2.387	0.178	2.565
Improvement			21%

Table 4-2 Estimated energy per multiplication

The proposed multiplier has 0.17% error rate that is estimated by 10,000 random patterns. In other words, the conventional multiplier needs 10,000 cycles to compute the 10,000 random patterns, but the proposed multiplier needs 10,017 cycles to compute the 10,000 random patterns. Compared with the energy reduction (21%), the performance penalty (0.17%) is very light.

If we synthesize the multiplier with power optimization constraint, the multiplier energy with conventional synthesis strategy can be reduced to 2.8pJ. Our proposed method can also apply the power optimization constraint, and the energy of the proposed multiplier can also be reduced. The ratio of the energy reduction by proposed method is almost the same as the ratio of the energy reduction by conventional method.

4.2 Comparison of Energy Reduction Techniques

Although relaxing the timing constraint is an effective method for energy reduction, it directly causes the performance (clock frequency) degradation. The

proposed energy-efficient design can compensate the clock frequency degradation, and it has only light performance penalty.

Exploiting parallelism or pipelining is also an effective method to compensate for the loss in performance. They can achieve the same clock frequency and data throughput, but the data latency is increased such that high data parallelism is required.

In this experiment, we compare the three different energy reduction techniques for multiplier of DSP core. The multiplier is the major component of DSP core, and it consumes the most energy among the datapath (adder, shifter and multiplier) in DSP core. The different techniques that consist of parallel design, pipeline design and proposed design, they are evaluated from area, energy, and performance penalty. The structure of the 8-bit multiplier is Booth-encoded Wallace-tree.

The baseline DSP model is similar as MIPS architecture [42]. It is a single-issue and in-order pipeline with 5 pipeline stages. The conceptual architecture is shown in Figure 4-1. It has forwarding path and only the arithmetic instructions are considered such that the ideal CPI of this DSP model is 1.

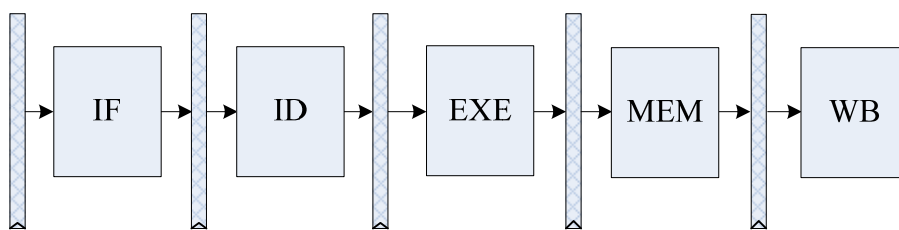


Figure 4-1 Baseline DSP model

The DSP model with parallel multiplier design and pipeline multiplier design are shown in Figure 4-2 and Figure 4-3 respectively. Although the two designs can reduce the multiplier energy and achieve the same throughput, they may suffer data hazards from limited instruction level parallelism. When the result of multiplier is

needed to the next instruction immediately, it can not be forwarded immediately such that these systems require a stall cycle. If the application has very poor instruction level parallelism, the performance of the systems will be degraded seriously.

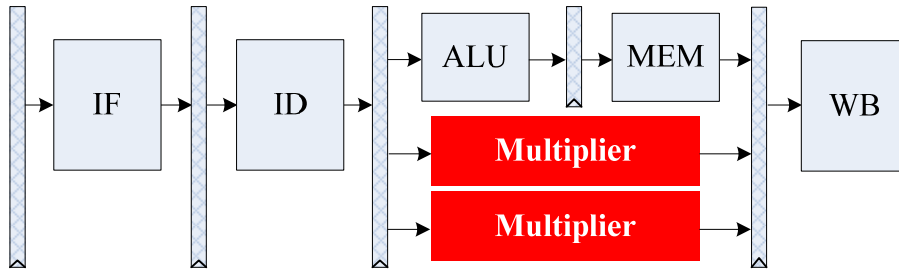


Figure 4-2 DSP model with parallel multiplier design

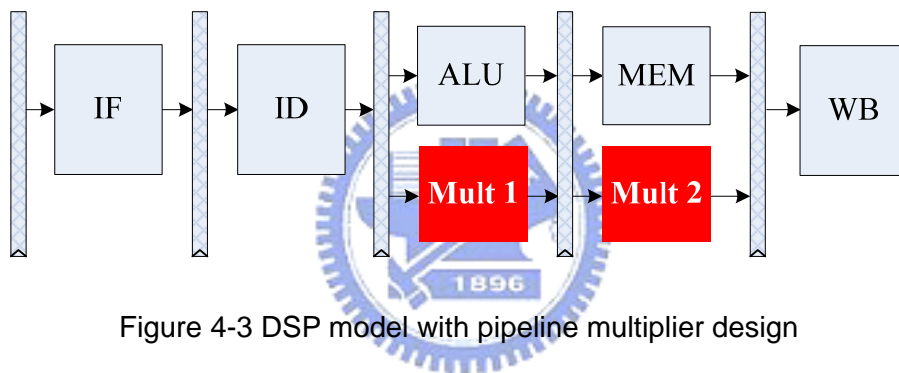


Figure 4-3 DSP model with pipeline multiplier design

The DSP model with proposed multiplier design is shown in Figure 4-4. The multiplier is augmented with the detection logic. When the detection logic detects a computation error in multiplier, all of the previous stages are stalled one cycle and the next stage is inserted a bubble (no-op) signal. Since the multiplier has one additional cycle to re-compute the multiplication such that the computation error can be corrected, but it spend a one-cycle latency penalty.

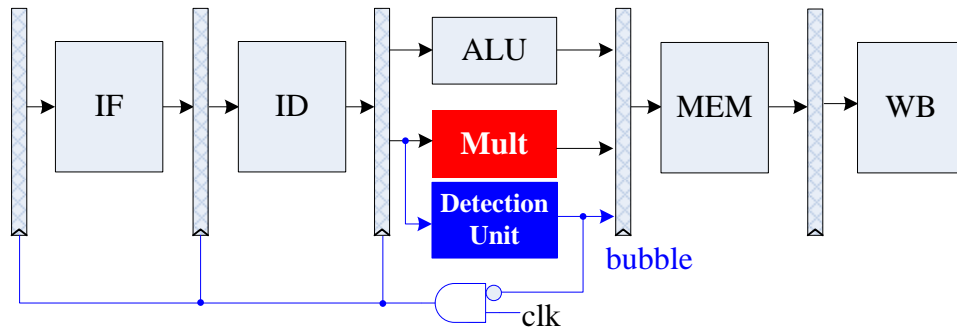


Figure 4-4 DSP model with proposed multiplier design

We use the sub-set of UMC 90nm CMOS cell library to evaluate area of multiplier and overhead. The operating period is 1.2ns that is the peak performance of the multiplier. The area of the multiplier is shown in Table 4-4. The single cycle design has the maximum. Compared with single cycle design, all of the three designs can reduce area effectively. Although the parallel design has two multipliers, the total area of the parallel design is smaller than the total area of the single cycle design. The timing constraint of the parallel design can be relaxed to double of the clock period such that the average area per multiplier is the minimum of four designs.

Design	Area (um ²)
Single cycle	6816
Parallel	5967
Pipeline	4175
Proposed	4764

Table 4-3 Estimated total area

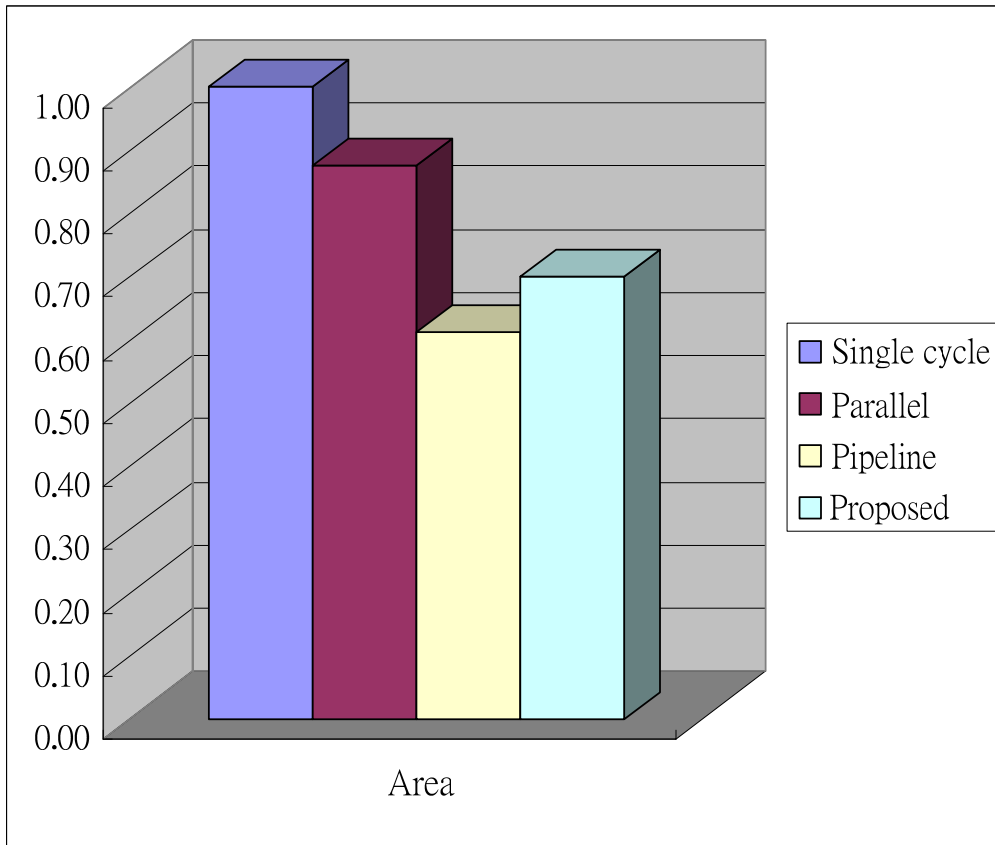


Figure 4-5 Area is normalized by the result of single cycle design

We use the MIPS compiler (gcc 4.21) to compile the benchmarks. One benchmark is color space transform of JPEG [43], and the input file is 64 x 64 Lena image. Another benchmark is a 16 taps finite impulse response (BDTI benchmark [44]), and the input file has 200 samples.

After the benchmark is compiled, we count the cycle count and generate the input patterns for multiplier based on the different DSP models. We take the assembly code of Chroma part to explain how we count the cycle count, and the code is shown in Figure 4-6.

```

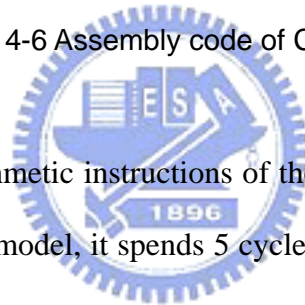
$L15:
...
lw    $3,0($2)
lw    $2,8($fp)
#nop
mult $3, $2
mflo  $2
sw    $2, 52($fp)
...
lw    $3, 0($2)
lw    $2, 4($fp)
#nop
mult $3, $2
mflo  $3
lw    $2, 52($fp)
#nop
addu $2, $3, $2
sw    $2, 48($fp)
...

lw    $3, 0($2)
lw    $2, 0($fp)
#nop
mult $3, $2
mflo  $3
lw    $2, 48($fp)
#nop
addu $2, $3, $2
sw    $2, 44($fp)
...

$L14:
lw    $2,60($fp)
#nop
slt   $2, $2, 8
bne   $2, $0, $L15
...

```

Figure 4-6 Assembly code of Chroma



We only consider the arithmetic instructions of the code — 3 multiplication and 2 addition. For baseline DSP model, it spends 5 cycles to execute the 5 instructions. For DSP model with parallel or pipeline multiplier design, it spends 7 cycles to execute the 5 instructions, because “addition after multiplication” incurs data hazard. The output of multiplier can not forward to adder immediately. For the DSP model with proposed multiplier design, it spends 5~8 cycles to execute the 5 instructions according to the computation errors in multiplier.

Hence, the total cycle count of finishing color space transform of the Lena image and 200 samples of 16-tap FIR is shown in column 2 and column 4 of Table 4-4. Both the parallel and pipeline design spend longer execution time, because they has many stall cycles according to the data hazards. “Addition follows multiplication” is very common in many applications, such as image, audio, and video.

Column 3 and column 5 of Table 4-4 show the energy consumption of these

multipliers to compute the multiplications of color space transform and the 16-tap FIR.

Design	Color space transform		FIR	
	Cycle count	Energy (nJ)	Cycle count	Energy (nJ)
Single cycle	34,816	61.37	6,400	10.45
Parallel	47,104	42.44	9,600	6.96
Pipeline	47,104	46.09	9,600	8.62
Proposed	34,876	46.74	6,412	7.43

Table 4-4 Simulation results

Figure 4-7 shows the simulation results that are normalized by the results of single cycle design. All of the three techniques can reduce the energy consumption effectively, but both the parallel and pipeline design spend longer execution time according to the data hazards.

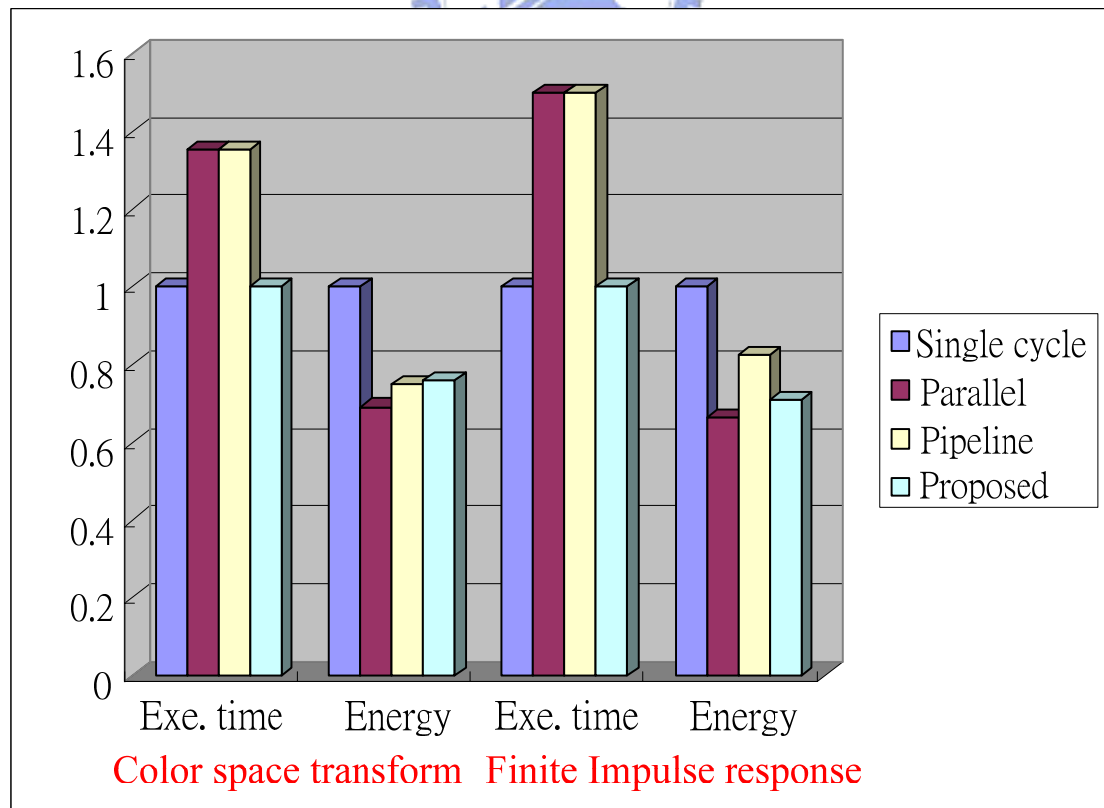


Figure 4-7 Normalized by the result of single cycle design

The energy-delay value is shown in Figure 4-8, the results are normalized by the result of single cycle design. We can find that the proposed design is the most energy-efficient.

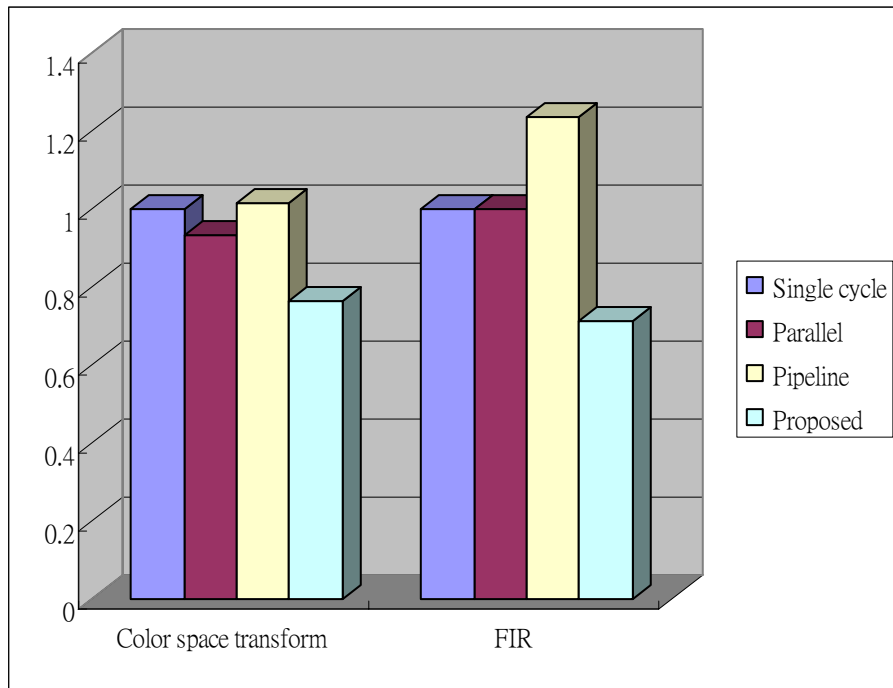
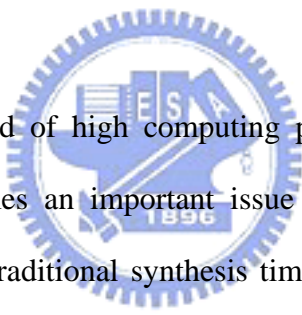


Figure 4-8 Energy-delay value normalized by single cycle design

5 CONCLUSIONS



For the increasing demand of high computing power and high mobility, the energy reduction now becomes an important issue in the VLSI designs. In the synchronous digital circuits, traditional synthesis timing constraint is based on the given clock period for function correctness, but that always makes the energy soar when the clock frequency and timing constraint approach the peak value. In this thesis, we propose a design method for improving energy efficiency of functional units. It exploits data-dependent delay to reduce synthesis timing constraint such that the energy consumption can be reduced effectively and the desired clock period will not be degraded, but it will cause computation errors and spend performance penalty for correcting the errors.

For the computation errors, we generate the detection logic to detect and spend one-cycle latency penalty to correct. The detection logic is generated by analyzing the violating paths under transition delay such that it can detect errors more precisely. If the error is detected by detection logic, all the previous pipeline stages


are latched one more cycle to re-compute the data, and the next stage is inserted a bubble (no-op) signal. We also propose a design flow which systematically determines the timing constraint and fine tune the number of violating paths for maximizing energy reduction and minimizing performance penalty.

In our simulation, we use the proposed technique in 8-bit multiplier to reduce energy consumption with the UMC 90nm CMOS cell library. The energy consumption can be reduced about 10% ~ 29% and the performance penalty is negligible (<1%). We further compare our proposed technique with the techniques of exploiting parallelism (parallel design and pipeline design) in a multiplier of DSP core. All the techniques can reduce the circuit energy effectively, but the techniques of exploiting parallelism have worse performance. Because exploiting parallelism design need high instruction level parallelism otherwise it incurs serious data hazards and stall cycles.

Our future work is to improve the energy and area of detection logic. The complexity of detection logic increases drastically when the error rate lightly increases. That limits the degree of relaxing the timing constraint. Because the detection logic is a Boolean function, we can insert some don't care patterns such that the complexity of the function can be simplified effectively. We now investigate the systematic method that can insert some don't care patterns efficiently.

At the same time, because energy is proportional to the square of the supply voltage, voltage scaling is one of the most effective methods for energy reduction. We will apply the same concept to reduce the supply voltage for energy reduction.

REFERENCE

- 
- [1] P. P. Gelsinger, “Gigascale integration for teraops performance-challenges, opportunities, and new frontiers,” in *Proc. DAC*, 2004
- [2] V.Venkatachalam, M. Franz, “Power reduction techniques for microprocessor systems,” *ACM Computing Surveys*, Sep. 2005
- [3] I. Buchmann. Batteries in a Portable World. [Online]. Available: <http://www.cadex.com>
- [4] K. Lahiri, A. Raghunathan, S. Dey, D. Panigrahi, “Battery-driven system design : a new frontier in low power design,” in *Proc. VLSI*, 2002
- [5] A. P. Chandrakasan, S. S. Bradersen, and R. W. Brodersen, “Low-power CMOS digital design,” *IEEE J. Solid-State Circuits*, pp. 473-483, Apr. 1992
- [6] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd Edition, Prentice Hall, 2003
- [7] D. Emst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T.

- Austin, K. Flautner, and T. Mudge, "Razor : a low-power pipeline based on circuit-level timing speculation," in *Proc. Micro*, 2003
- [8] G. Wolrich, E. McLellan, L. Harada, J. Montanaro, and R. Yodlowski, "A high performance floating point coprocessor," *IEEE J. of Solid-State Circuits*, Oct. 1984
- [9] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for Better than Worst-Case Design," in *Proc. ASP-DAC*, Jan. 2005
- [10] A. Keshavarzi, K. Roy, and C. Hawkins, "Intrinsic leakage in low power deep submicron CMOS ICs," in *Proc. ITC*, 1997
- [11] L. Wei, Z. Chen, K. Roy, M. C. Johnson, Y. Ye, and V. K. De, "Design and optimization for dual-threshold circuits for low-voltage low-power applications," *IEEE Trans. VLSI Systems*, Mar. 1999
- [12] Z. Chen, C. Diaz, J. D. Plummer, M. Cao and W. Creene, "0.18 μ m dual Vt MOSFET process and energy-delay measurement," in *Proc. IEDM*, 1996
- [13] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE J. Solid-State Circuits*, vol. SC-19, Aug. 1984
- [14] A. Chatterjee, "An investigation of the impact of technology scaling on power wasted as short-circuit current in low voltage static CMOS circuits," in *Proc. ISLPED*, 1996
- [15] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," in *Proc. IEEE*, Apr. 1995
- [16] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999
- [17] K. Usami and M. Igarashi, "Low-power design methodology and applications utilizing dual supply voltages," in *Proc. ASP-DAC*, Jan. 2000

- [18] S. H. Kulkarni and D. Sylvester, "Power distribution techniques for dual VDD circuits," in *Proc. ASP-DAC*, 2006
- [19] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanazawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE J. of Solid State Circuits*, Mar. 1998
- [20] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. VLSI Systems*, Dec. 1997
- [21] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Trans. VLSI Systems*, Dec. 1997
- [22] C. Im, H. Kim, and S. Ha, "Dynamic voltage scheduling techniques for low-power multimedia applications using buffers," in *Proc. ISLPED*, Aug. 2001
- [23] H. Saputra, M. Kandemir, N. Vijaykrishnan, M. Irwin, J. Hu, C.-H. Hsu, and U. Kremer, "Energy-conscious compilation based on voltage scaling," *LCTES*, 2002
- [24] L. Benini, P. Siegel, and G. De Micheli, "Automatic synthesis of gated clocks for power reduction in sequential circuits," *IEEE Design Test Computer Magazine*, pp. 32-40, 1994
- [25] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Systems*, 1994
- [26] J. Monteiro, S. Devadas, A. Ghosh, "Sequential logic optimization for low power using input-disabling," *IEEE Trans. Computer-Aided Design*, 1998
- [27] L. Benini, G. D. Micheli, A. Liroy, E. Macii, G. Odasso, and M. Poncino, "Synthesis of power-managed sequential components based on computational

- kernel extraction,” *IEEE Trans. Computer-Aided Design*, Sep. 2001
- [28] G. Paci, P. Marchal, and L. Benini, ”Exploration of low power adders for a SIMD data path,” in *Proc. ASP-DAC*, Jan. 2007
- [29] S. Dhar, D. Maksimovic, and B. Kranzen, ”Closed-loop adaptive voltage scaling controller for standard-cell ASICs,” in *Proc. ISLPED*, Aug. 2002
- [30] J. Choi, J. Jeon, and K. Choi, ”Power minimization of functional units by partially guarded computation,” in *Proc. ISLPED*, Jul. 2000
- [31] B. Razavi, *Design of Analog CMOS Integrated Circuits*, New York: McGraw-Hill, 2001
- [32] S. Padmanaban, M. K. Michanel, and S. Tragoudas, ”Exact path delay fault coverage with fundamental ZBDD operations,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2003
- [33] H.-C. Tsai, K.-T. Cheng, and V. D. Agrawal, ”A testability metric for path delay faults and its application,” in *Proc. ASP-DAC*, Jan. 2000
- [34] Synopsys Prime Time User Guide (Fundamentals). [Online]. Available: <http://www.synopsys.com>
- [35] H.-C. Chen and D. H.-C. Du, ”Path sensitization in critical path problem,” *IEEE Trans. Comput. -Aided Design Integrated Circuits*, Feb. 1993
- [36] P. McGeer and R. Brayton, ”Efficient algorithms for computing the longest viable path in a combinational network,” in *Proc. DAC*, 1989
- [37] B. Konemann, J. Barlow, P. Chang, R. Gabrielson, C. Goertz, B. Keller, K. McCauley, J. Tischer, V. Iyengar, B. Rosen, and T. Williams, ”Delay test: the next frontier for LSSD test systems,” in *Proc. Int. Test Conf.*, Oct. 1992
- [38] A. Kristic and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Publishers, Boston, 1998.
- [39] H. Choi and S. H. Hwang, ”Practical use of transition mode delay to solve the

problem of floating mode delay under highly correlated input streams,” in *Proc. ICCD*, Oct. 1998

[40] M. Horowitz, T. Indermaur, and R. Gonzalez, “Low-power digital design,” in *Proc. ISLPED*, Oct. 1994

[41] S. Raj, S. B. K. Vrudhula, and J. Wang, “A methodology to improve timing yield in the presence of process variations,” in *Proc. DAC*, 2004

[42] J. L. Hennessy, and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufman Publishers, 4th Edition, 2007

[43] Independent JPEG Group. [Online]. Available: <http://www.ijg.org>

[44] Berkeley Design Technology Inc. [Online]. Available: <http://www.bdti.com>





作者簡歷

林彥呈，西元 1984 年 6 月 21 日出生於台中縣。西元 2006 年取得國立中興大學電機工程學系學士學位，並於同年在國立交通大學電子工程研究所攻讀碩士。西元 2008 年在劉志尉教授指導下，取得碩士學位。本篇論文「利用與資料相依之延遲改善運算單元之能量效率」為其碩士論文。

