2004

Energy Aware Design and Analysis for Synchronous and Asynchronous Circuits

Jia Di

ENERGY AWARE DESIGN AND ANALYSIS FOR SYNCHRONOUS AND ASYNCHRONOUS CIRCUITS

By
Jia Di

2004

UCF

ENERGY AWARE DESIGN AND ANALYSIS FOR SYNCHRONOUS AND ASYNCHRONOUS CIRCUITS

by

JIA DI
B.S. Tsinghua University, China, 1997
M.S. Tsinghua University, China, 2000

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2004

Major Professor: Dr. Jiann S. Yuan

ABSTRACT

Power dissipation has become a major concern for IC designers. Various low power design techniques have been developed for synchronous circuits. Asynchronous circuits, however, have gained more interests recently due to their benefits in lower noise, easy timing control, etc. But few publications on energy reduction techniques for asynchronous logic are available.

Power awareness indicates the ability of the system power to scale with changing conditions and quality requirements. Scalability is an important figure-of-merit since it allows the end user to implement operational policy, just like the user of mobile multimedia equipment needs to select between better quality and longer battery operation time.

This dissertation discusses power/energy optimization and performs analysis on both synchronous and asynchronous logic. The major contributions of this dissertation include:

1) A 2-Dimensional Pipeline Gating technique for synchronous pipelined circuits to improve their power awareness has been proposed. This technique gates the corresponding clock lines connected to registers in both vertical direction (the data flow direction) and horizontal direction (registers within each pipeline

stage) based on current input precision.

2) Two energy reduction techniques, Signal Bypassing & Insertion and Zero Insertion, have been developed for NCL circuits. Both techniques use Nulls to replace redundant Data 0's based on current input precision in order to reduce the switching activity while Signal Bypassing & Insertion is for non-pipelined NCL circuits and Zero Insertion is for pipelined counterparts. A dynamic active-bit detection scheme is also developed as an expansion.

3) Two energy estimation techniques, Equivalent Inverter Modeling based on Input Mapping in transistor-level and Switching Activity Modeling in gate-level, have been proposed. The former one is for CMOS gates with feedbacks and the latter one is for NCL circuits.

ACKNOWLEDGMENTS

I would like to thank Theseus Logic, Inc. for their financial support and the opportunity to work with such a novel and exciting technology. I would like to thank Dr. Jiann S. Yuan for his technical and editorial advice that has helped shape this work. His keen insight into IC design and support are the key factors for the success of this research. I would like to thank the committee members who have taken the time to review and comment on this dissertation.

I am deeply grateful to my mom and dad for their love and support in my whole life. But most of all I would like to thank my loving wife Yan, for her understanding, care, encouragement, and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

1.1 Objective

This dissertation intends to familiarize the reader with the design techniques and methods of power optimization and estimation in synchronous and asynchronous digital circuits. The main focus will be on low power techniques to improve power awareness in synchronous circuits and Null Convention Logic (NCL) circuits.

1.2 Why Low Power?

Power dissipation has long been a major concern of IC designers. There are two main reasons for low power design. One is the dramatic decreasing of feature size and the increasing of chip density and clock frequency. The resulting high power dissipation could make the chip temperature grow up so that cooling device must be applied. But these devices also make the product have high price and become less reliable. The other reason is the growing demand of mobile communication and computing devices [1]. These devices use batteries as power source. Since the amount of power that a battery can provide is limited, how to make the device work for a longer time without recharging the battery is critical. Besides the attempts to increase the energy stored in the batteries, making the chip consume less power is the goal for IC designers.

1.3 Power Dissipation in CMOS Circuits

There are three major components of power dissipation in CMOS circuits: charging/discharging power, short circuit power, and leakage power. Figure 1 shows the power dissipation in CMOS inverter (leakage power is not shown).

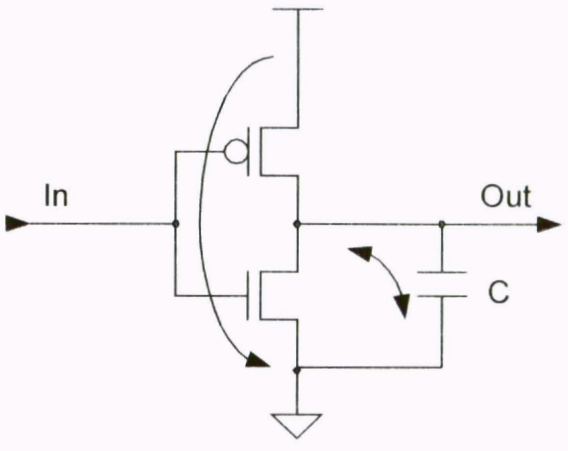


Figure 1 Power dissipation in CMOS inverter

During the output transition from either '0' to '1', alternatively, from '1' to '0', bother n- and p-transistors are on for a short period of time. This results in a short current pulse from V_{dd} to V_{ss}. This current causes a "short-circuit" power dissipation that is dependent on the input rise/fall time, the load capacitance and gate design. During the transition, current is also required to charge or discharge the output capacitance load. This current causes charging/discharging power dissipation. This term is usually the dominant term. When the circuit is in static mode, there is some small static dissipation due to reverse bias leakage between diffusion regions and the substrate. In addition, subthreshold conduction can contribute to the static dissipation. This dissipation is called leakage power dissipation [2].

1.4 Synchronous vs. Asynchronous

Currently there are two major logic design categories: synchronous logic and asynchronous logic. Between these two, synchronous logic is the dominant figure in IC market. Most circuit designs are built with synchronous logic. The biggest advantage of synchronous logic, which is controlled by a global clock and its derivatives, is that synchronous logic makes it easy to determine the maximum operating frequency of a design by finding and calculating the longest delay path between registers in a circuit [3]. As the feature size keeps scaling down and the System-On-a-Chip (SOC) technology makes it possible to group multi-million gates on one chip, it is getting extraordinarily difficult to determine the critical path delays. And with the growing needs of high operating frequency, the clock tree in synchronous circuit causes a lot of problems like high power dissipation, thermal effects, noise and EM radiation, etc.

Asynchronous logic, on the other hand, does not have a clock tree. So time is no longer discrete. While research in asynchronous design goes back to the 1950s and has received varied levels of attention over the decades, there is now a major and ongoing resurgence of interest. The advantages of asynchronous logic include no clock skew, average-case performance, high energy efficiency, robust input timing requirements, and low noise/emission. But asynchronous logic also faces some problems like circuit hazards, high design complexity, and lack of CAD tool support. Null Convention Logic (NCL), patented by Karl Fant and Scott Brandt in April of 1994, is used as asynchronous logic example in this dissertation.

1.5 Research Challenges

This dissertation proposes power estimation and reduction techniques for both synchronous logic and NCL. Since NCL is still conceptual new, there is no previous work studying energy estimation and reduction for NCL. NCL differs significantly from Boolean logic; so traditional Boolean power optimization and estimation techniques cannot be applied to NCL circuits directly.

Null states account for approximately half of the total operating cycles of NCL circuits. These Null states are only used to allow pipeline stages to reset and do not transfer data. But they are essential in NCL circuit operation and cost additional switching. To estimate switching activity in gate-level, the presents of Null states must be considered.

NCL is a speed-independent circuit design style. That means the delay of individual gate does not affect the function of the circuit. The unique structure of NCL leads itself to pipelining, even though a clock is not present. To reduce energy dissipation, the speed-independency and the pipeline structure need to be maintained.

1.6 Dissertation Overview

This dissertation is organized into six chapters. Chapter 2 presents previous work and contains an in-depth discussion of power reduction techniques and estimation methods. In Chapter 3, a 2-Dimensional Pipeline Gating technique to improve power awareness in pipelined synchronous circuits is developed. This method is then implemented and tested in unsigned and signed array multipliers. In Chapter 4, the energy awareness problem of NCL is discussed. Then two techniques, Signal Bypassing & Insertion and Zero Insertion, are proposed to solve this problem and to design energy-

aware NCL circuits. These two techniques are implemented and tested in parallel adders, array multipliers, and Finite Impulse Response (FIR) filters. A dynamical active-bit detection scheme for adaptive energy-aware multiplier design is introduced at the end of this chapter along with an energy model. Chapter 5 is concentrated on two power estimation methods. The equivalent inverter modeling method based on input mapping is for transistor-level CMOS gate modeling. The switching activity modeling method is for gate-level NCL circuits. Chapter 6 highlights the contributions of this dissertation and provides direction for future research.

CHAPTER TWO: PREVIOUS WORK

2.1 Low Power Research For Synchronous Circuits

2.1.1 Power Estimation Methods

In order to analyze and design low power circuits, designers need to know the power information during the design cycle. Since during the design cycle the chip has not been made yet, the power dissipation has to be estimated. Due to the different phases in the design cycle, there are different power estimation techniques for different "levels" of circuit design. Figure 2 shows the low power design flow [4-5].

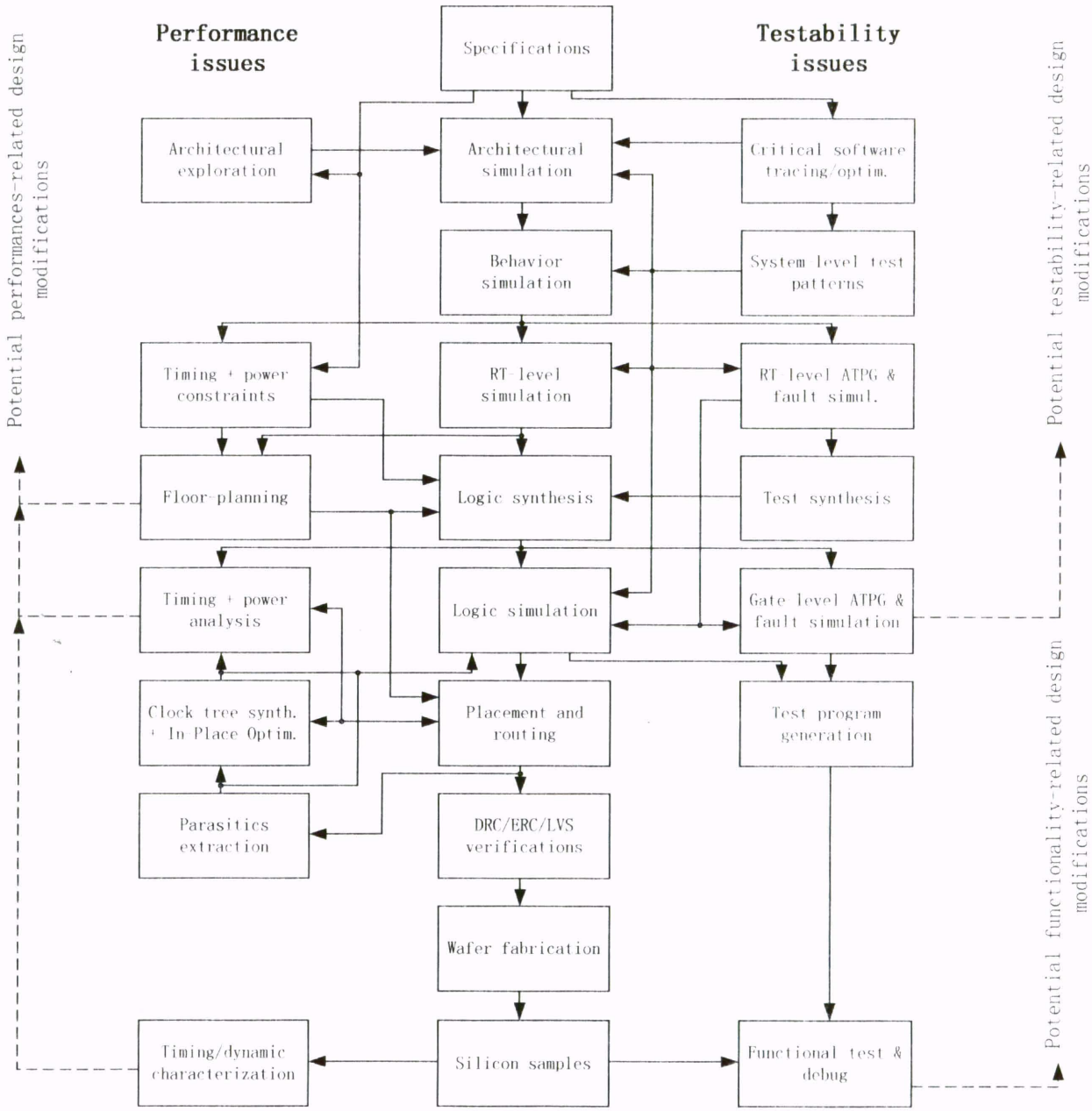


Figure 2 Low power design flow

2.1.1.1 Software-Level Power Estimation

To estimate power in this level, firstly typical application programs must be identified to be executed on the system. Since the program usually has millions of

instructions and operating cycles, it is impossible to estimation its power in lower levels. Most of the techniques in this level are macro-modeling, an estimation approach that is extensively used for behavioral and RT-Level estimation. These techniques include characterizing power cost of a CPU module by estimating the average capacitance that would switch when the given CPU module is activated, estimating power consumption of microprocessor using the switching activities on buses, and profile-driven program synthesis approach [6-9].

2.1.1.2 Behavior-Level Power Estimation

In this level, no information about the gate-level structure presents. So power estimation in this level must resort to abstract notions of physical capacitance and switching activity. There are three major kinds of models used in this level. Using information-theoretic models, entropy is used to measure power. If the circuit structure is given, the total module capacitance is calculated by traversing the circuit netlist and summing up the gate loadings. Using complexity-based models, circuit complexity is modeled by equivalent gates or the area of optimized signal-output Boolean functions [10-16]. Using synthesis-based models, some RT-Level templates are assumed and the estimation is based on these assumptions. A quick synthesis capability is required to form the RT-Level structure. Approaches like static profiling and dynamic profiling are developed [17-20].

2.1.1.3 RT-Level Power Estimation

Most RT-Level power estimation techniques use regression-based, switched-capacitance models for circuit modules. Such techniques are called power macro-

modeling. Firstly every component in the high-level design library is characterized by simulating it under pseudorandom data and a multivariable regressive curve is fitted. Then the variable values for the macro-model equation are extracted either from static analysis of the circuit structure and functionality or by performing a behavioral simulation of the circuit. After this, the power macro-model equations for high-level design components are evaluated. These are found in he library by plugging the parameter values in the corresponding macro-model equations. Finally, the power dissipation for random logic or interface circuitry is estimated by simulating the gate-level description of these components or by performing probabilistic power estimation [21-27].

2.1.1.4 Gate-Level Power Estimation

In this level, the structure of the whole circuit is known. After simulating all input patterns, the average power dissipation can be achieved. But as the number of primary inputs increase, the total number of input patterns becomes so large that it is impossible to simulate them all. Most of the techniques in this level use the statistical information of the inputs instead. The information includes signal probability, transition probability, transition density, etc. The use of statistical information makes the simulation pattern-independent. After accurately modeled the signal correlation, these statistical information can be propagated through the circuit and the power dissipation of each node can be calculated.

2.1.1.5 Transistor-Level Power Estimation

In this level, detailed physical and mathematical equations are used to calculate power dissipation accurately. The techniques in this level are pattern-dependent. Although the results are quite accurate, the computing complexity makes them only be used in limited cases.

2.1.2 Power Reduction Techniques

To achieve low power design, many power reduction techniques have been developed in different levels.

2.1.2.1 Transistor Sizing

Transistor sizing is a circuit-level low power technique that targets on the short circuit power. It is the operation of enlarging/reducing the width of the channel of a transistor [28]. Changing the channel width of transistors will affect both delay and power. The effect of transistor sizing can be seen as trading extra speed for lower power dissipation. Earlier approaches were based on the assumption that the power dissipation is proportional to the active area, i.e., the area occupied by active devices. Recent study shows that the power dissipation is a convex function to the active area [28].

Since studying transistor-sizing problem of complex CMOS gates can involve complicated mathematical derivations and calculations, CMOS inverter is a good example to analyze. Equation (1) was derived for the maximum short-circuit dissipation under the no-load condition at the output of the inverter [29]

$$P_{SC} = \frac{\tau\beta}{12}(V - 2V_t)^3 f \qquad (1)$$

Here, τ is the input transition time, β is the gain-factor of the transistor, V is the supply voltage, V_T is the threshold voltage, and f is the transition frequency. The gain factor, β, is determined by the width of the transistor W and the mobility of the carrier responsible for the transition (μ_p for a low-to-high transition and μ_n for high-to-low). So equation (1) can be reduced to (2)

$$P_{SC} = k\mu W\tau f \tag{2}$$

where k is a process and voltage dependent constant of proportionality [28]. It is clear from (2) that P_{SC} is directly proportional to both the width of the transistors and the input transition time. Although derived from no-load condition, this equation still holds in the general loading case.

In [30], different drivers drive different loads were simulated and the relation between various parameters were drawn. Analytical derivation of the optimum value of W under high fan-out condition is discussed in [28]. A convex curve representing transistor size vs. average power dissipation is shown in Fig. 3 [28].

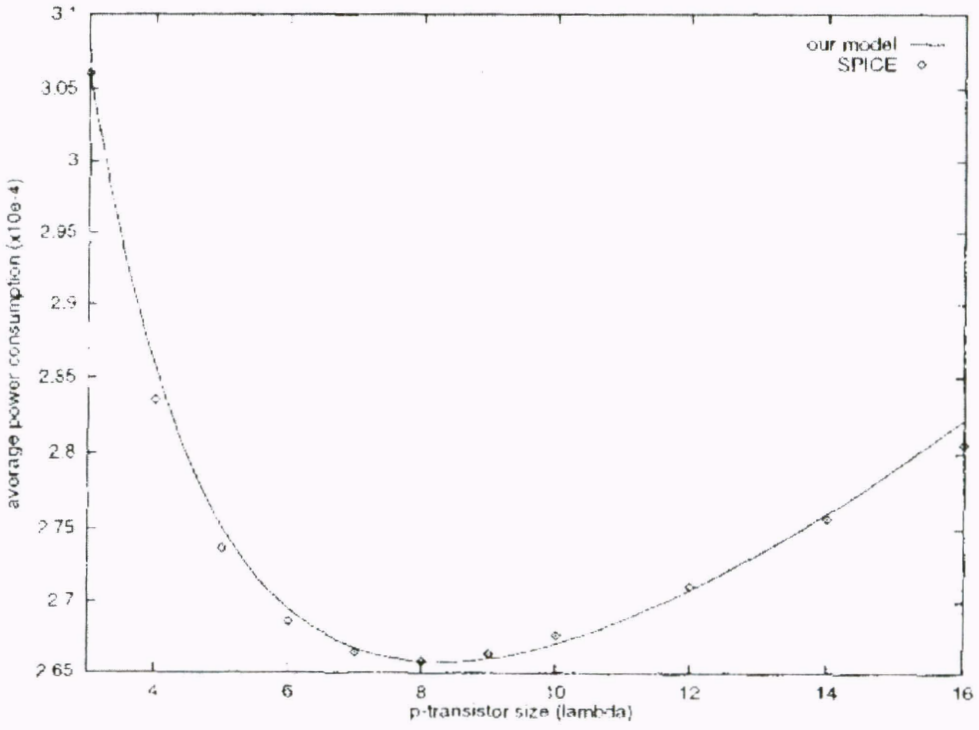


Figure 3. Transistor size vs. average power dissipation of inverter with load

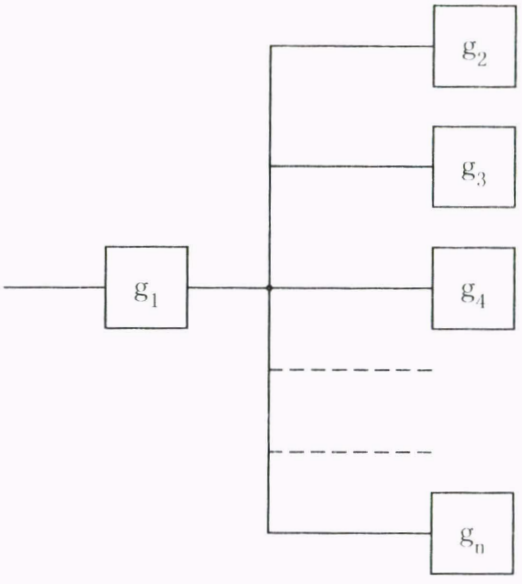


Figure 4. Sample circuit structure for calculating optimum transistor size

For the circuit in Fig. 4, detailed derivation shows the P_{SC} attains a minimum when

$$W_1 = \frac{\sqrt{\phi \frac{\mu'}{\mu} C_{L1} \left(\sum_{2}^{n} W_i f_i \right)}}{\sqrt{\left(\frac{k_1}{k} + \mu\tau \right) f_1}} \tag{3}$$

where W_1 is the power optimal size for the transistor in the driver inverter g_1, ϕ is a process dependent constant, μ' is the mobility of the carrier responsible for the opposite transition in the gates g_2 to g_n, C_{L1} is the total load capacitance of gate g_1. The power optimal p-transistor size is given by substituting μ_p for μ and μ_n for μ' in (3), and the power optimal n-transistor size can be obtained by substituting μ_n and μ_p, respectively [28]. It was shown in [28] that the power reduction rate is 35.35% under 10-inverter fan-out load.

2.1.2.2 Transistor Reordering

Transistor reordering is also a circuit-level power reduction technique. Although it is often implemented together with transistor sizing, transistor reordering mainly targets on the Dynamic Capacitive Switching Power, $P_{dynamic\text{-}C}$. The basic idea of transistor reordering is by adjusting the order of the transistors in a serial connected CMOS chain based on the behaviors of different inputs to achieve lower switching power. An illustration of this idea is shown in Fig. 5 [31].

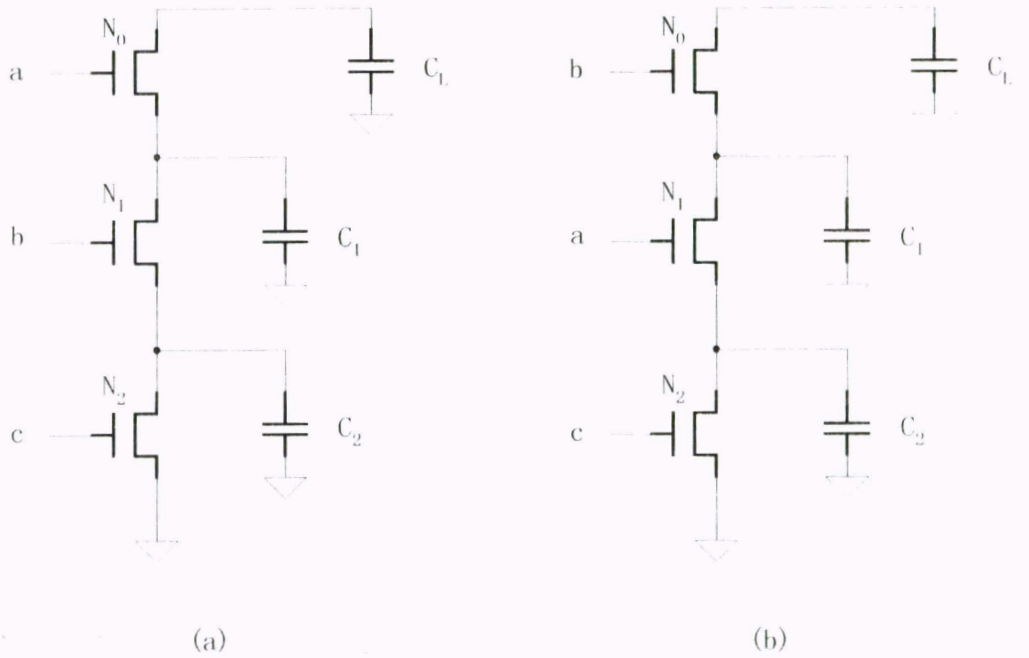


Figure 5. Illustration of transistor reordering

Figure 5 shows two alternate implementations of the nMOS component in a three-input NAND gate. In both gates, node N0 is the output of the gate, with two other internal nodes, N1 and N2, being present. Three inputs, a, b, and c, are connected to the corresponding transistors. Assume a vector 110 followed by 011 is applied to the three inputs in alphabetic order, and all three inputs arrive simultaneously. This sequence causes the capacitances C_L, C_1, and C_2 to be initially charged following which the capacitances C_1 and C_2 in Fig. 5(a) are discharged, while only capacitance C_2 in Fig. 5(b) is discharged. Same input sequence could cause different power dissipation while applied to different orders of transistor chains.

To determine the appropriate transistor order, some information of inputs must be known. One is the percentage of time that the specific input stays in logic high; the other is how often this input makes a transition. Since the exact information of actual inputs

cannot be known beforehand, only probabilistic information is needed. The former information is denoted as Signal Probability, defined by

$$p_i = \lim_{N->\infty} \frac{\sum_{k=1}^{N\times S} x_i(k)}{N \times S} \tag{4}$$

where p_i is the signal probability of input x_i, N is the total number of global clock cycles, S is the total number of time slots during one clock cycle due to the different delays of gates, $x_i(k)$ is the value of x_i during the interval of time instances k and $k+1$.

The latter one is denoted as Transition Density, defined by

$$D_i = \lim_{N->\infty} \frac{\sum_{k=1}^{N\times S} \left(x_i(k)\overline{x_i(k+1)} + \overline{x_i(k)}x_i(k+1) \right)}{N \times S} \tag{5}$$

where D_i is the transition density of input x_i.

With Signal Probability and Transition Density, the order of transistors can be determined. Detailed derivation and calculation are referred to [31] and [32]. A good summary of reordering rules is given in [32]. For NAND gate, always place the input signal with high signal probability near ground, and place the input signal with low transition density near ground. For NOR gate, always place the input signal with low signal probability near power supply, and place the input signal with low transition density near power supply.

2.1.2.3 Supply Voltage and Threshold Voltage Scaling

The equation of the Dynamic Capacitive Switching Power is recalled in (6). It is clear that $P_{dynamic\text{-}C}$ is quadratically proportional to the supply voltage because he voltage swing ΔV is determined by V_{DD}. So supply voltage reduction is one of the most efficient

ways to reduce power dissipation. But the supply voltage cannot be simply reduced. The propagation delay of a CMOS inverter under α–power law is shown in (7) [33]

$$P_{dynamic-C} \propto CV_{DD}\Delta VfN \tag{6}$$

$$T_g = K\frac{V_{DD}}{(V_{DD}-V_{th})^{\alpha}} \tag{7}$$

In (7), K is a proportionality constant specific to a given technology, V_{th} is the threshold voltage of MOS transistor, α is the power factor in α-power law model and its value is between 1 and 2. While V_{DD} is reduced, the total propagation delay will increase. The price could be high under throughput constraints.

From (7), if the threshold voltage could decrease with the reduction of V_{DD}, the increment of T_g could be compensated. But the decrement of V_{th} causes another problem. The equation of leakage current, I_l, is shown in (8) [33]

$$I_l = WI_s e^{\frac{V_{th}}{V_o}} \tag{8}$$

where W is the effective transistor width, I_s is the zero-threshold leakage current, and V_o is the subthreshold slope. When V_{th} is reduced, the leakage current, which has becoming more and more important in IC power dissipation, will increase, thus increase the power dissipation.

Several practical approaches have been developed to reduce the leakage current while using lower V_{DD} and V_{th}: Multi-threshold CMOS (MTCMOS) technology gates a high-V_{th} transistor with an inactive mode; Dual-V_{th} technology assigns gates on the critical paths to low V_{th} for speed, while gates that are not timing critical are assigned high V_{th} since they can tolerate larger delay; Clustered Voltage Scaling (CVS) assigns low supply voltage to circuits that have excessive slacks [34]. Other techniques like

Variable-Threshold CMOS (VTCMOS) [35] and Low-Swing Clock Logic [36] also take advantage of adjusting V_{DD} and V_{th}. These techniques all share the same goal: while satisfying the delay constraints, adjusting the supply and threshold voltages to lower power dissipation and control the leakage power within a tolerate range.

2.1.2.4 Clock Gating

In CMOS digital circuits, sequential part is always the major contributor of the total power dissipation. The main reason is the existence of the clock signal. Clock is the only signal that switches all the time. It is also most likely to drive a heavy load. Thus, reducing the clock power is an efficient way to minimize the total power dissipation.

Considering that there are a great deal of unnecessary switching activity caused by clock, a controlled clock which, based on certain conditions, can be slowed down or stopped completely with respect to the master clock, will bring significant power savings due to the following factors [37]:

1) The load on the master clock is reduced and the number of required buffers in the clock tree is decreased.
2) The flip-flop receiving the derived clock is not triggered in idle cycles.
3) The excitation function of the flip-flop triggered by the derived clock may be simplified since it has a do not care condition in the cycle when the flip-flop is not triggered by the derived clock.

Clock gating technique is an architecture-level power reduction technique. An example of clock gating technique used in pipeline is shown in Fig. 6 [38].

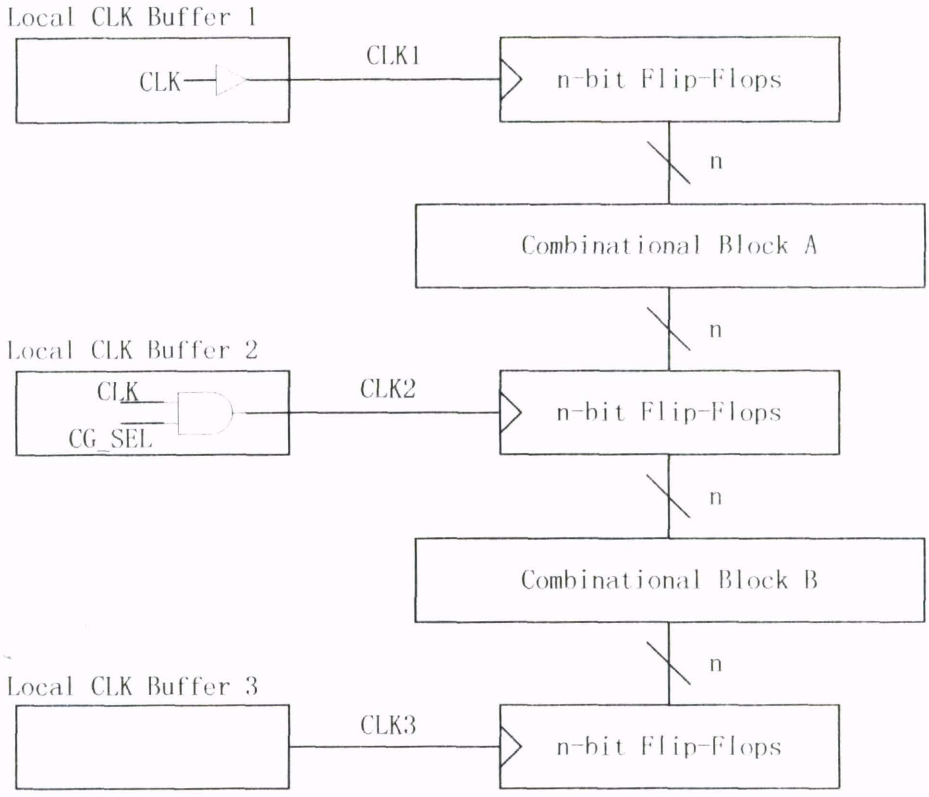


Figure 6. Clock gating example

A local clock buffer 1 is not clock-gated and always fires clock signal, CLK1, which feeds n-bit flip-flops. A local clock buffer 2 is clock-gated, and if CG_SEL signal is logic low then CLK2 is not fired. Hence, power consumption from CLK2 distribution and n-bit flip-flops can be neglected. Furthermore, dynamic power consumption of the block (combinational block B) following the clock-gated flip-flops can be saved during the clock-gating period [38]. Although clock gating only requires simple control circuitry, caution must be taken on glitch and additional clock skews.

2.1.3 Power Awareness

Besides those techniques that can be used in most of the situations, there are also power reduction techniques that are focused on specific conditions. An important parameter, power awareness, is introduced in [39]. Power awareness indicates the ability of the system power to scale with changing conditions and quality requirements. Scalability is an important figure-of-merit since it allows the end user to implement operational policy, just like the user of mobile multimedia equipment needs to select between better quality and longer battery operation time. The examples include that a well-designed system must gracefully degrade its quality and performance as the available energy resources are depleted [40]. These cannot be applied unless the circuit is designed in power-aware style.

2.2 Asynchronous Logic, NCL, And Energy Awareness

As a totally different design style compared to traditional synchronous circuit, asynchronous circuit has many potential advantages over the synchronous counterpart, including lower power dissipation by consuming power only when needed, free interchangeability of components between systems by avoiding global clock, higher performance by working under average performance instead of worst performance, lower noise by eliminating high-switching clock signal, and so on. Many asynchronous design methodologies have been proposed [41-45]. An asynchronous logic example in delay-insensitive design style, Null Convention Logic™ (NCL) is introduced in this section.

NCL is encoded in multi-rail encoding methodology. Multi-rail encoding is widely used in asynchronous circuits. Unlike Boolean logic, which uses single wire to express data 0 and 1, multi-rail logic uses at least two wires to interpret one signal value.

NCL is a symbolically complete logic that expresses process completely in terms of the logic itself and inherently and conveniently expresses asynchronous circuits. A more detailed introduction of NCL is in [44]. The logic families used in NCL are called threshold gates. Fig. 7 shows an *m*-threshold, *n*-input threshold gate, denoted as ThmnX0.

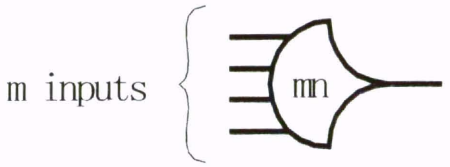


Figure 7. Thmnx0 gate

The function of threshold gate has hysteresis property, which is, when at least *m* out of *n* inputs are logic high, the output becomes logic high; when all *n* inputs are logic low, the output becomes logic low; otherwise the output remains unchanged. These threshold gates are an extension of the C-element (equals to a ThnnX0 gate) commonly used in delay-insensitive circuit design. The simplest NCL encoding is dual-rail logic, in which two wires are used for one signal. The truth table of dual-rail NCL is shown in Table 1 [44].

Table 1. Dual-rail NCL truth table

	Wire 1	**Wire 0**
Invalid	1	1
Data 1	1	0
Data 0	0	1
Null	0	0

There are two valid states in NCL: one is Data state, such as Data 0 and 1 in dual-rail encoding; the other is Null, represented by all wires being logic low. From Table 1 it is clear that in NCL multi-rail encoding, whatever the data value of the signal is, only one wire is logic high. Also, whatever the data value is, including Data 0, there is one wire in logic high.

The operation of NCL circuit includes Data-Null cycles, just as data-spacer cycles in some other multi-rail encoding logic. That means after a Data state, all signals in the circuit go to a Null state before next Data state comes. This Data-Null sequence makes the number of switching of a wire be determined only by the number of logic highs of this wire.

In asynchronous circuits, there is no timing signal like clock, so concept of power exchanges to energy, which is the total power dissipation during a period of time. Also, the concept of power awareness is substituted by energy awareness.

CHAPTER THREE: IMPROVE POWER AWARENESS — 2-DIMENSIONAL PIPELINE GATING TECHNIQUE

3.1 Problem Definition

The power dissipation in CMOS circuit has three components: switching power, short-circuit power, and leakage power. Among these components, switching power is the dominant figure. When a node in circuit is switching, the load capacitance on this node will dissipate power due to the charging/discharging operation. If the switching activity could be reduced, the total power dissipation will be saved. For Boolean non-pipelined multipliers, starting from reset-to-zero state, low input precision calculation (like 0001×0001) dissipates much less power than high input precision calculation (like 1111×1111) because there are much less switching activities in internal nodes. Here the input precision is defined as the number of useful input bits (without padded 0's in high order bits) during the calculation. For example, the input precision of 0101 is 3, while the input precision of 1000 is 4. So Boolean non-pipelined multipliers are said to have natural power awareness to the changing of input precisions.

In pipelined multipliers, registers are important elements. Clock is connected to each register. In each clock cycle, a transition will occur on the clock input node of each register. This transition is independent of input data and will cause power dissipation even when the current input data of the register is the same as the current data output.

Since in deeply pipelined designs, the number of registers is much larger than that of other elements, these designs do not have the natural power awareness to the changing of input precision due to the large portion of power dissipated on clock input nodes. The power dissipation in deeply pipelined multipliers is nearly stable under different input precisions. Figure 8 shows the average power dissipation under different input precisions of a deeply pipelined 16-bit unsigned array multiplier.

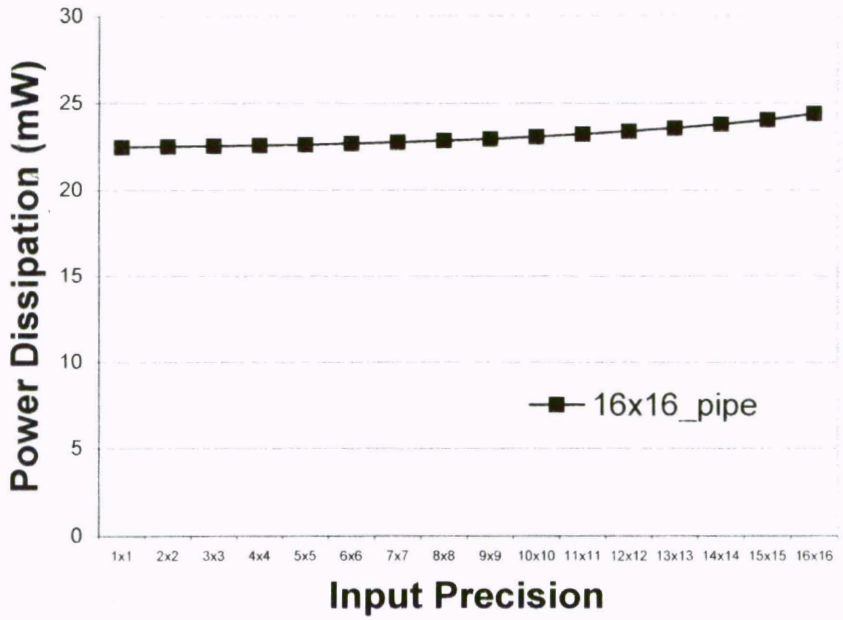


Figure 8. Power dissipation of a 16-bit pipelined array multiplier under different input precisions

For signed multipliers using 2's complement number representation, this problem is even worse. The Baugh-Wooley algorithm for signed multiplication is used as an example in this chapter. The equation of Baugh-Wooley algorithm for an $n \times n$ multiplication is shown in (9).

$$A\times B=-2^{2n-1}+\left(\overline{a_{n-1}}+\overline{b_{n-1}}+a_{n-1}b_{n-1}\right)\cdot 2^{2n-2}+\sum_{0}^{n-2}\sum_{0}^{n-2}a_i b_j 2^{i+j}+\left(a_{n-1}+b_{n-1}\right)\cdot 2^{n-1}$$
$$+\sum_{0}^{n-2}b_{n-1}\cdot\overline{a_i}\cdot 2^{i+n-1}+\sum_{0}^{n-2}a_{n-1}\cdot\overline{b_i}\cdot 2^{i+n-1} \qquad (9)$$

The tablet form of a 4×4 multiplication process using modified Baugh-Wooley algorithm is shown in Fig. 9. X and Y are 4-bit operands with the first bit as sign bit, and S is the 7-bit output. There are two major differences between Fig. 9 and 4×4 unsigned multiplication process shown in Fig. 10. One is that there are six inversed partial products in Fig. 9 but none in unsigned multiplication. The other is that there is an individual term "1" to be added to produce S_4 in Fig. 9 but none in Fig. 10.

X				X_3	X_2	X_1	X_0
Y				Y_3	Y_2	Y_1	Y_0
				$\overline{X_3Y_0}$	X_2Y_0	X_1Y_0	X_0Y_0
			$\overline{X_3Y_1}$	X_2Y_1	X_1Y_1	X_0Y_1	
		$\overline{X_3Y_2}$	X_2Y_2	X_1Y_2	X_0Y_2		
	X_3Y_3	$\overline{X_2Y_3}$	$\overline{X_1Y_3}$	$\overline{X_0Y_3}$			
			1				
S	S_6	S_5	S_4	S_3	S_2	S_1	S_0

Figure 9. 4×4 signed multiplication process

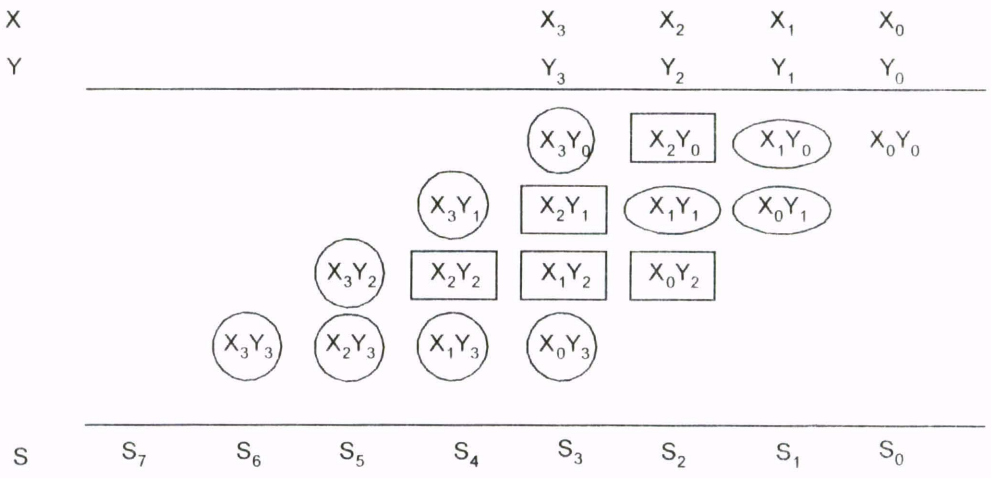


Figure 10. 4×4 unsigned multiplication process

These two differences bring reconfiguring problem for signed multipliers to operation under different input precisions. In unsigned multiplier, if two operands with less precision than the designed multiplier length to be multiplied, it will not cause any problem. For example, if using a 4×4 unsigned multiplier to calculate 101×011, just do it as 0101×0011. But in signed multiplier, there are some inversed terms inside. If these terms are not the corresponding partial products that should be inversed, incorrect result will occur. Also, the individual "1" also needs to appear on correct place. For example, if using the signed multiplier to multiply two signed operands 101 and 011, calculating them as 0101 and 0011 will cause wrong result. The reason is for a 3×3 signed multiplication process, X_2Y_0, X_2Y_1, X_1Y_2, and X_0Y_2 should be inversed and the individual "1" should appear in the column containing X_2Y_1. So unlike unsigned multiplier, signed multiplier cannot be automatically reconfigured for different input precisions.

Commonly used method to solve this problem is sign extension. Sign extension is to repeat the sign bit to fill the vacant high order bits in the operand until the length of the

operand matches the length of multiplier. For the example in last paragraph, instead of 0101×0011, 1101×0011 should be used. The problem of sign extension method is that the extended sign bits are totally redundant and will cause more power and delay. When the difference between the length of multiplier and the length of operands is large, for example, calculating signed number 11×11 using a 16×16 multiplier, a lot of extended bits are in logic high. These bits will cause significant redundant power dissipation. The use of sign extension will also make the signed multiplier lose the natural power awareness as that exists in unsigned multiplier.

3.2 2-Dimensional Pipeline Gating Technique

Kim et al., [46] introduced a clock gating method to design reconfigurable multiplier. This method is to selectively disable pipeline stages by gating clocks and to select correct results by multiplexers. Very little additional area cost is needed (only several AND2 gates and multiplexers) to implement this technique. Good power and latency saving can be achieved due to the reduced switching activities of registers in corresponding pipeline stages. The outputs of the multiplier are selected from different stages to ensure the correctness and obtain latency reduction. The basic idea of this method is shown in Fig. 11. This technique can be seen as 1-dimensional pipeline gating because it only considers gating clocks to unnecessary stages along data flow direction. As the computational width of multiplier growing from 4-bit, 8-bit, to 32-bit and 64-bit, 1-dimensional pipeline gating is far from enough.

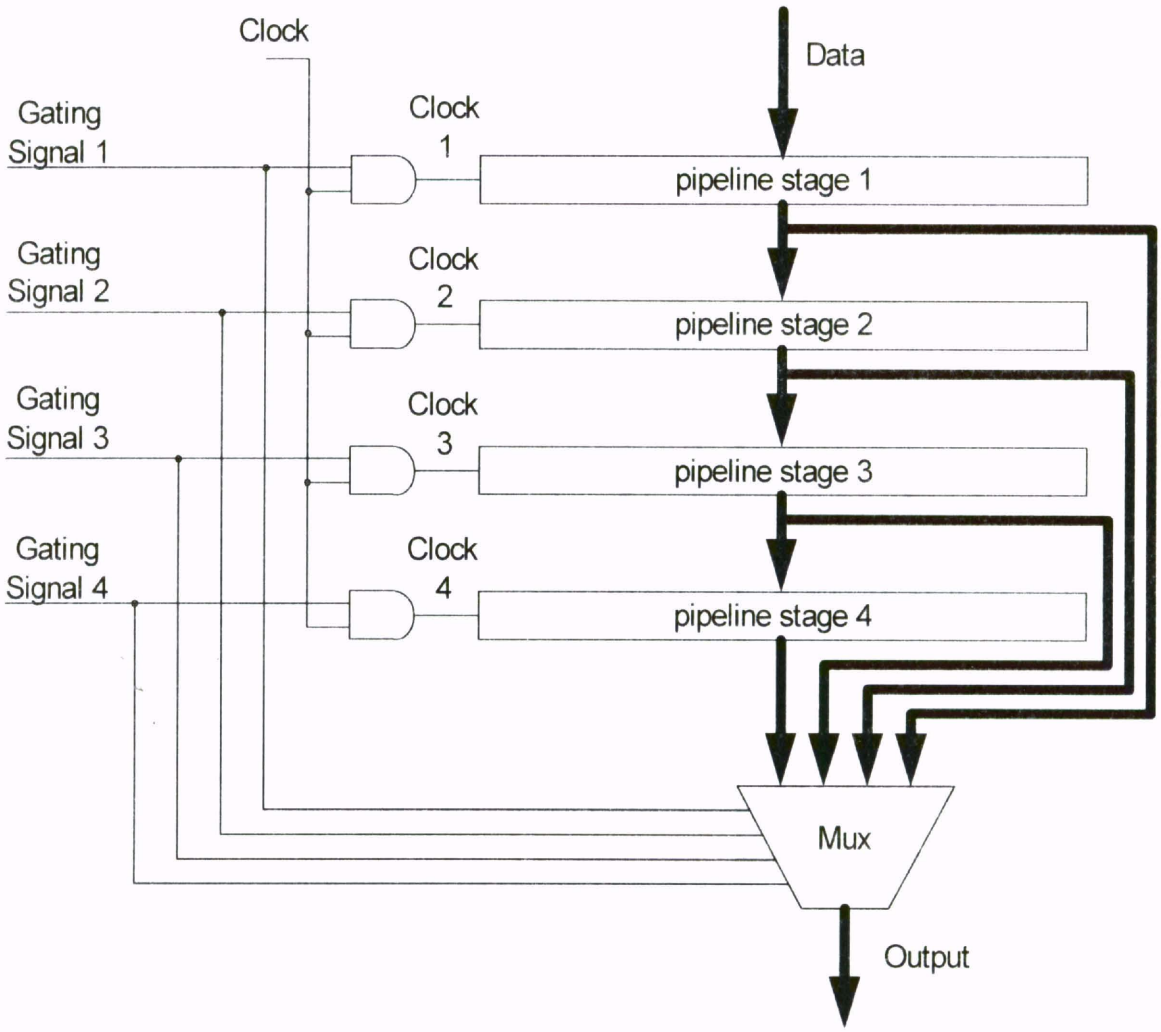


Figure 11. 1-Dimensional pipeline gating technique

2-D pipeline gating is to gate clock to the registers in both vertical direction (data flow direction in pipeline) and horizontal direction (within each pipeline stage), while 1-D pipeline gating technique gates clock in vertical direction only. The principle of 2-D pipeline gating technique is shown in Fig. 12.

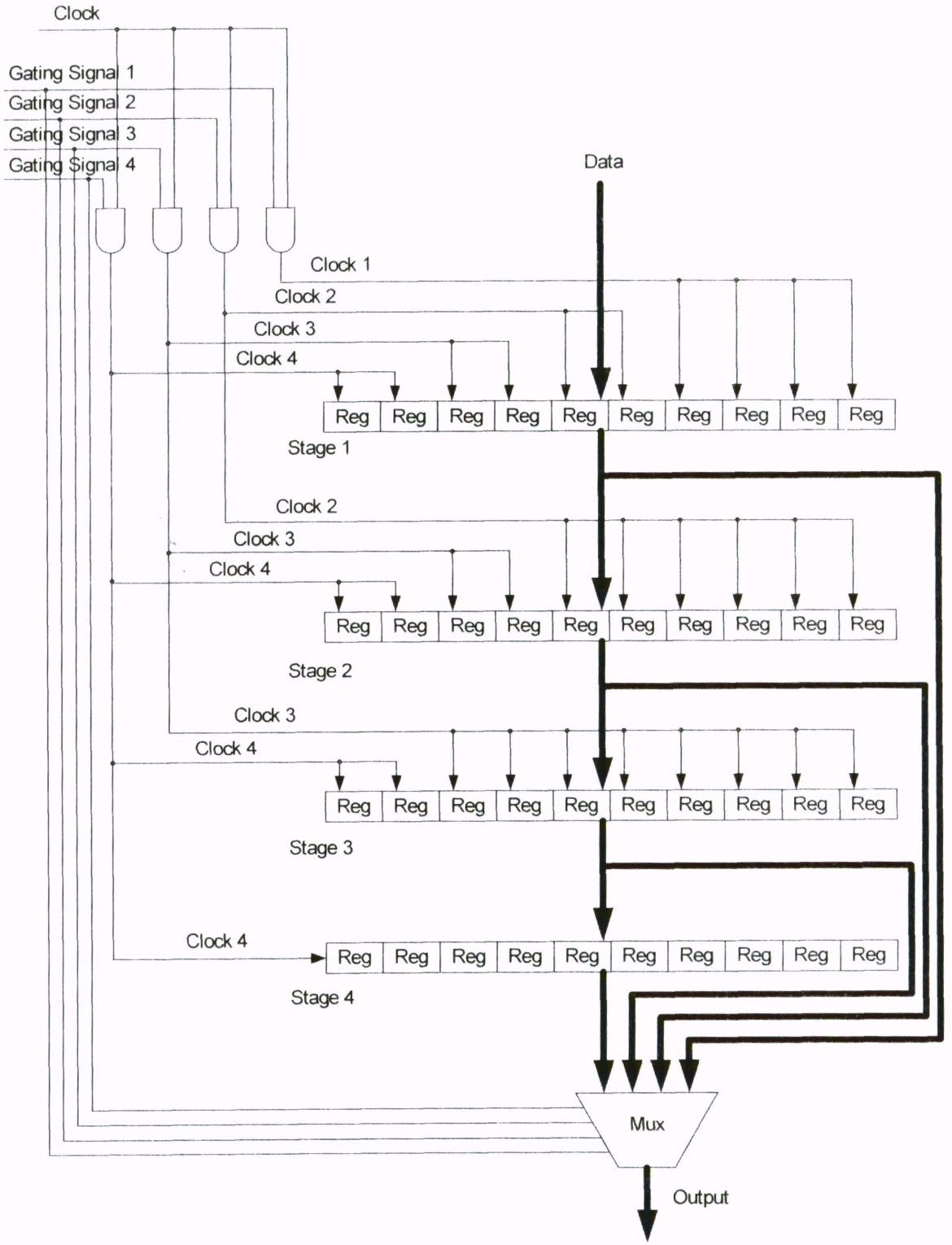


Figure 12. 2-Dimensional pipeline gating technique

In the 1-D pipeline gating scheme shown in Fig. 11, the system clock is gated by different gating signals to generate sub-clocks. Each sub-clock is connected to one pipeline stage and drives all registers in that stage. If under a certain case the results could come directly from stage 3, then the *Gating Signal 4* is set effective and *Clock 4* is disabled. The output of register 3 is then bypassed through a multiplexer, which is also controlled by the clock gating signals, to the system output. Since the *Clock 4* is disabled, the total number of switching is reduced. Also, since the system output now comes from stage 3 instead of stage 4, the pipeline latency is reduced.

In a real pipeline, the data going through a register in a certain pipeline stage is most likely to correlate with the data going through the register in the previous stage. So if under a certain case one pipeline stage could be disabled, some of the registers in its previous stage may also be redundant and could be disabled too. This happens especially in such pipelines in which only some data are processed in this stage, others are just passed to the next stage. Computer arithmetic circuits like multipliers and adders always contain such pipelines [47]. By applying 2-D pipeline gating technique to these circuits, significant power saving can be achieved.

In the 2-D pipeline gating scheme shown in Fig. 12, when under a certain case pipeline stage 4 could be disabled, some of the registers in previous stages (the first two registers in stage 1, 2, and 3) could also be disabled if the data going through them was to be processed only in stage 4 thus is no longer useful. These registers can be disabled by using *Clock 4* as their clock inputs. For the same reason, if stage 3 needs to be disabled, the third and fourth registers in stage 1 and 2 could also be disabled. The total number of transition is further reduced compared to that in 1-D pipeline gating system. As the

number of registers in each stage as well as the total number of stages in the pipeline (pipeline depth) increase, this further benefit becomes more and more significant. As shown later in this chapter, the 16-bit unsigned multiplier using 2-D pipeline gating has more than 54% power saving over the same multiplier using 1-D technique. And this number is 55.6% for signed multiplier.

3.3 Power-aware Unsigned Array Multiplier Design

To design power-aware pipelined multiplier using 2-D pipeline gating technique, firstly the multiplication process should be examined. The 4×4 unsigned multiplication process is shown in Fig. 10.

In Fig. 10, X and Y are inputs while S is the output. When the input precision is 4, for example, calculating 1111×1111, S is generated based on all inner partial products. If the input precision is 3, for example, calculating 0111×0111, the partial products containing X_3 or Y_3 are all zero (these products are enclosed by a circle in Fig. 10), and S only has six digits instead of eight. From a reset-to-zero state, there is no need to let registers propagate these zeros because the reset state of register is zero. So clocks connected to these registers can be disabled. If the input precision is 2, for example, calculating 0011×0011, the partial products containing X_2 or Y_2 (the ones enclosed by a rectangular in Fig. 3) can also be disabled. If the input precision is 1 as 0001×0001, the partial products enclosed by an ellipse in Fig. 10 containing X_1 or Y_1 can be disabled. As the length of output S decreases, the number of necessary pipeline stages is also reduced. The circuit structure of a 4-bit pipelined unsigned array multiplier using 2-D pipeline gating technique is shown in Fig. 13.

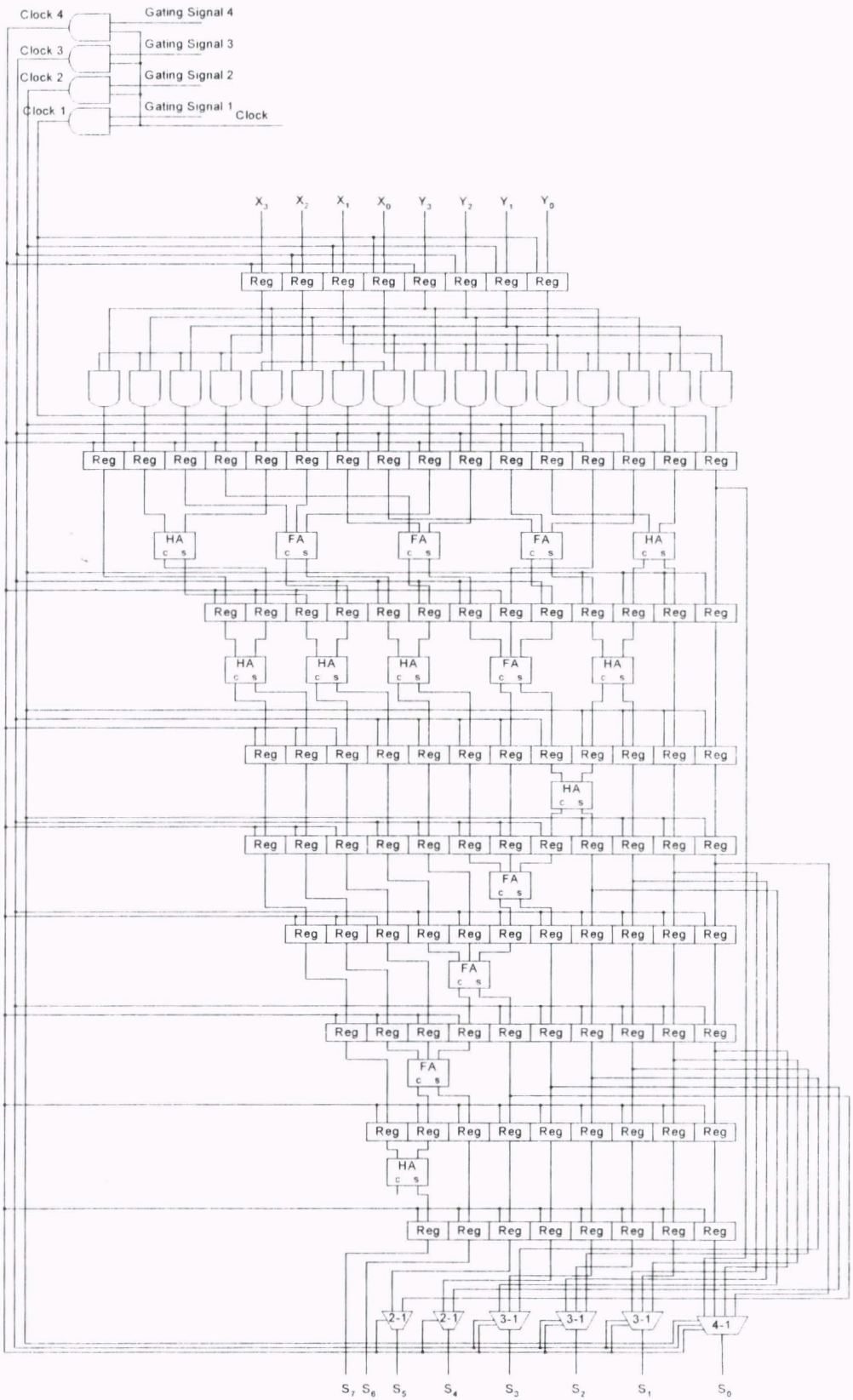


Figure 13. A 4-bit pipelined unsigned array multiplier using 2-D pipeline gating

In Fig. 13, "HA" represents half adder; "FA" represents full adder; "Reg" represents register; "n-1" represents n-to-1 multiplexer. Current input precision information is provided through four gating signals from CPU. These signals are combined with system clock to generate four sub-clocks, which are connected to the corresponding registers in all pipeline stages. Under a certain input precision, one or more sub-clocks may be disabled. The registers connected to these sub-clocks will not function during the calculation. The multiplexers select correct outputs from corresponding stages. For example, while performing 0001× 0001, only S_0 has useful value. This value is selected from the stage right after the AND matrix. Except for this register and the two registers in the first stage for X_0 and Y_0, all other registers do not function because their clocks have been disabled. The power dissipation is reduced significantly. The output S_0 is from the first stage after the AND matrix instead of the eighth one, thus the pipeline latency has also been reduced by a factor of eight.

The detection of current input precision is a typical interrupt-response scheme for a CPU. For example, when the user of digital camera pushes the button to reduce the resolution, an interrupt is sent to the CPU. Then CPU reads the corresponding register and sets up the clock gating signals based on the register value. So the additional area cost is very low, just a few AND gates and some multiplexers are needed. The clock gating signals are also used as the control signals of these multiplexers.

Based on the discussion above, a set of unsigned array multipliers were designed. The computation lengths of these multipliers are 4-bit, 8-bit, and 16-bit, respectively. Both 1-D and 2-D pipeline gating techniques have been applied to each multiplier. These nine multipliers were synthesized by Synopsys Design Analyzer and simulated in

Powermill. During the simulation, the multipliers were given data in different input precisions. The power dissipation were recorded and compared. The simulation result comparisons are shown in Fig. 14 to 17.

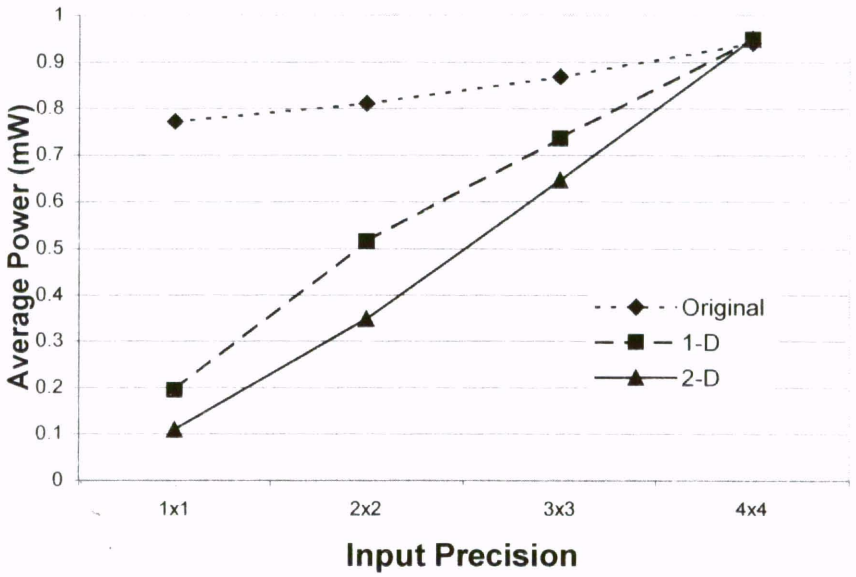


Figure 14. Average power comparison of 4-bit unsigned multipliers

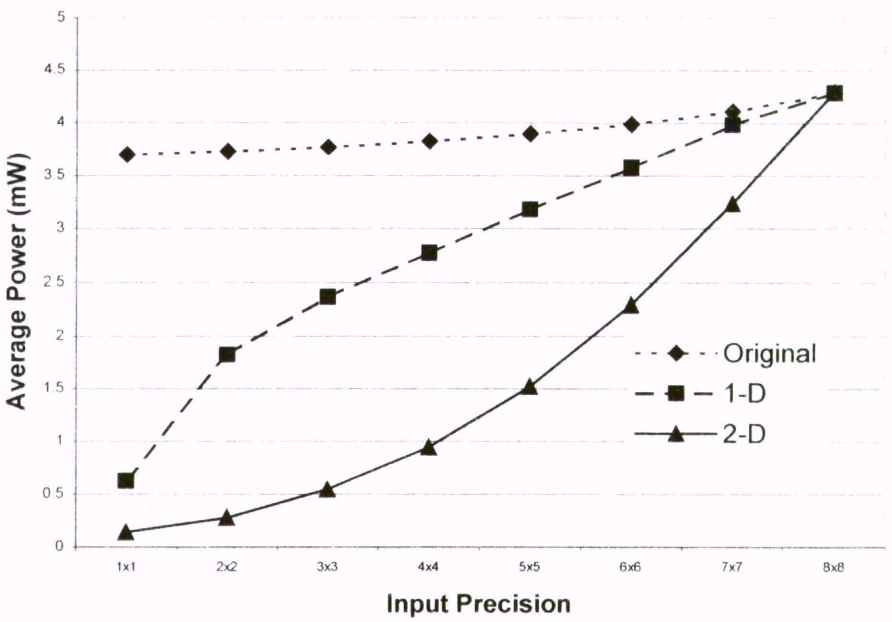


Figure 15. Average power comparison of 8-bit unsigned multipliers

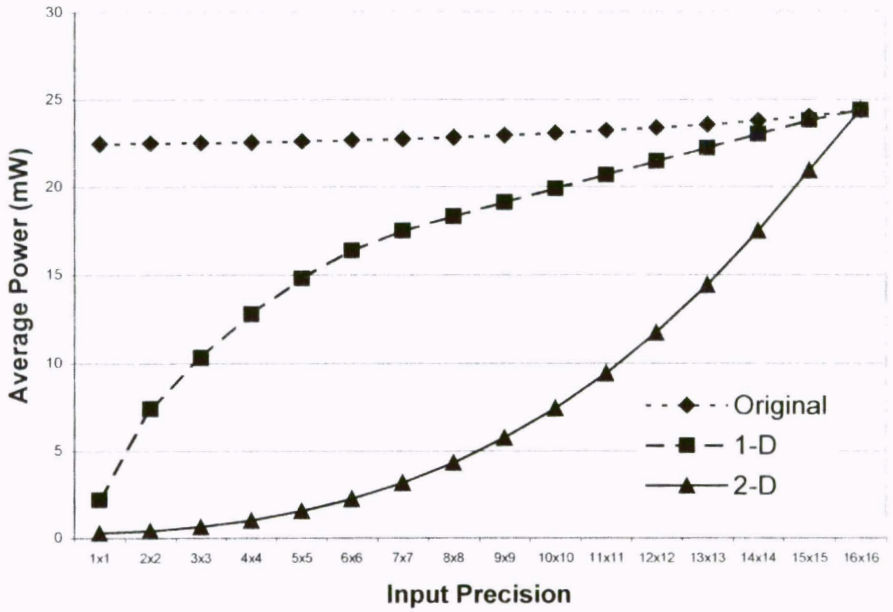


Figure 16. Average power comparison of 16-bit unsigned multipliers

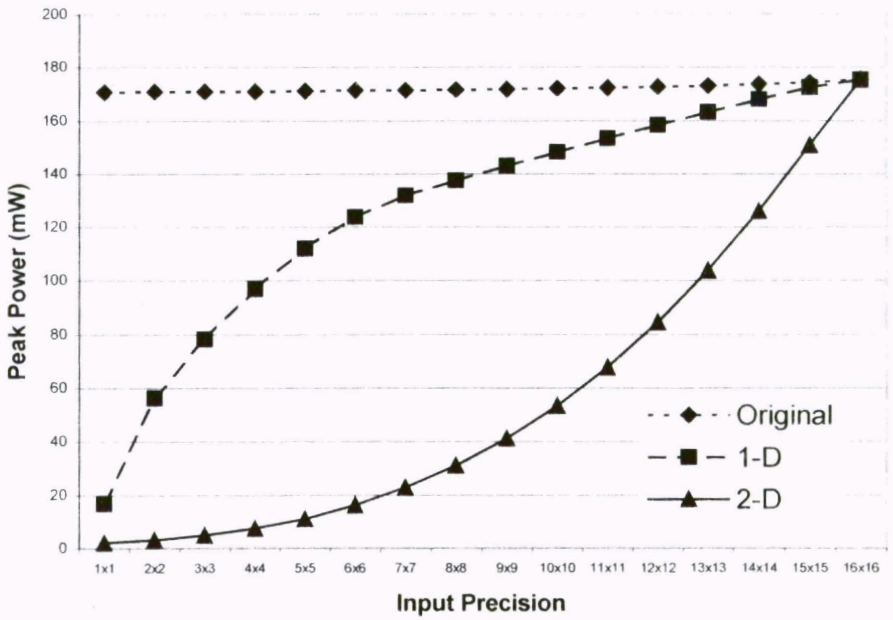


Figure 17. Peak power comparison of 16-bit unsigned multipliers

In Fig. 14 to 17, "Original" represents the simulation data of the unchanged pipelined designs; "1-D" and "2-D" represent the simulation data of the designs using 1-D and 2-D pipeline gating techniques, respectively.

From these figures, several observations are made:

1. Among all three multipliers, the designs using 1-D and 2-D pipeline gating techniques have lower power dissipations compared to the original designs under different input precision.
2. Among all three multipliers, the designs using 2-D pipeline gating techniques show significant power savings over the corresponding designs using 1-D pipeline gating technique. This advantage is not large in 4-bit multiplier (14.3% under equal input precision probability), but becomes much greater in 8-bit multiplier (41.5% under equal input precision probability), and is quite significant in 16-bit multiplier (54.4% under equal input precision probability). As shown in Fig. 14 to 17, the data of designs using 1-D pipeline gating technique show convex curves while that of designs using 2-D pipeline gating technique show concave curves. The reason for this difference is that as the length of multiplier goes up, the number of registers in horizontal direction as well as in vertical direction increases sharply. 1-D pipeline gating technique only deals with the vertical pipeline stage increment, while 2-D pipeline gating technique controls the registers in both directions. For example, in a 64×64 array multiplier, there will be 65536 registers in the pipeline stage right after the AND matrix. When this multiplier is performing 1×1 multiplication, only the last register containing X_0Y_0 has useful value. Since 1-D technique only gates the

clocks to the rest of the stages, these 65536 registers will all be functioning. In 2-D pipeline gating technique, on the other hand, only the register containing useful data will be functioning; all other 65535 registers are disabled. Actually, the largest difference between these two techniques occurs when the current input precision is half the designed precision. Under this case, there are lots of registers in middle pipeline stages that are propagating redundant zeros. 1-D technique cannot deal with them. But 2-D pipeline gating technique has the ability to disable them accurately.

3. The overhead of implementing 1-D and 2-D techniques are the same. It is very small (0.03% in 16-bit multiplier).
4. Peak power dissipation affects the system reliability in operating under power constraints. 1-D and 2-D pipeline gating techniques both have the ability to reduce system peak power dissipation. But the same as average power dissipation, 2-D technique has great advantage over 1-D technique under different input precisions.

The pipeline latency reduction of the designs using 1-D and 2-D pipeline gating techniques is the same. The comparison data of latency saving as well as other data are shown in Table 2.

Table 2. Data comparison table of unsigned multipliers

Multiplier length		4-bit	8-bit	16-bit
Under equal input precision probability	**Average power saving**			
	1-D vs. Original	29.5%	27.8%	25.7%
	2-D vs. Original	39.6%	57.8%	66.2%
	2-D vs. 1D	14.3%	41.5%	54.4%
	Peak power saving			
	1-D vs. Original	31.0%	28.7%	26.1%
	2-D vs. Original	43.4%	60.0%	67.3%
	2-D vs. 1D	18.0%	43.9%	55.8%
	Latency reduction			
	1-D vs. Original	36.1%	39.1%	44.1%
	2-D vs. Original	36.1%	39.1%	44.1%
Overhead	1-D	0.01%	0.02%	0.03%
	2-D	0.01%	0.02%	0.03%

3.4 Power-aware Signed Array Multiplier Design

To solve the sign extension problem and make the signed multiplier have good power awareness, several important analysis and modifications beside the 2-D pipeline gating technique have to be made. A selective method is proposed in this section. The pipeline stage right after the AND matrix stage of a 4×4 power-aware signed multiplier is shown in Fig. 18.

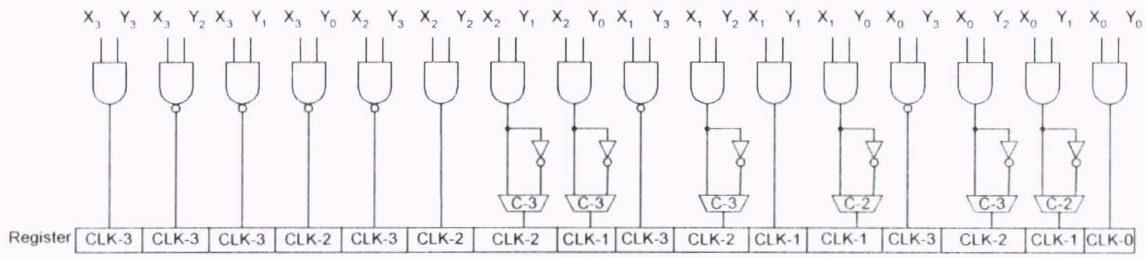


Figure 18. The first pipeline stage after AND matrix in 4×4 power-aware signed multiplier

In Fig. 18 the partial products X_3Y_0, X_3Y_1, X_3Y_2, X_0Y_3, X_1Y_3, and X_2Y_3 are inversed by connecting to NAND gates. Another input (not shown in Fig. 18) called *const_in* is added to the proper adder as the individual "1". Inner products X_2Y_1, X_1Y_2, X_0Y_2, X_2Y_1, X_0Y_1, and X_1Y_0 are connected to 2-to-1 multiplexers with their inversions. These multiplexers are controlled by different control signals indicating the current input precision. These signals are just as the clock gating signals issued by CPU. The outputs of these multiplexers along with all other outputs of AND/NAND gates are connected to the registers forming next pipeline stage. These registers, just as designing power-aware unsigned array multipliers, are connected to different gated clocks controlled by clock gating signals based on current input precision.

When current input precision is 4×4, all multiplexers are switched to the non-inversed data; all four types of clocks are enabled; the *const_in* bit is set to logic high. Then the multiplier is able to perform 4×4 signed multiplication as shown in Fig. 9.

When current input precision is 3×3, the multiplexers for X_2Y_1, X_1Y_2, X_0Y_2, and X_2Y_1 are switched to their inversed data; *CLK-3* is disabled; the *const_in* bit is set to logic low. Note that the clock connected to the output of NAND gate whose input is X_3Y_0 is

CLK-2, not *CLK-3*. Since X_3 and Y_3 are all zero, this NAND gate will generate logic high. This "1" becomes the individual "1" needed for 3×3 multiplication.

When current input precision is 2×2, the multiplexers are all switched to the inversed data; both *CLK-2* and *CLK-3* are disabled; the *const_in* bit is still logic low. For the same reason, the clock connected to the output of NAND gate whose input is X_2Y_0 is *CLK-1*, not *CLK-2*. This bit becomes the individual "1" for 2×2 multiplication. Note, there is no 1×1 multiplication for signed multiplier because there has to be a sign bit.

By applying the modifications above, the 4×4 pipelined signed multiplier is able to perform 3×3 and 2×2 multiplication without sign extension. During 3×3 and 2×2 multiplication process, the gated registers will not function, so that the power dissipation is saved. Also, the redundant power dissipation caused by sign extension is avoided. The same as in applying 1-D or 2-D technique on unsigned multipliers; the output bits can be selected from different pipeline stages prior to the last stage. So the pipeline latency can also be reduced.

Based on the technique described above, just as the testing scheme of unsigned multiplier, nine pipelined signed array multipliers with lengths of 4-, 8-, and 16-bit are designed as original architecture, the designs using 1-D pipeline gating technique, and the power-aware designs using 2-D pipeline gating technique. All designs are also synthesized by Synopsys Design Analyzer, and then simulated in Powermill. The results comparisons are shown in Fig. 19 to 22.

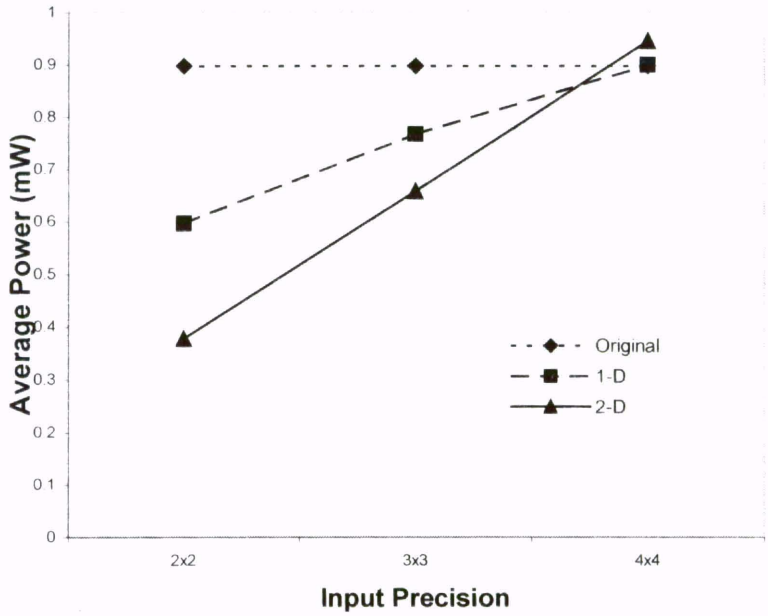


Figure 19. Average power comparison of 4-bit signed multipliers

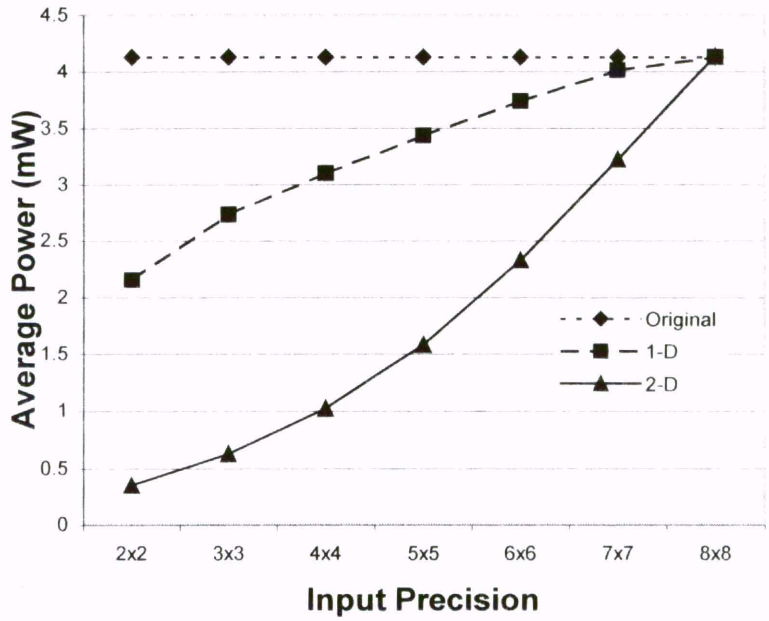


Figure 20. Average power comparison of 8-bit signed multipliers

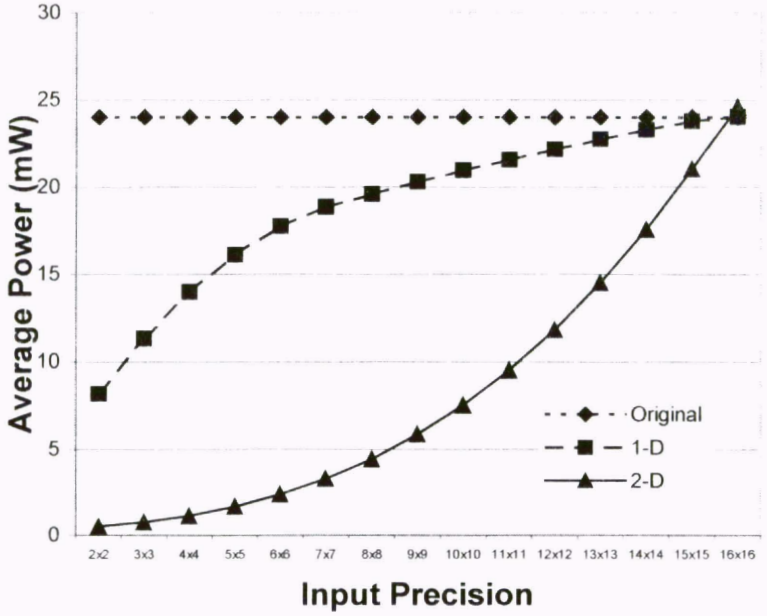


Figure 21. Average power comparison of 16-bit signed multipliers

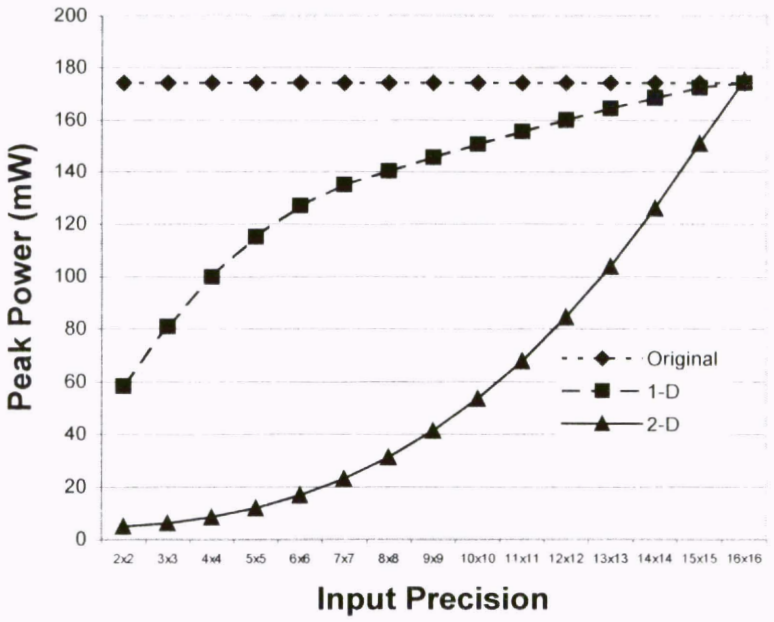


Figure 22. Peak power comparison of 16-bit signed multipliers

From Fig. 19 to 22 several observations could be made:

1. The same as in unsigned multiplier results, the 2-D designs have great advantage over the other two groups in terms of average and peak power dissipation. There are two reasons for this difference: one is the same as in unsigned multiplier design, which is, 2-D technique not only gates the redundant pipeline stages like 1-D technique does, but also disables the unused registers within the useful pipeline stages. The other reason is the use of sign extension brings more switching to 1-D designs. But the 2-D power-aware designs do not have this problem. In 16-bit multiplier, the 2-D design has 55.6% average power saving and 55.8% peak power saving over the design using 1-D technique.
2. The overheads of the 2-D designs are a little larger than that in unsigned multiplier. But they are still very small, only 0.23% in 16-bit design.

The latency reductions of the 2-D designs are the same as those designs using 1-D technique. The comparison in data form is shown in Table 3.

Table 3. Data comparison table of signed multipliers

Multiplier length		**4-bit**	**8-bit**	**16-bit**
Under equal input precision probability	**Average power saving**			
	1-D vs. Original	16.0%	19.4%	21.1%
	2-D vs. Original	26.5%	55.3%	65.0%
	2-D vs. 1D	12.5%	44.6%	55.6%
	Peak power saving			
	1-D vs. Original	17.5%	20.6%	21.7%
	2-D vs. Original	28.4%	55.6%	65.4%
	2-D vs. 1D	13.1%	44.0%	55.8%
	Latency reduction			
	1-D vs. Original	36.1%	39.1%	44.1%
	2-D vs. Original	36.1%	39.1%	44.1%
Overhead	**1-D**	0.15%	0.03%	0.01%
	2-D	0.52%	0.10%	0.23%

3.5 Application of Power-aware Multipliers on FIR Filter Design

As an application of power-aware multipliers, a high-throughput, power-aware FIR filter design method is introduced in this section. FIR filters are essential elements in DSP systems. There are different implementations of FIR filters. To shorten the critical path in order to achieve high throughput, Data-Broadcast structure is used in this section [48]. A 3-tap Data-Broadcast FIR filter is shown in Fig. 23.

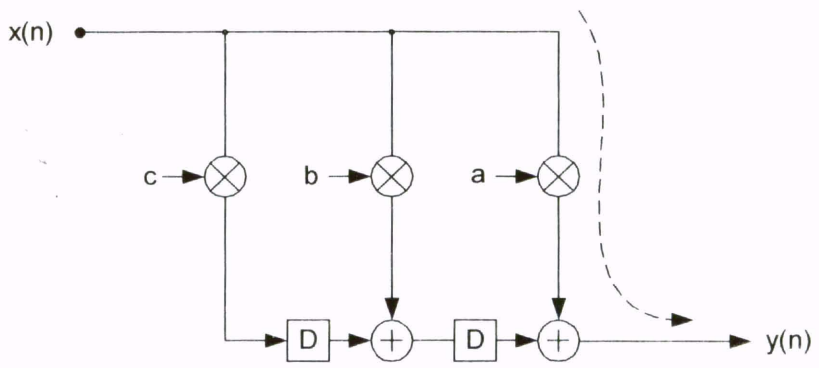


Figure 23. Data-broadcast structure of FIR filter

There are three multipliers and two adders. The input-output relationship is shown in (10).

$$y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2) \tag{10}$$

The dashed line in Figure 23 shows the critical path. The length of this critical path is $T_M + T_A$, where T_M is the time taken for multiplication and T_A is the time taken for addition. The period of operating clock must be longer than this length. This results in a very low clock rate. If this FIR is used in a real-time application, the sampling frequency, f_{sample}, must be less than the operating frequency of this FIR filter, that is

$$f_{sample} \leq \frac{1}{T_M + T_A} \tag{11}$$

To improve the throughput of the FIR filter, one commonly used method is to pipeline the multipliers. Since the multiplication time T_M is usually much larger than the addition time T_A, much shorter critical path length can be achieved by carefully balancing the pipeline stages. Figure 24 shows a pipelining scheme for FIR filter.

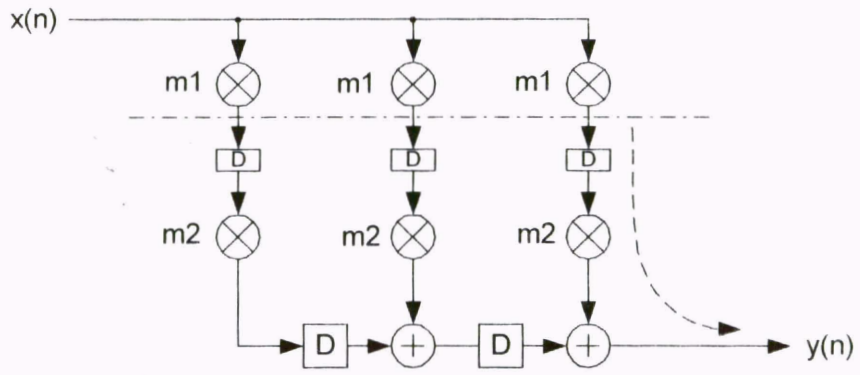


Figure 24. Improve the throughput of FIR by pipelining multipliers

In Fig. 24, each multiplier in Fig. 16 is divided into two pipeline stages. A series of registers is added between the two sub-multipliers. The time taken by each stage of the multiplier is denoted by T_{M1} and T_{M2}, respectively; and the delay time in the added registers is denoted by T_{DR}. If the pipeline is perfectly balanced so that $T_{M1} = T_{DR} + T_{M2} + T_A$, the critical path shown as dashed line in Fig. 24 is much shorter than that of Fig. 23.

By pipelining multipliers can only achieve limited throughput improvement. Assume the number of pipeline stages that the multipliers in Fig. 24 can be divided to approaches infinity; the slowest stage will contain the adder and a very small part of

multiplier. So the length of critical path is approaching T_A. When the word length of input data and coefficients is short, T_A is small enough for the FIR to operate in high sampling frequency. In recent years, the word length of FIR filter has been growing from 8-bit, 16-bit, up to 32-bit and 64-bit. Under long word length condition, addition also takes significant time. Instead of pipelined multipliers, adders become bottleneck in these FIR filters under such conditions.

To further improve throughput of FIR filters, the critical path in addition process needs to be shortened too. So adders, as well as multipliers, need to be pipelined. Pipelining one adder changes the timing relationship between the two inputs of the next adder. Unlike pipelining multipliers, which doesn't change the relative timing sequence between adder inputs, pipelining adders just likes adding delay elements to the paths between adders. So additional delay elements need to be added between next adder and its corresponding multiplier. The goal is to maintain the timing difference between the two inputs of the adder as one clock cycle. The revised FIR filter structure is shown in Fig. 25.

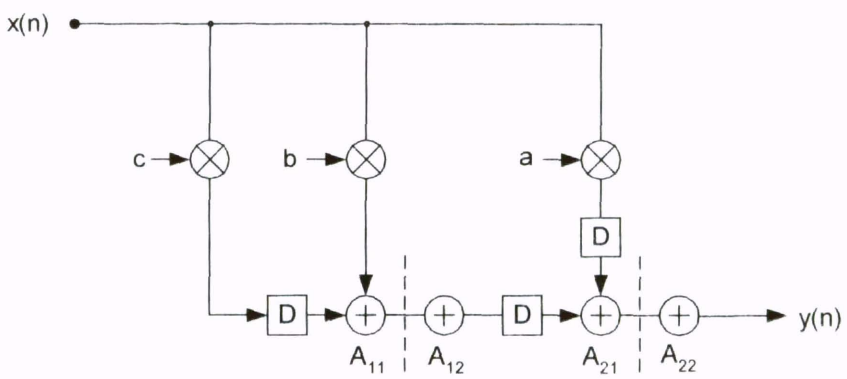


Figure 25. Revised adder pipelining scheme

For a balanced pipeline design, each pipeline stage takes almost the same calculation time. By fine-grain pipelining multipliers and adders, very high throughput can be achieved. For an N-tap FIR filter, if the multipliers are divided to M pipeline stages, and the adders are divided to P pipeline stages, the FIR filter structure is shown in Fig. 26.

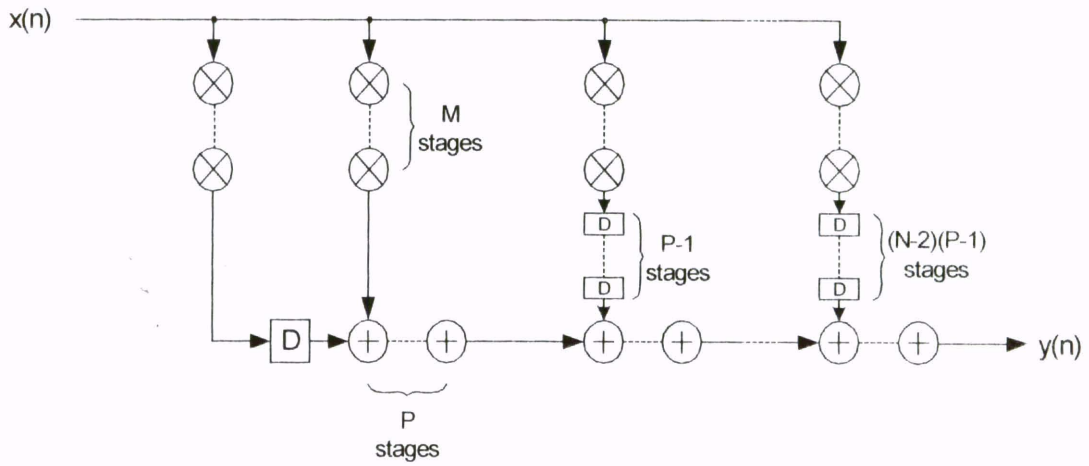


Figure 26. N-tap FIR filter structure by pipelining multipliers and adders

Along with the achieved high throughput, pipelining multipliers and adders also causes two problems. Firstly the power dissipation will become larger because some registers are added between the pipeline stages; secondly the total latency from the input to output also becomes larger because the pipelined addition paths are longer. To solve these problems, 2-D pipeline gating technique is used to design power-aware multipliers and adders. These elements are able to scale their power and latency with the changing of input precision. To maintain the correctness of the calculation, 2-D pipeline gating technique needs also be implemented on the additional delay elements to select output

from the corresponding pipeline stage thus keep the timing relationship. Based on these discussions, a set of 4-tap FIR were designed and tested. The word length of input data and coefficients are all 16-bit. So the multipliers are 16×16 and the adders are 32×32. These designs are synthesized by Synopsys Design Compiler and then simulated in Powermill. The simulation results are shown in Fig. 27-29.

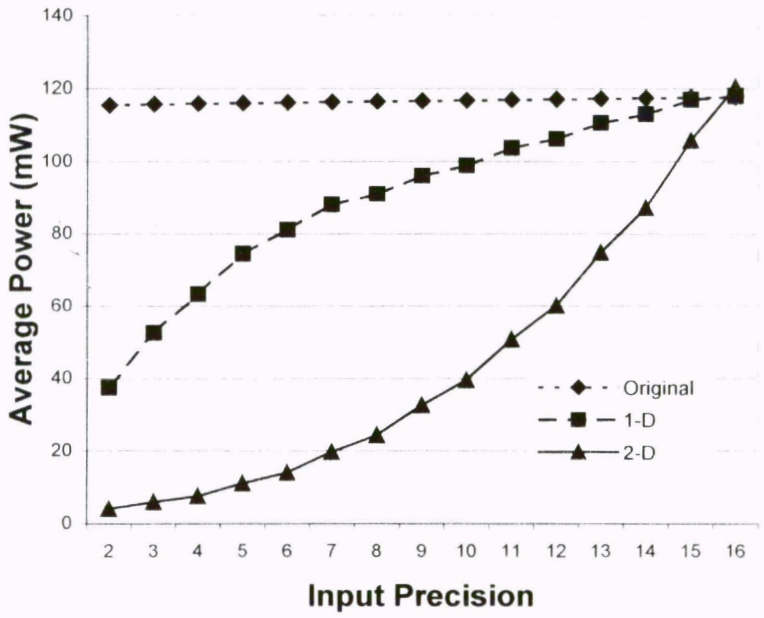


Figure 27. Average power dissipation of the designed FIR filters

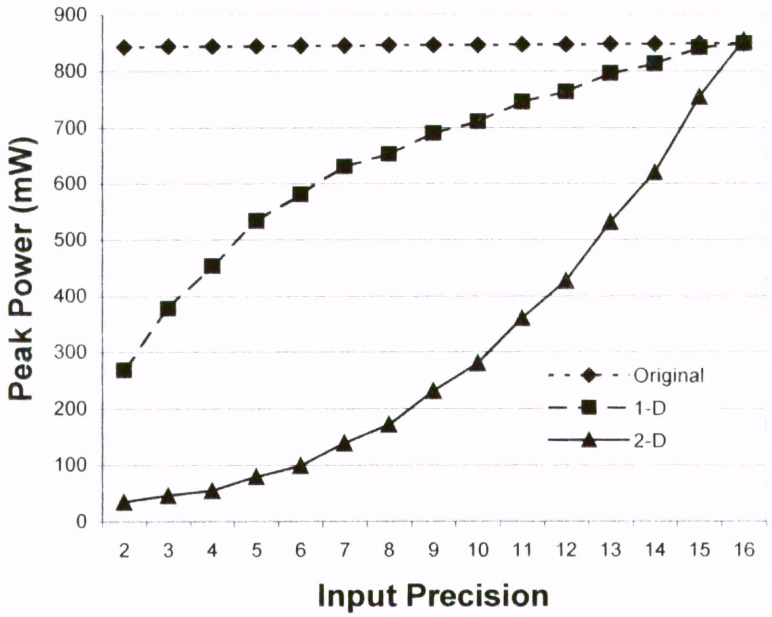


Figure 28. Peak power dissipation of the designed FIR filters

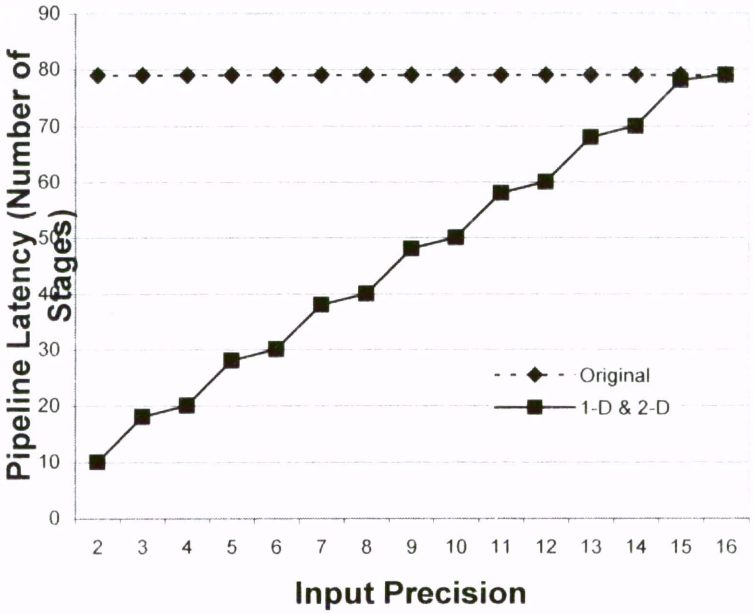


Figure 29. Normalized pipeline latency of the designed FIR filters

Several discussions are listed as below:

1. The throughput of the pipeline is determined by the slowest stage. Since multipliers and adders are fine-grain pipelined, the delay in each pipeline stage is very small. The technology used in synthesis process is 0.24μm static CMOS logic. Simulation results show that the designed FIR filter is able to work under 1.25GHz clock rate. If using dynamic gate or transistors of smaller channel length, even higher throughput is expected.
2. The same as in multipliers design, the average power dissipation as well as peak power dissipation are significantly reduced by applying 2-D pipeline gating technique. The power reduction rate of 2-D design is much better than that of 1-D design.

On reducing pipeline latency, 1-D and 2-D gating techniques have the same rate of advantage. By selecting the correct outputs from corresponding stages, the total pipeline latency is significantly reduced.

CHAPTER FOUR: IMPROVE ENERGY AWARENESS OF NCL CIRCUITS

4.1 Energy Aware Problem In NCL Circuits

In CMOS circuit, if ignore leakage and short-circuit energy, the energy dissipation is proportional to the number of switching. A well-known formula for energy consumption is

$$E = \sum_{gates} \frac{1}{2} \cdot C_{gate\text{-}load} \cdot V_{dd}^2 \cdot Num \qquad (12)$$

where C is the gate load capacitance, V_{dd} is the supply voltage, and Num is the number of gate switches [49].

In Boolean logic, there is no Null state. All Data states are adjacent to each other. So if the current data value is the same as the previous one, there is no transition. For example, in a multiplication cycle, starting from reset-to-zero state, if one or more left most bits of the two multiplicands are zero, the energy dissipation will be smaller than that of all bits are one.

But in NCL multi-rail encoding logic, the situation is quite different. Figure 30 shows the energy dissipation versus different input precisions of an 8×8 NCL dual-rail multiplier.

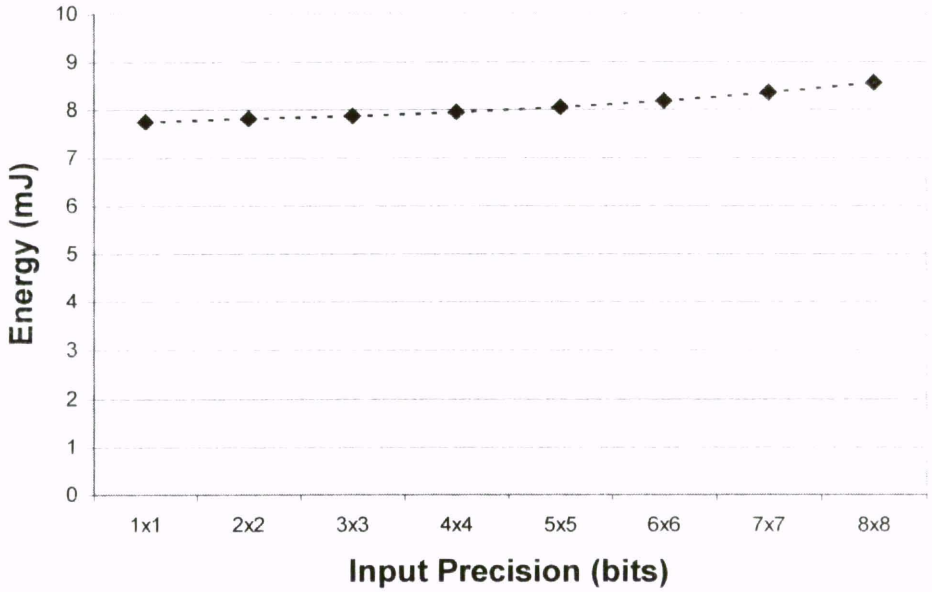


Figure 30. Energy dissipation of 8×8 dual-rail multiplier in different input precisions

From Figure 30 it is clear that the energy dissipated in different input precisions are almost the same. The reason is that in NCL multi-rail encoding, Data states are separated by Null states so that no two adjacent states could be the same. And from the encoding truth table, Data 0 is the same as Data 1 in terms of switching activity. From a Null state to next Data state, no matter Data 0 or 1, one wire will switch to logic high. When next Null state comes, this wire will switch to logic low. So in 00000001×00000001 calculation, although all other bits in the output are Data 0's except the least significant one, the total number of switching is the same as that in 11111111×11111111. This is the major problem to design energy-aware circuits in NCL multi-rail encoding logic.

4.2 Signal Bypassing & Insertion

4.2.1 Signal Bypassing & Insertion Technique

In order to make energy-aware design, the number of switching must be reduced with the changing of input precisions. To reduce the number of switching in multi-rail encoding circuit, some adjacent state pairs must be the same. Since for any Data state, its previous and following states are all Null state, if Null could be used as Data 0 in some Data states, the number of switching could be reduced. For example, in a multiplication cycle, used Null instead of Data 0 in the unused left most bits of the multiplicands. More specifically, for a 3×3 multiplier, while calculating 011×011, this multiplier is actually working as a 2×2 multiplier. If N11×N11 could be used, where N represents Null, the energy dissipation should be reduced. Unfortunately, this simple thought does not work in most of the cases. Figure 31 demonstrates the problem in multiplication.

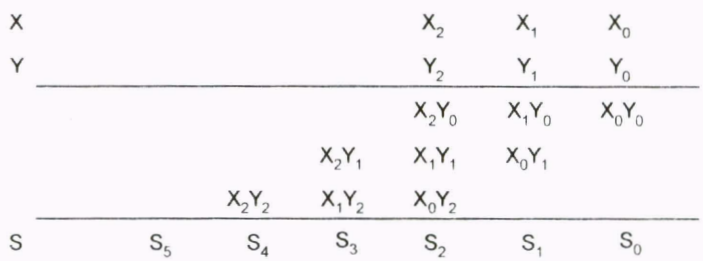


Figure 31. 3×3 multiplication

Figure 31 shows a 3×3 multiplication. X and Y are inputs while S is the output. Now Null is given to X_2 and Y_2 while Data 1 is given to the rest of input bits. Then X_2Y_0 and X_0Y_2 will also become Null instead of Data 0. So S_2, the addition of X_2Y_0, X_0Y_2 and

X_1Y_1, will become Null instead of the correct result. Also, since X_2Y_1 and X_1Y_2 are both Null, S_3 will also become Null instead of the correct result. So the result is incorrect.

The reason for the failure is that in a Data state, when the inputs of a functional block contain both Data and Null, the output of this block is most likely to be incorrect. But since these Null inputs suppose be Data 0 in regular calculation, the output of these blocks should be Data.

To solve this problem while still using Null as part of the inputs, a technique named Signal Bypassing & Insertion is proposed in this section. Signal Bypassing is to let the Data input of a block "bypass" this block to the output when the other inputs of this block are all Nulls. Signal Insertion is to insert a Data 0 as the input of a block to replace a Null input when at least two of other inputs of this block are Data. Signal Bypassing & Insertion guarantee the correctness of circuit operation while reducing the number of switching. Figure 32 illustrates the implementation of this technique to 3×3 multiplier.

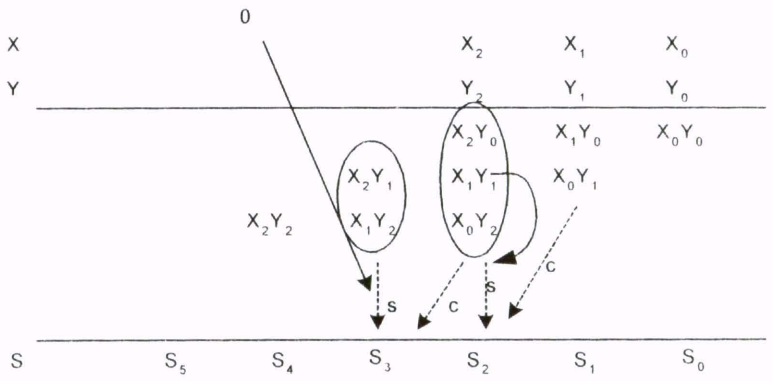


Figure 32. Signal bypassing & insertion illustration

In Figure 32, to perform N11×N11, since X_2Y_0 and X_0Y_2 both should be zero in 011×011, X_1Y_1 could be "bypassed" to the sum of this adder to be added up with the carry coming from S_1 to calculate S_2. Also, since X_2Y_1 and X_1Y_2 both should be zero in 011×011, a zero could be "inserted" to the sum of this adder and to be added up with the carry coming from S_2 to calculate the correct S_3. After these two adjustments, N11×N11 can be used as 011×011 to saving energy while maintaining the correctness of the calculation. The circuit implementation is shown in Figure 33.

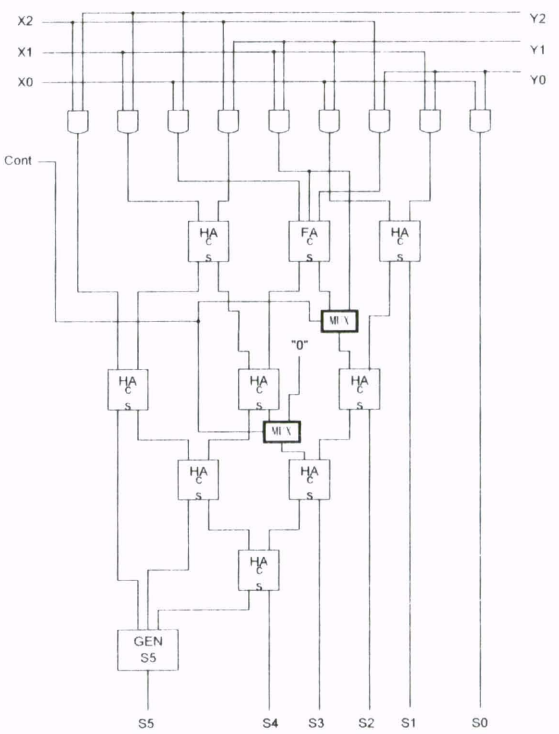


Figure 33. Implementation of Signal Bypassing & Insertion on 3×3 multiplier

In Figure 33, HA means half adder; FA means full adder; GEN_S5 is a functional block to calculate S_5. Two 2-to-1 multiplexers are used to interleave data paths from 3×3

to 2×2 operation. One control signal is added to control the operation of multiplexers. This control signal comes from the central control unit of the whole device. This unit is able to "know" the current input precision and set up the control. So the device prior to this multiplier is able to provide Nulls to unused left most bits of the multiplicands and the following device is able to select the useful bits from the result of this multiplier. Signal Bypassing & Insertion is a gate-level energy-aware design technique that can be used to design energy-aware NCL multi-rail encoding circuits.

4.2.2 Design Examples Of Signal Bypassing & Insertion

4.2.2.1 Energy-aware Pipeline Components

Pipeline structure is widely used in modern digital circuit design. The basic pipeline components in NCL circuit include registers and hand-shaking signal generators. For a pipelined circuit to be energy-aware, these pipeline components must be controlled by the central energy-aware control unit as well as the combinational logic circuits between the registers. The basic structure of NCL pipeline components is shown in Figure 34.

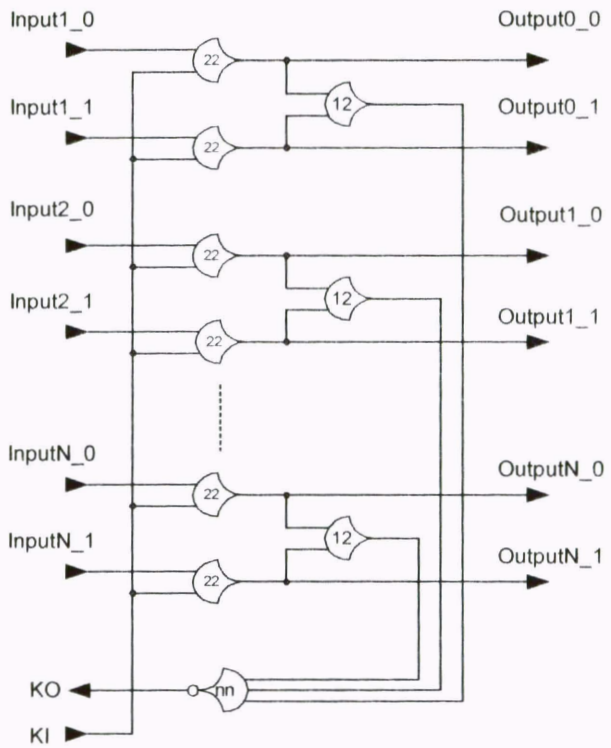


Figure 34. NCL pipeline components

In Figure 34, N inputs and N outputs of the registers are encoded in dual-rail logic. KI and KO are handshaking signals. KI is to inform the registers that the following stage has completed its storing step while KO is to inform the previous stage that these registers have completed a storing step. There are three kinds of threshold gates in Figure 34: TH22X0, TH12X0 and Inversed-THnnX0. The output of TH22X0 is logic high only when the two inputs are both high, and the output is logic low only when the two inputs are both low, otherwise the output remains unchanged. The behavior of TH12X0 is just like an OR gate. The output of Inversed-THnnX0 is logic high only when all *n* inputs are logic low, and the output is logic low only when all *n* inputs are logic high, otherwise the output remains unchanged. Suppose the circuit is now in a Null state, all signals are logic

low except KO. When all inputs become Data and KI becomes logic high, the outputs become Data. Since in dual-rail encoding, one of the two rails will be logic high in a Data state, the outputs of all TH12X0 gates become logic high. When all outputs become Data, the inputs of Inversed-THnnX0 are all logic high. Then KO becomes logic low indicating this Data storing step is complete. The Null storing step has the inverse procedure: when the inputs and KI are all Null, the outputs of the register and TH12X0 gates are all logic low. When all outputs become Null, KO will become logic high indicating this Null storing step is complete.

From these procedures it could be noticed that if the input precision has changed and some inputs are given Null instead of Data in a Data storing step, the registers "do not care". The corresponding outputs are automatically Nulls without changing of circuit structure. The problem lies in the KO generation circuit, also called completion circuit. In a Data storing step, when some of the inputs are replaced by Null, the outputs of the corresponding TH12X0 gates are logic low instead of logic high. So the inputs of Inversed-THnnX0 gates are not all logic high. KO will not become logic low. That means this Data storing step is never complete. Using Signal Bypassing & Insertion technique, two circuit structures are designed to solve this problem. The first one uses (N-1) multiplexers to selectively block the incorrect inputs to the Inversed-THnnX0 gate and make the design fully scalable to the changes of input precision. The sample circuit structure is shown in Figure 35(a). This method adds too many additional circuits. To simplify this structure, the other one divides the inputs to several groups. Since the largest THnnX0 gate is TH44X0, each group has four inputs. The changing step of input precision is limited to four so that only the completion output of each group needs to be

selectively blocked. So the number of multiplexers reduces to $\frac{N}{4}$. The sample circuit structure is shown in Figure 35(b).

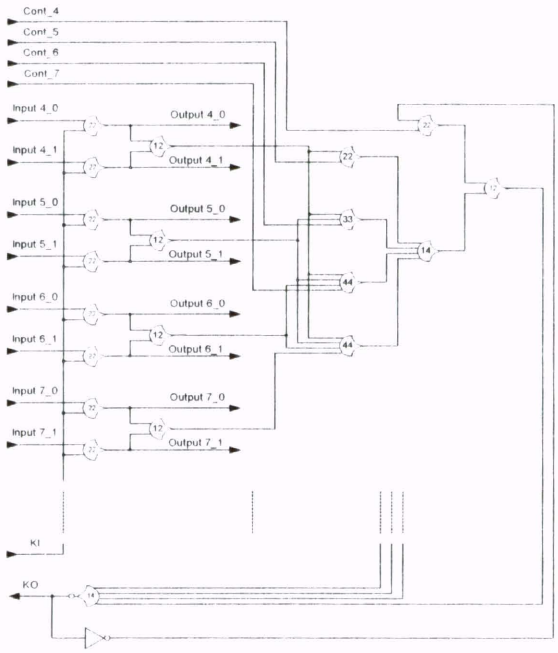


Figure 35(a). Energy-aware pipeline components — Fully scalable design

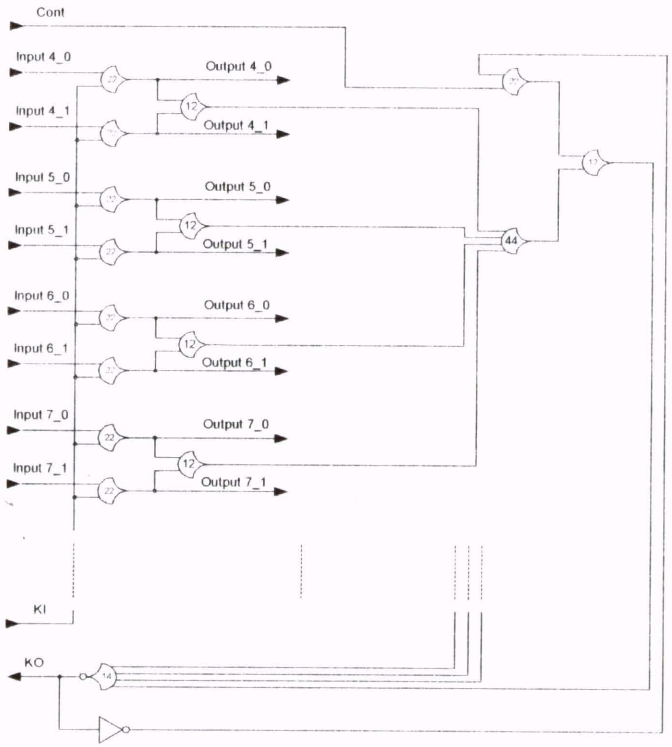


Figure 35(b). Energy-aware pipeline components — Quasi-scalable design

These design methods are implemented to design 16-bit pipeline components. The simulation results of energy dissipation and energy-delay product of these designs are shown in Figure 36. Figure 36(a) is the energy dissipation comparison and Figure 36(b) is the energy-delay product comparison.

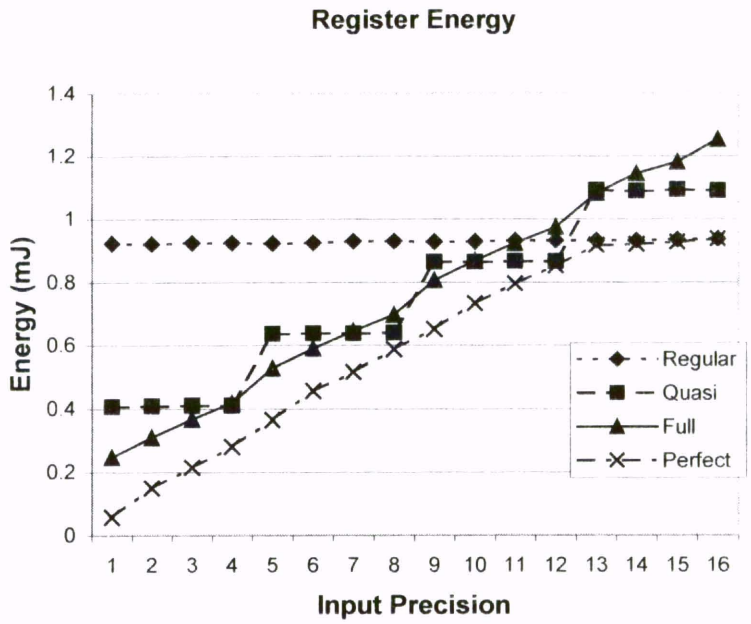


Figure 36(a). Energy dissipation comparison of different designs of pipeline components

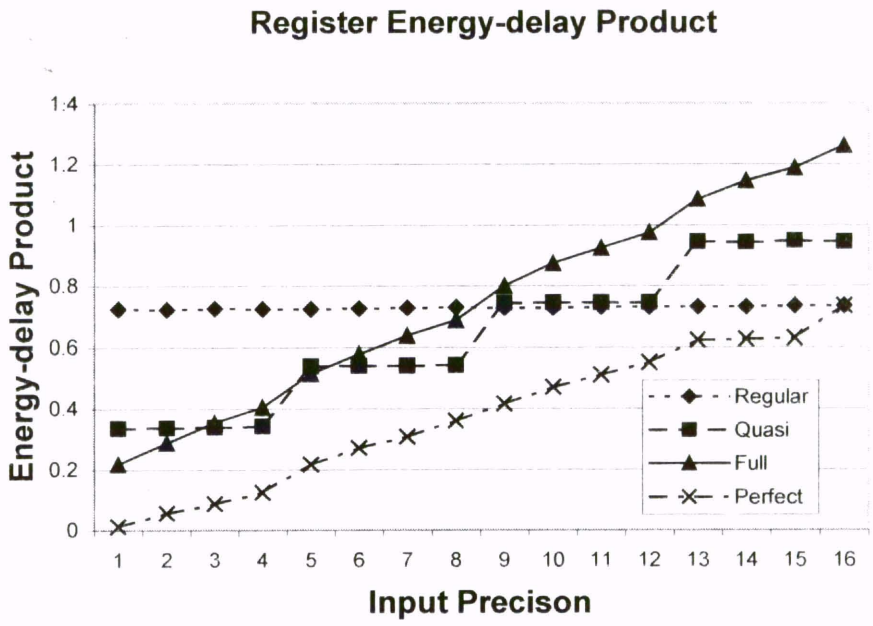


Figure 36(b). Energy-delay product comparison of different designs of pipeline components

In Figure 36, "Regular" means the original none-energy-aware design; "Quasi" means the quasi-scalable design"; "Full" means the fully scalable design; "Perfect" means the specific design that only performs operation of the designed input precisions. Several observations could be made from these figures:

1) Like energy-aware multiplier designs, in fewer input precision operation both fully and quasi-scalable designs show better energy dissipation and energy-delay product than original design. With the input precision increasing, the additional circuits in fully and quasi-scalable designs make them dissipate more energy than the original design.
2) The amount of additional circuits in fully scalable design is so large that in half of the situation the fully scalable design is worse than the original one in terms of energy-delay product. Considering the increased complexity and area, this design method doesn't show any benefit. But the quasi-scalable design has successfully solved this problem by compromising between the scalability and cost. Figure 8 shows in most of the cases, quasi-scalable design is better than the original one.

4.2.2.2 Energy-aware Multiplier Design

Using the technique described in Section 4.2.1, an 8×8 array multiplier is designed. The structure of the array multiplier is shown in Fig. 37[47]. This multiplier is fully adaptive to perform energy efficient calculation. Given different input precisions, this multiplier is simulated using Cadence SPICE simulator to get the energy and delay information. Figure 38 is the delay, energy and energy-delay product results of this 8×8 multiplier.

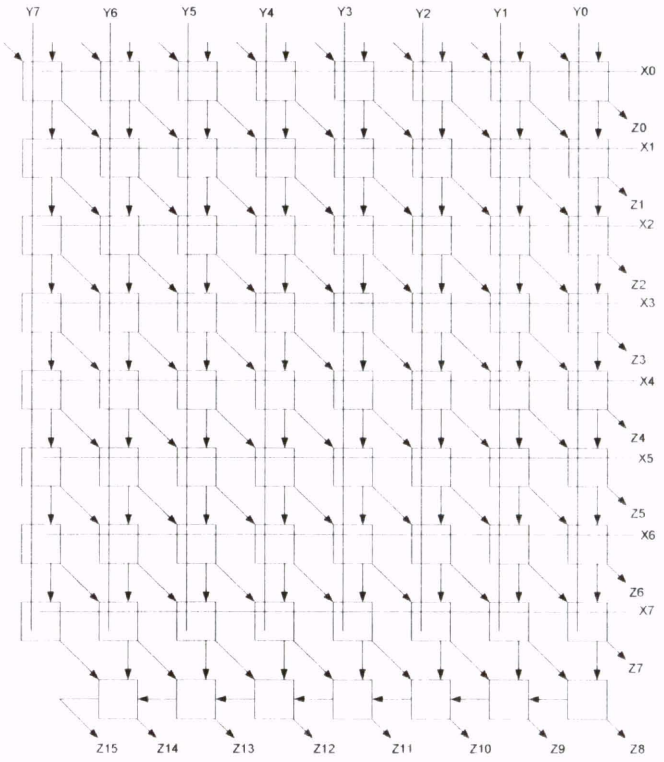


Figure 37. Array multiplier structure

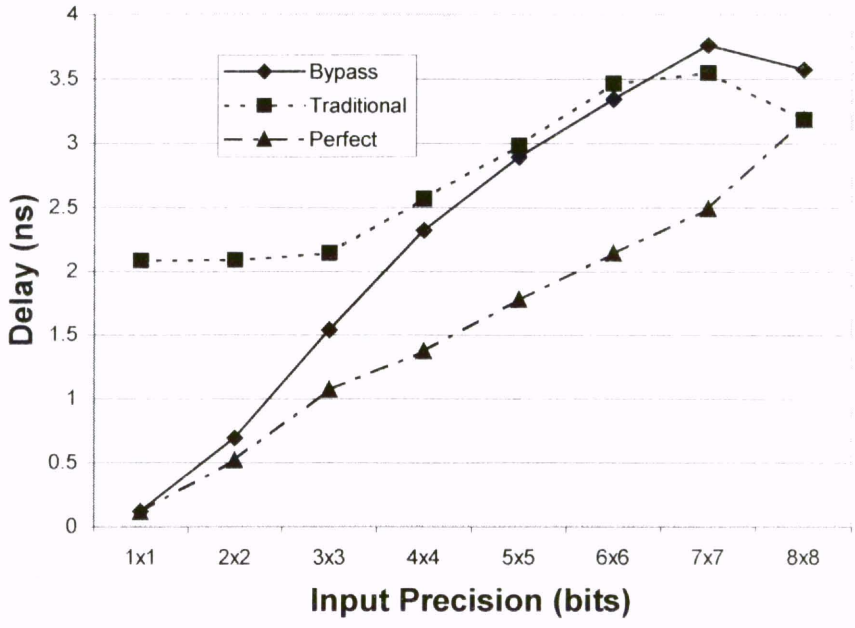


Figure 38(a). Simulation results of 8×8 multiplier — Delay comparison

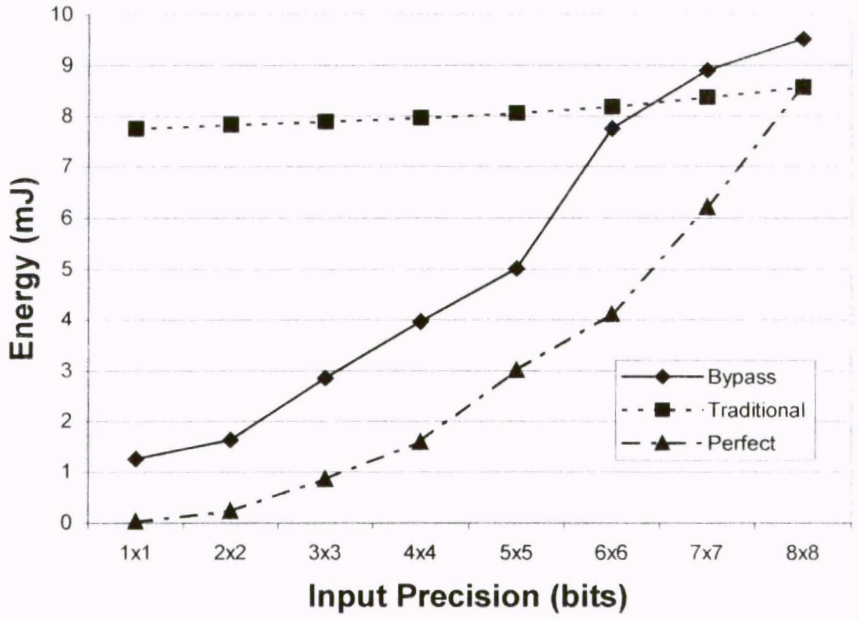


Figure 38(b). Simulation results of 8×8 multiplier — Energy comparison

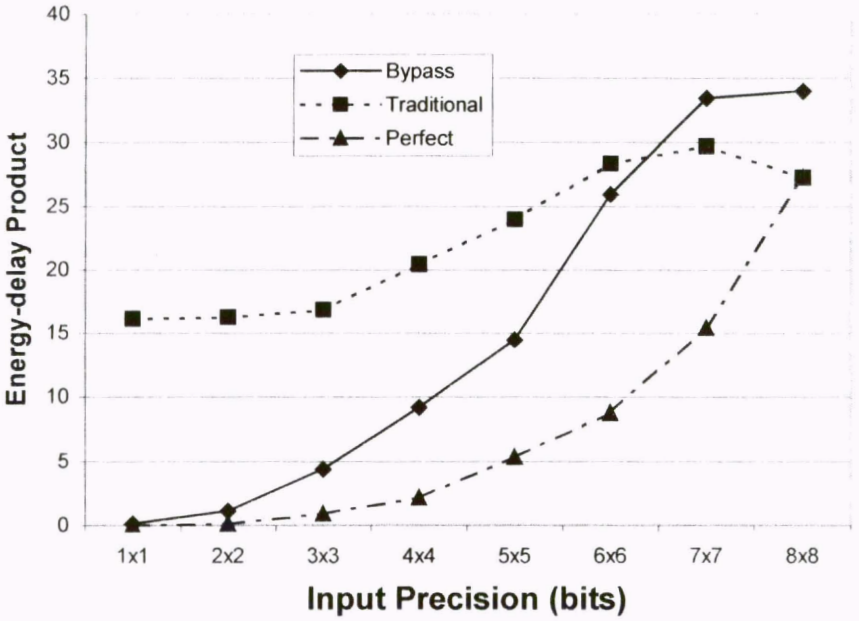


Figure 38(c). Simulation results of 8×8 multiplier — Energy-delay product comparison

In these figures, "Bypass" represents the simulation data of the designs using Signal Bypass & Insertion technique. These designs use Null to replace the redundant zeros in the high order bits of the multiplicands. "Traditional" represents the data of the original designs, which only use Data 0 and Data 1. "Perfect" represents the data of specially designed multipliers that only perform multiplication of the designed input precisions. For example, a 7×7 multiplier only performs 7×7 multiplication. The energy dissipation and delay of these "perfect" designs are the best because they consume only as much as energy and delay as their scenarios demand.

From these figures, several observations are made:

1) For energy dissipation, Bypass designs show great advantage over Traditional designs in fewer input precision multiplications. The reason for this advantage is that the existence of Nulls in high order bits of the multiplicands reduces the total number of switching. As the number of input precision increases, this advantage reduces due to the additional energy dissipated in multiplexers. At the designed input precision, Bypass designs dissipated more energy than Traditional designs. But in most of the situations, Bypass designs can save lots of energy.

2) For delay, Bypass designs also show significant advantage over Traditional designs in fewer input precision multiplications. The reason is that Bypass designs don't need to wait for the calculations of Data 0 in the high order bits of the output to complete. The same as that of energy, this advantage also reduces with the increase of input precision due to the additional delay in multiplexers. But in most of the situations, Bypass designs are faster.

3) Energy-delay product is a comprehensive measurement of circuit performance. The results of energy-delay product indicate the same trend as that of energy and delay: In most of the input precisions, Bypass designs are much better.

Bypass design needs some additional multiplexers to implement signal bypass & insertion. It can be easily calculated that for a K×K array multiplier, maximum (K-2) multiplexers are needed for signal bypassing and maximum $(K-2)^2$ multiplexers are needed for zero insertion. So total (K-2) × (K-1) multiplexers are needed in maximum. Because for a K×K multiplier, $o(K^2)$ adders are needed for calculation and there are fewer transistors in a 2-to-1 multiplexer than that in an adder, the additional area cost is bounded. Also, the area cost can be further reduced by merging zero insertion multiplexers as shown in Figure 39.

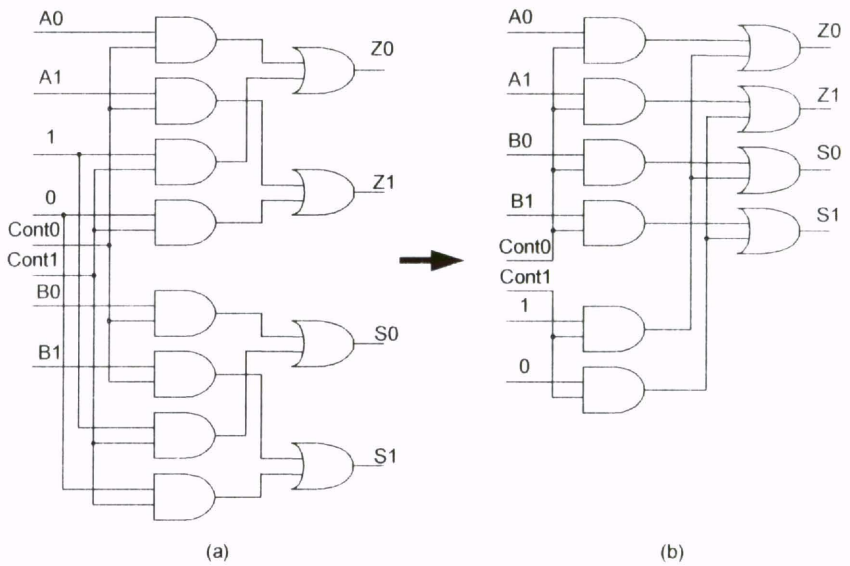


Figure 39. Area reductions in Mutiplexer

(a) Before reduction

(b) After reduction

In Figure 39(a), two zero-insertion multiplexers controlled by same control signal are shown. Because when the control signal is effective, the outputs of both multiplexers are Data 0 (in dual-rail logic), these multiplexers could be merged into one multiplexer as shown in Figure 39(b). For a K×K multiplier, all $(K-2)^2$ zero insertion multiplexers could be replaced by (K-2) merged multiplexers.

4.2.2.3 Energy-aware Counter Design

Counter is another important functional block in digital designs. Depending on a control signal, a counter can let the previous output data increase by one or remain the same. Since in the increment circuit, any input bit only used to calculate higher order bits and does not affect lower order bits, Null can be used directly to replace the redundant zeros in left most bits. But for multi-rail encoding circuit, the counter must be able to generate Data-Null sequence. The structure of a NCL counter is shown in Figure 40 [50].

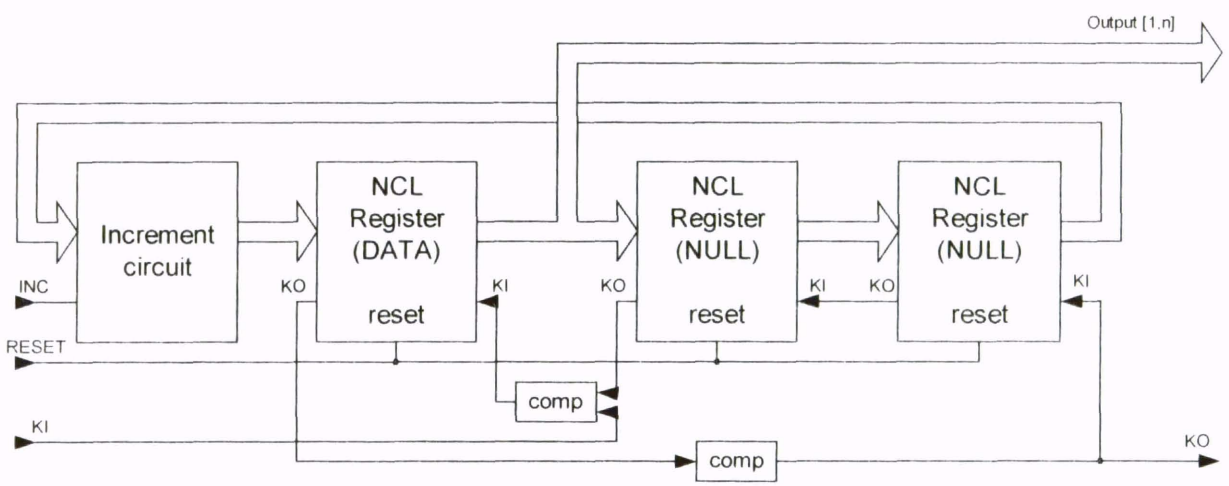


Figure 40. NCL counter structure

From Figure 40 it is clear that energy-aware NCL pipeline components must also be used in counter design. Four combinations of the increment circuit and pipeline

components, original increment circuit with original pipeline components (Regular & Regular), new increment circuit with quasi-scalable pipeline components (New & Quasi), new increment circuit with fully scalable pipeline components (New & Full), and specifically designed increment circuit with corresponding pipeline components (Perfect), are tested. Simulation results of energy and energy-delay product comparison are shown in Figure 41(a) and (b), respectively. From Figure 41 it is clear that two new counters show better performance in terms of energy dissipation and energy-delay product in fewer input precision operations. Since quasi-scalable design used less additional circuit than fully scalable design, the counter built by new increment circuit with quasi-scalable pipeline components is the best.

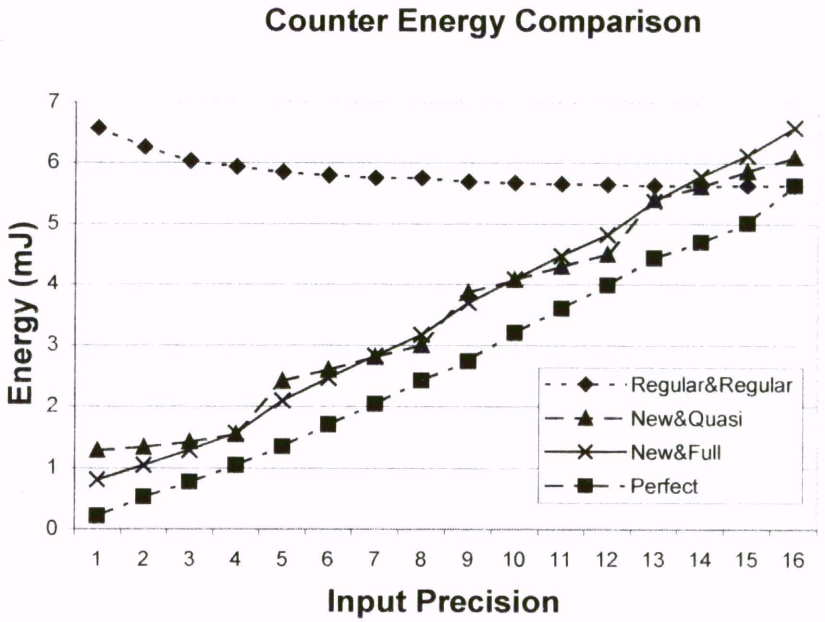


Figure 41(a). Comparison of different designs of counters — Energy dissipation comparison

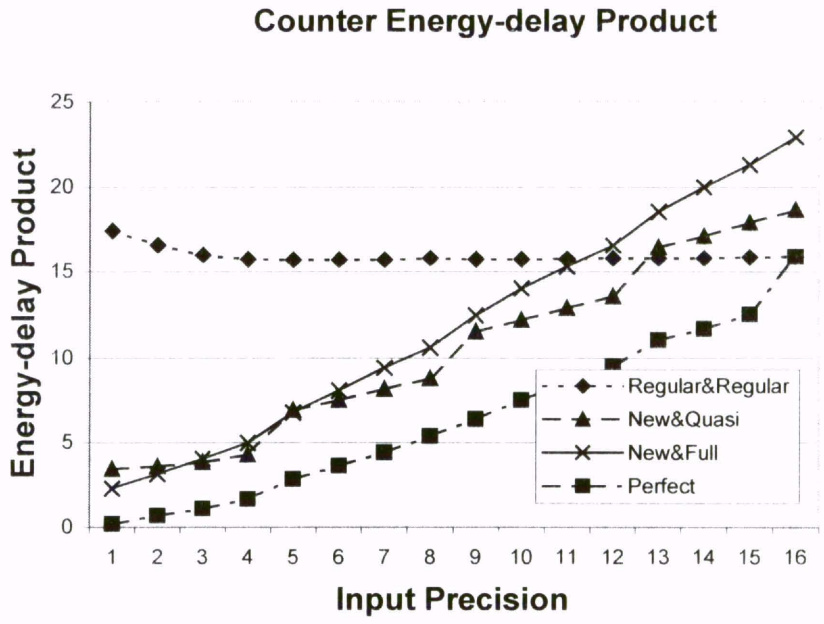


Figure 41(b). Comparison of different designs of counters — Energy-delay product comparison

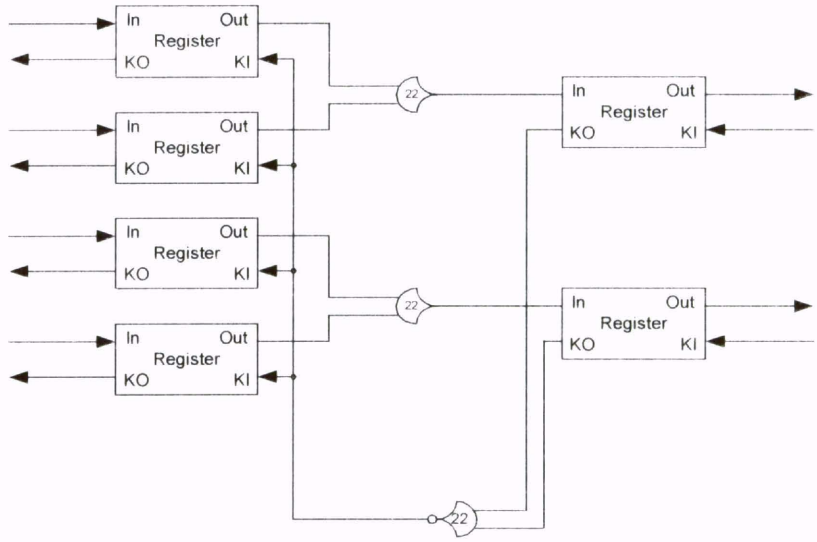
4.3 Zero Insertion Technique

Signal Bypassing & Insertion is for unpipelined NCL circuits. In this section, another technique named the Zero Insertion is proposed to design energy-aware bit-wise completion pipelined arithmetic circuits. While maintaining the speed-independency, both energy dissipation and delay are reduced.

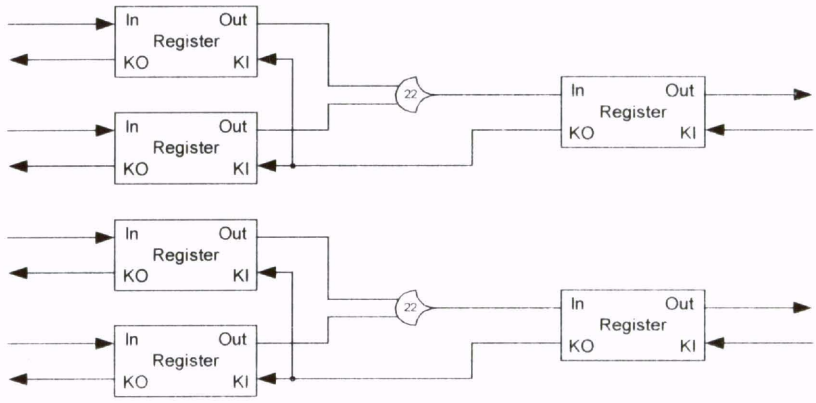
4.3.1 Bit-wise Completion

Completion strategy is part of the handshaking protocol in NCL circuits. The function of completion detection is to generate correct request/acknowledge signal to previous pipeline stage indicating the state of current stage. There are two major

completion-detecting strategies in NCL: full-word completion and bit-wise completion. Figure 42 shows the difference between these two strategies.



(a)



(b)

Figure 42. Comparison of full-word completion and bit-wise completion

(a) Full-word completion

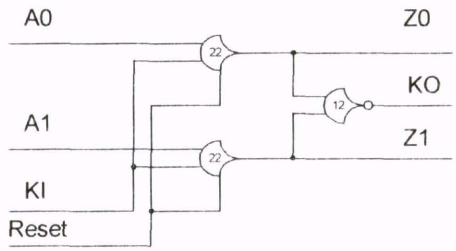
(b) Bit-wise completion

Each NCL register in Fig. 42 has two inputs and two outputs. Besides DataIn (In) and DataOut (Out) in dual-rail format, each register has a *KI* input and a *KO* output. The *KI* input is usually connected to the completion signal coming from the next stage, and the *KO* output is the completion signal of this stage that is usually connected to the *KI* input of the previous stage. In full-word completion, the completion signal for each bit in $register_i$ is conjoined by the completion component, whose single-bit output is connected to all *KI* lines of $register_{i-1}$. On the other hand, bit-wise completion only sends the completion signal from bit *b* in $register_i$ back to the bits in $register_{i-1}$ that took part in the calculation of bit *b* [50].

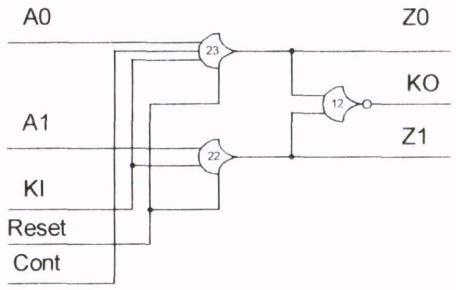
Bit-wise completion may require fewer logic levels than that of full-word completion, thus increasing throughput. It will never reduce throughput, since in the worst case all the bits of $register_{i-1}$ are used to calculate each bit of $register_i$, such that the completion logic and therefore throughput does not change by selecting bit-wise completion rather than full-word completion [50]. In this section, bit-wise completion scheme is used to design energy-aware arithmetic circuits.

4.3.2 Zero Insertion Technique

Zero Insertion technique is for bit-wise completion circuit. To avoid the Incomplete Evaluation problem, for such blocks whose inputs contain both Data and forced Null, this technique is to "insert" Data 0's to the forced Null inputs of this block. Since in the original design, those forced Null inputs are supposed to be Data 0's, this insertion guarantees the correctness of circuit operation. Figure 43 illustrates the implementation of this "insertion" in a bit-wise completion register cell.



(a)



(b)

Figure 43. Zero Insertion illustration in a bit-wise completion register cell

(a) Original bit-wise completion register cell

(b) Revised bit-wise completion register cell

The logic gates used in Fig. 43 are all threshold gates described in [44]. It can be seen in Fig. 43 that the only change from Fig. 43(a) to Fig. 43(b) is that Th22x0 gate is replaced by Th23x0 gate so that another control signal could be included in the circuit (Th12x0 is the same as an 2-input OR gate). By comparing the circuit structure of Th22x0 and Th23x0 in Fig. 44 and Fig. 45, respectively, it is clear that this change only costs 6 more transistors. But now a Data 0 can be easily inserted to replace the original Null output while the *Cont* signal is in logic high. So compared to the Signal Bypassing & Insertion techniques, the overhead in energy and delay could be significantly reduced.

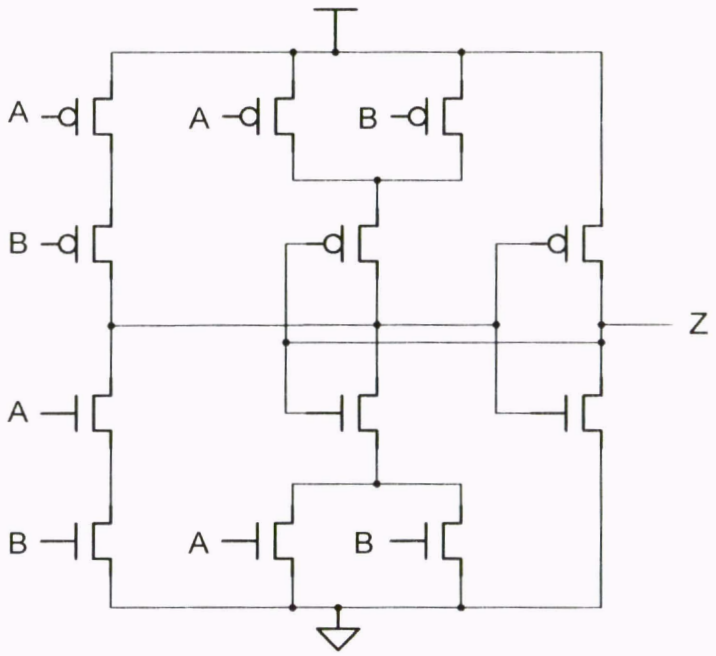


Figure 44. The circuit structure of Th22x0

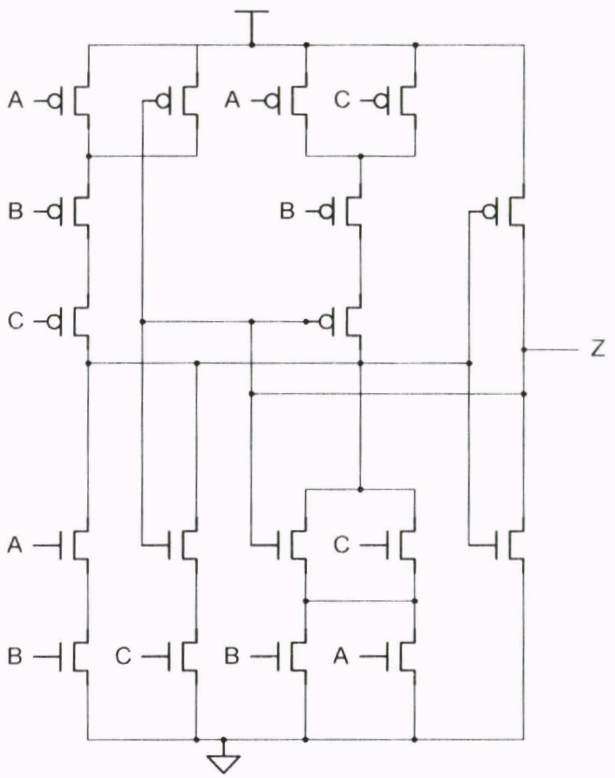


Figure 45. The circuit structure of Th23x0

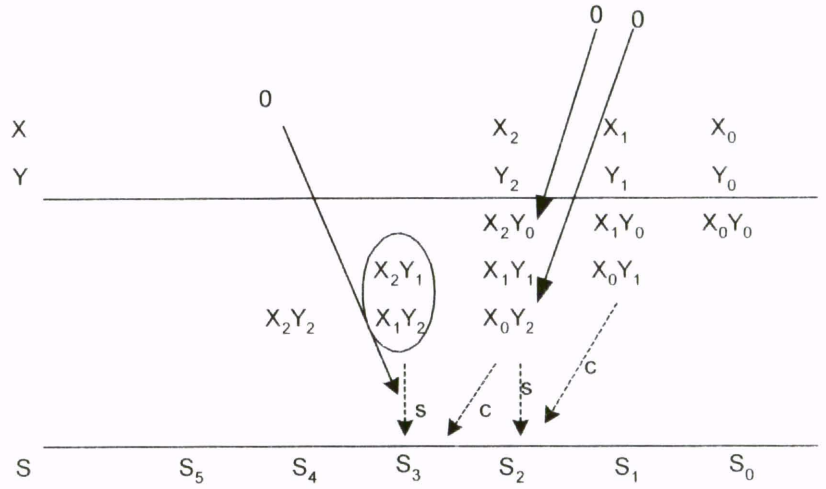


Figure 46. Illustration of Zero Insertion in 3×3 multiplication

As shown in Fig. 46, to perform N11×N11 with X_2 and Y_2 are forced to be Null, since X_2Y_0 and X_0Y_2 both should be zero in 011×011, the forced Null value of these two terms are replaced by Data 0's. Also, since X_2Y_1 and X_1Y_2 both should be zero in 011×011, Data 0 is also inserted to the sum of this adder to replace the forced Null, and to be added up with the carry coming from S_2 to calculate the correct S_3. After these two adjustments, N11×N11 can be used to calculate 011×011 in order to save energy while maintaining the correctness of the calculation. One control signal is added to control the operation of insertion. This control signal also comes from the central control unit of the entire device. The cost of this control signal generation will be discussed later in this chapter.

4.3.3 Maintain The Speed Independency

4.3.3.1 Speed Independency In NCL Circuits

Circuits that satisfy the property of correct operation for arbitrary gate delays but allow isochronic forks in the interconnecting wires are called speed-independent [51]. A

formal definition of speed-independency can be found in [52]. Using intermediate value (Null) and feedback strategy, NCL circuits are speed-independent. The basic operation of NCL circuits is shown in Fig. 47.

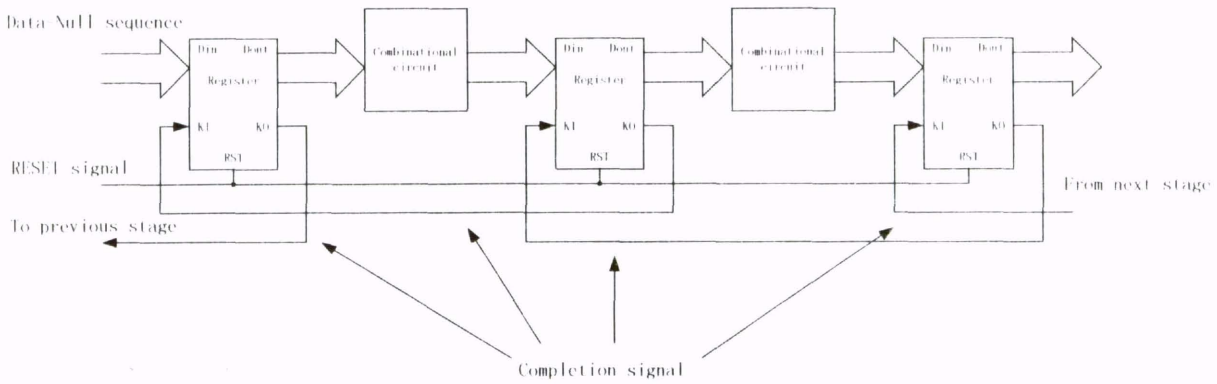


Figure 47. Operation of NCL circuit

In Fig. 47 between every two pipeline stages, there is a combinational logic block. The completion signal from each stage to its previous stage indicates if the calculation and registration of this stage are finished. When a calculation is finished, the completion signal becomes "Request-for-Null". Then a set of Null is allowed to propagate through the previous stage of registers to the combinational block of this stage. After all outputs of this block become Null, the completion signal is set to "Request-for-Data". Then a set of Data is allowed to propagate to this stage. This Data-Null sequence operation is independent of all gates' delay so that speed-independency is kept.

The generation of completion signal is shown in Fig. 48. There are *n* signals to be detected for completion. Each signal, represented by two wires, is detected by an inversed Th12x0 gate (Th12bx0), which acts as an NOR gate. If the output of this gate is 1, the

signal must be in Null state; otherwise the signal must be in Data state, no matter it is Data 0 or Data 1. Then all outputs of these Th12bx0 gates are connected to a Thnnx0 gate whose output is the completion signal. Remember that the output of Thnnx0 gate becomes 0/1 only when all *n* inputs to this gate are 0/1. So the completion signal is able to report current states of all *n* signals grouped by this Thnnx0 gate.

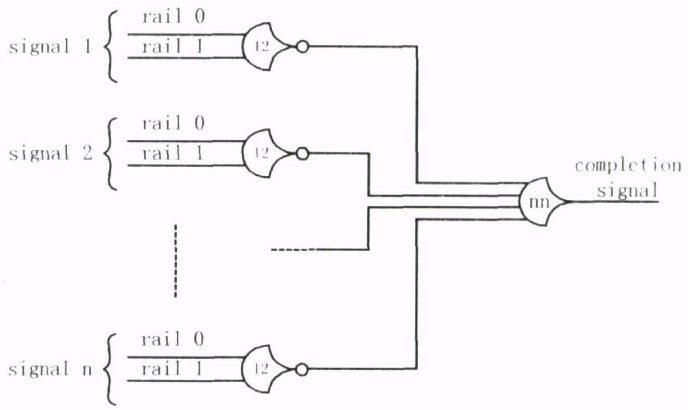


Figure 48. Completion signal generation

4.3.3.2 Selecting correct completion signals

During the modification process to make NCL circuits energy-aware, the speed-independency may be lost due to the break of Data-Null cycle operation. A general structure that causes this problem is shown in Fig. 49.

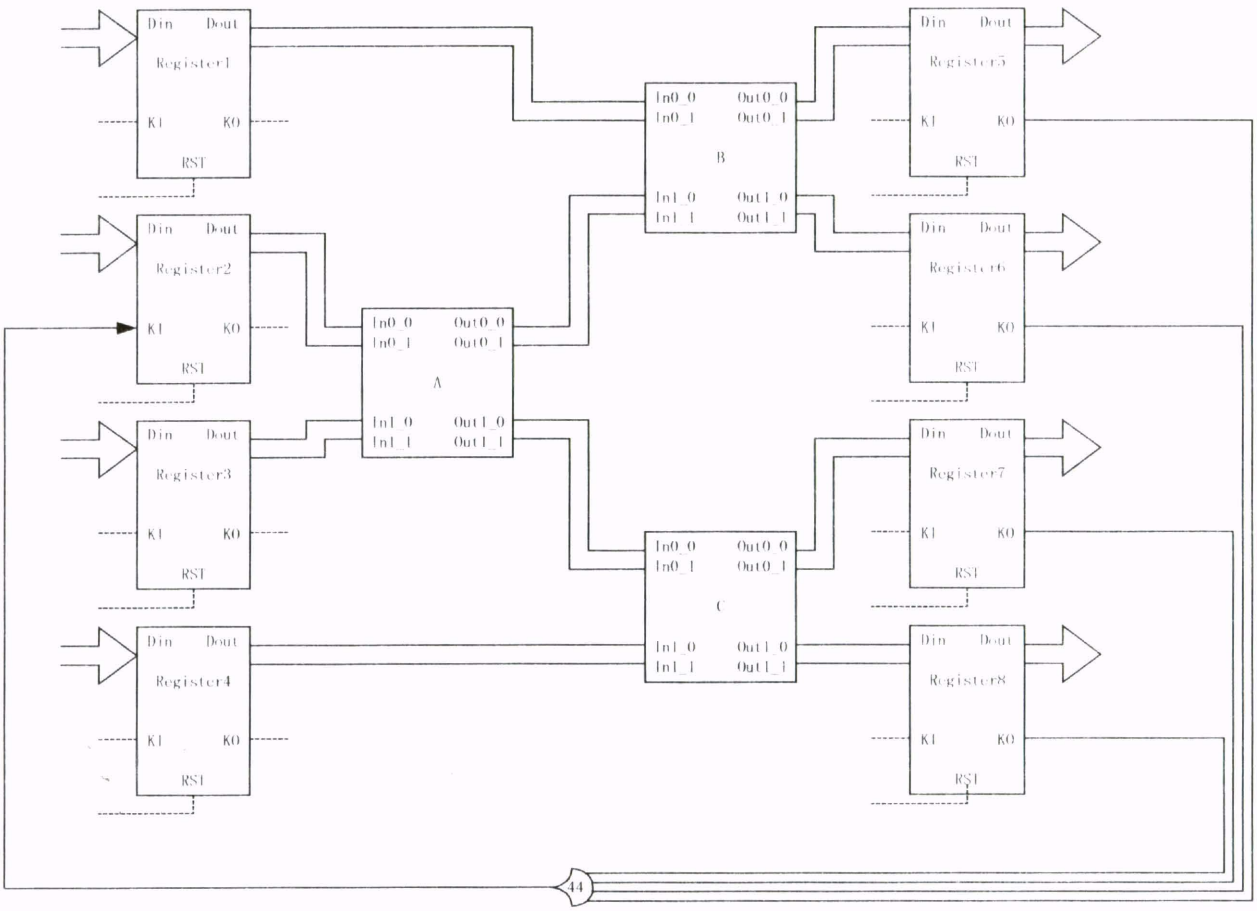


Figure 49. Example circuit to show the breaking of speed-independency

Registers 1 to 4 form the input stage while registers 5 to 8 form the output stage. Three functional blocks are in this stage; each has two inputs and two outputs. Suppose under a certain input precision, the inputs to register 1 and 2 are Data-Null cycles while the inputs to register 3 and 4 are forced to be Null all the time. To let block A generate the correct result, a Data 0 is inserted to register 3. So the outputs of block A and B are Data-Null cycles. But for block C, since one of its inputs is always Null, the outputs are most likely to be Null all the time because of the hysteresis property of NCL gates. Now consider the completion signal connected to register 2, since the output of register 2 is

used to generate all four outputs in this stage, its completion signal must combine all four completion signals of registers 5 to 8. But the outputs of registers 7 and 8 are always Null, so their completion signals are always in "Request-for-Data" state and never become "Request-for-Null". Then the completion signal to register 2 will stay "Request-for-Data" because of the hysteresis property of the Th44x0 gate used to generate this signal, and the next Null state will never pass this register. Thus the Data-Null sequence is broken.

To solve this problem and maintain speed-independency, corresponding completion signals from the next stage should be selected and generate the new completion signal to the previous stage due to different input precisions. A more specific example in multiplier design is "Edged adder". "Edged" adders are in the data path to generate the corresponding MSB of the output under certain input precision. From the circuit structure perspective, a 4×4 multiplication process is shown in Fig. 50 [50].

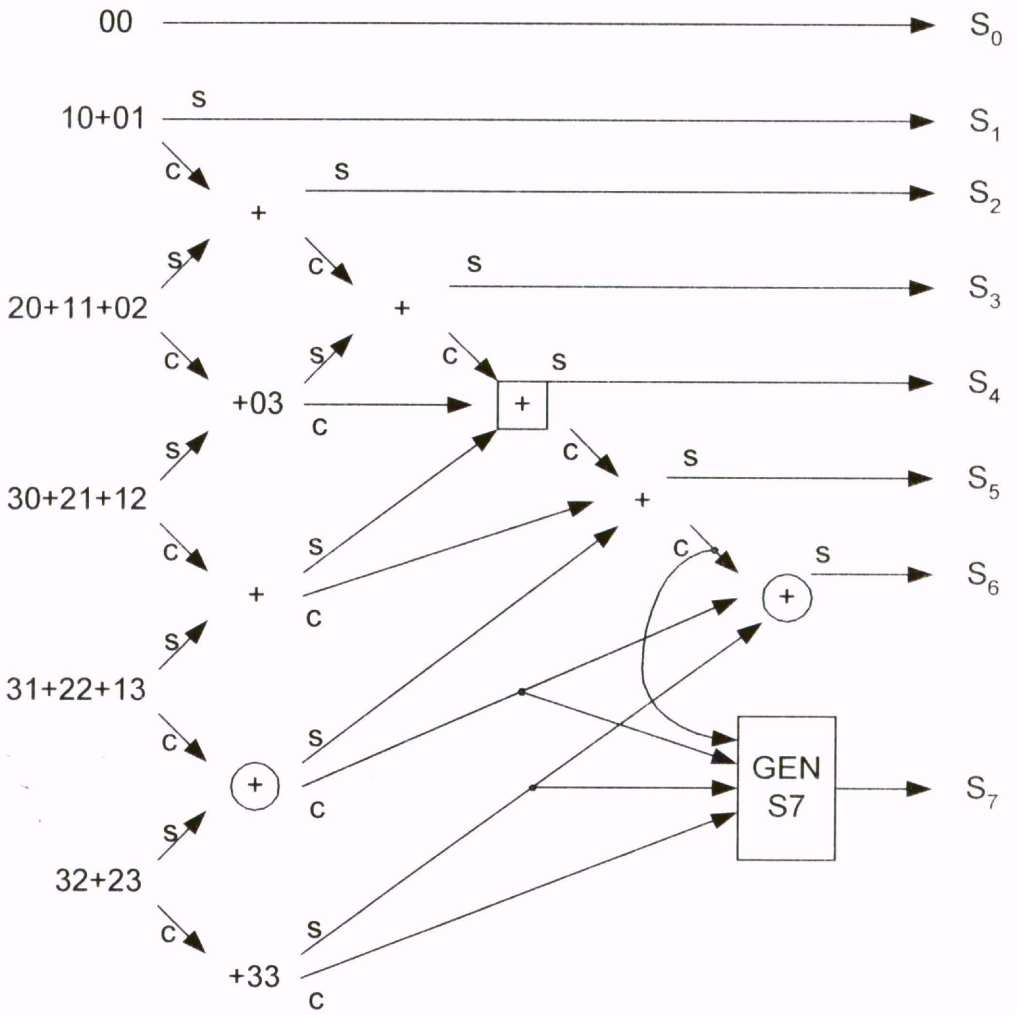


Figure 50. 4×4 multiplication process in circuit perspective

In Fig. 50 when performing 3×3 operation, the inputs of the two adders surrounded by circles will contain both Data 0's and Null. Since these two additions will never generate correct results, the following registers will keep sending "Request-for-Data" signals to the adders in the preceding stage. So the registers at the inputs of the adder performing "31+22+13" and the adder generating S_5 will never receive "Request-for- Null" signal and the Data-Null cycle will be broken. Same problem will occur to the

adder surrounded by a square while performing 2×2 operation. To solve this problem, only the completion of the sum needs to be considered since the output carries of these adders are always zero. The new completion circuit is shown in Fig. 51. The same control signal to switch the operation mode is used to select the correct completion signal.

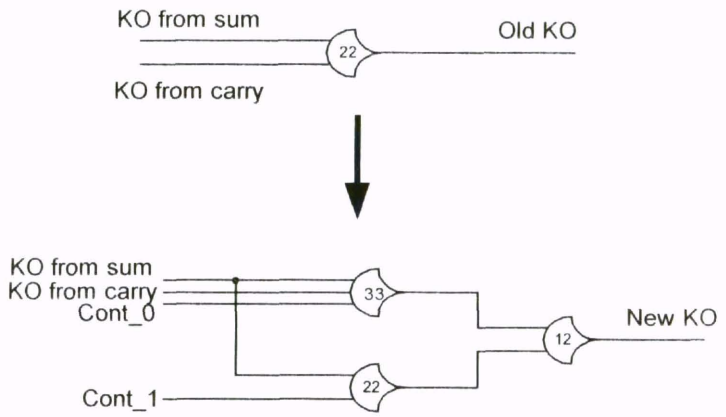


Figure 51. Circuit to solve the "edged" adder completion problem

4.3.4 Control signal generation and cost analysis

To implement the Zero Insertion technique, several control signals are needed to switch the operation mode of the circuit. These signals come from the central control unit. They are also dual-rail encoded and sent in Data-Null sequences. One way to generate these control signals is shown in Fig. 52.

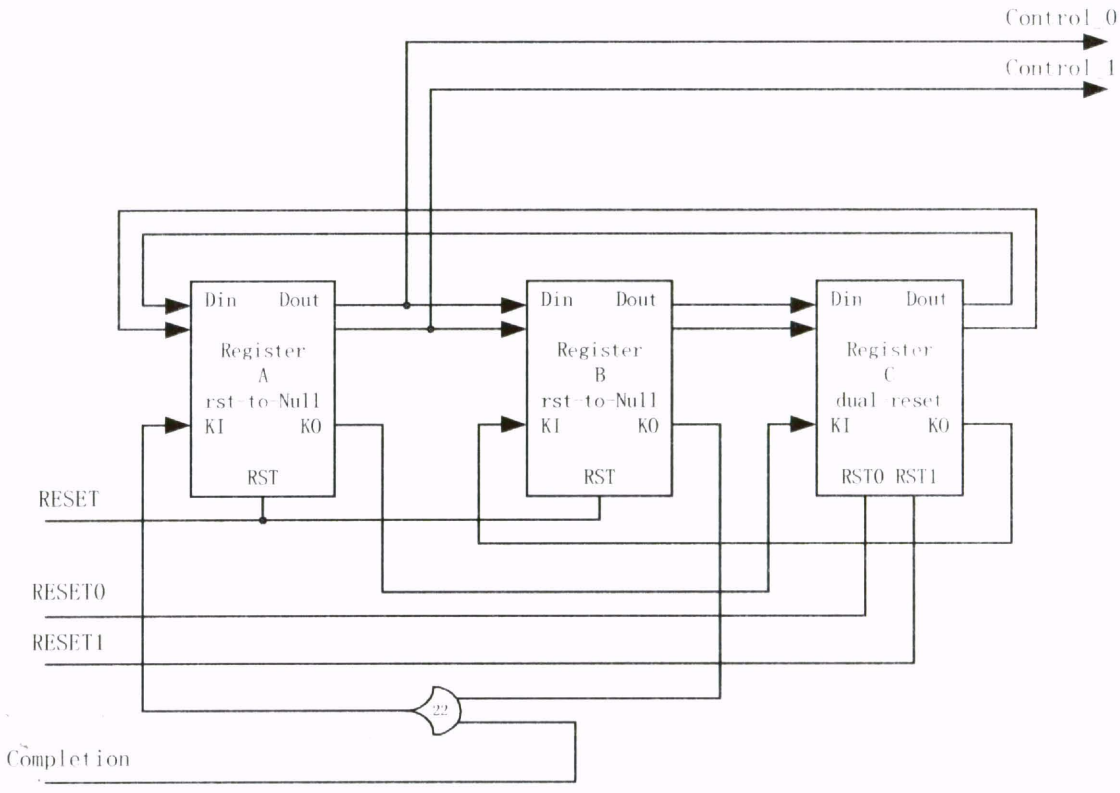


Figure 52. Control signal generator

Registers A and B are Reset-to Null registers, whose outputs become Null when the RESET signal is active. Register C is a dual-reset-register, the structure of which is shown in Fig. 53.

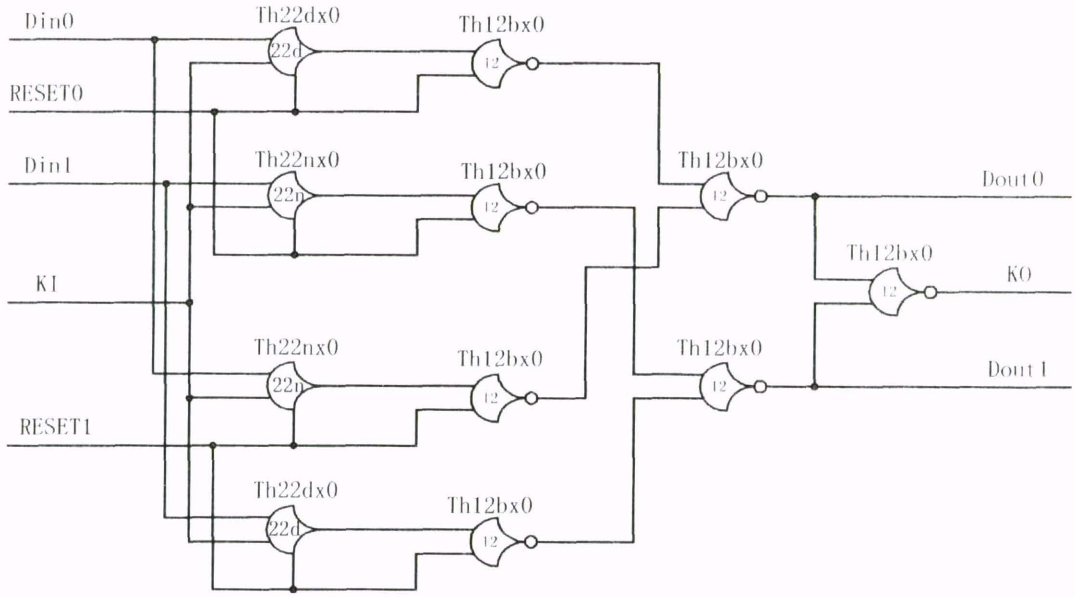


Figure 53. Dual-reset-register

There are three types of threshold gates in Fig. 53: Th22dx0 is a Th22x0 gate that outputs logic high while its reset signal is active; Th22nx0 is a Th22x0 gate that outputs logic low while its reset signal is active; Th12bx0 is an inversed Th12x0 gate. When the quality condition changes, such as user presses a selecting button, the central control unit resets all these three registers by activating RESET0, RESET1, and RESET. Then it deactivates RESET and one of the other two. By choosing which one between RESET0 and RESET1 remains active, the central control unit is able to set the output to Data 0-Null sequence or Data 1-Null sequence, thus controls the operation mode of the circuit. The input completion signal from the target circuit acts as the handshaking signal. Totally 18 threshold gates are used in this circuit. The overall additional area cost is very low compared to the area of the target circuit. For example, the area overhead caused by control signal generator of an energy-aware 8×8 Array multiplier is 8%. Since in both operation modes (Data 0-Null sequence and Data 1-Null sequence), there will always be

some gates that are idle, the power overhead caused by control signal generator is even less. The power overhead will be shown and discussed with each design application. Another scheme is let the target circuit detect the current input precision and set up its operation mode. But it requires much higher additional area cost.

As an example of the low area overhead of the Zero Insertion technique, the area overhead percentage comparison of Array Multipliers are shown in Fig. 54. These multipliers are fine-grain pipelined and are designed in different computation width. In Fig. 54 the overhead percentage goes down with the increment of computation width, and saturates at about 6.3%. Compared to the energy reduction rate shown in Fig. 59, this overhead percentage is very low. The exact transistor count of these multipliers are shown in Table 4.

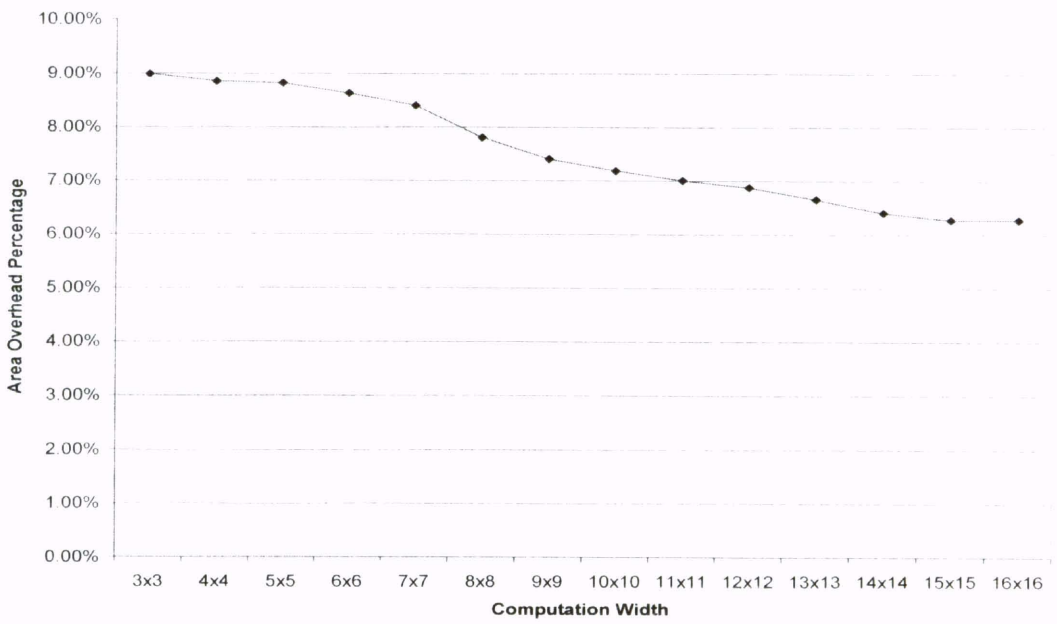


Figure 54. Area overhead comparison of Array multipliers using Zero Insertion technique

Table 4 Transistor count of Array multipliers using the Zero Insertion technique

	AND	Adder	Reg	Comp. Detect	Modified Reg	Modified Comp. Detect	Original Transistor Count	Additional Transistor Overhead	Overhead Percentage
3×3	9	6	33	12	3	7	1646	148	8.99%
4×4	16	14	71	22	8	10	3570	316	8.85%
5×5	25	26	112	36	15	13	6034	532	8.82%
6×6	36	41	181	53	24	16	9226	796	8.63%
7×7	49	58	247	72	35	19	13256	1116	8.40%
8×8	64	79	359	95	49	22	18556	1450	7.80%
9×9	81	102	487	120	68	25	24562	1818	7.40%
10×10	100	129	599	149	83	28	30634	2198	7.18%
11×11	121	158	781	180	115	31	38780	2724	7.00%
12×12	144	191	969	215	141	34	47448	3258	6.87%
13×13	169	226	1194	252	171	37	57410	3820	6.65%
14×14	196	265	1461	293	204	40	69322	4436	6.40%
15×15	225	306	1748	336	240	43	81430	5106	6.27%
16×16	256	360	1970	392	279	46	92942	5836	6.27%

4.4 Design Examples And Simulation Results

Using the Zero Insertion technique, the energy-awareness of NCL arithmetic circuits can be significantly improved. In this section, several design examples, including four types of 16-bit parallel adders, two types of parallel multipliers, and an 8×8 rank-order filter, are illustrated. The simulation results of energy and latency from Cadence are compared.

4.4.1 16-bit parallel adders

Parallel adders are essential functional blocks in arithmetic circuits. There are many algorithms for parallel adders. Four commonly used algorithms are chosen in this section: ripple-carry, carry-lookahead, carry-select, and carry-skip. The architectures of these adders can be found in [53]. These 16-bit adders are designed in NCL, and then revised by implementing the Zero Insertion technique to improve energy-awareness. In the rest of this section, "original design" refers to the direct NCL implementation of a

circuit structure, and "energy-aware design" refers to the revised design by implementing the Zero Insertion technique. Note that in energy-aware designs, the control signal generator is included. All these designs are pipelined in bit-wise completion strategy and each stage has a 2-gate delay, so that the throughput of all adders is the same. The carry-lookahead adder and carry-select adder are implemented into two-level structures while the other two are in one-level structures.

The simulation results are normalized. The definition of normalized energy dissipation and latency is given as follows:

$$E_{normal} \equiv \frac{E_{aware}}{E_{original}}, \quad T_{normal} \equiv \frac{T_{aware}}{T_{original}} \tag{13}$$

In (13) E_{normal} and T_{normal} are normalized energy dissipation and latency, respectively, E_{aware} and T_{aware} are energy dissipation and latency in energy-aware design, and $E_{original}$ and $T_{original}$ are energy dissipation and latency in the original design. When E_{normal} is less than 1, there is energy saving in energy-aware design; when E_{normal} is greater than 1, there is energy overhead in energy-aware design due to additional cost from the control circuit. The smaller E_{normal}, the larger saving of energy.

Figures 55 and 56 show the simulation results of the four adders and their energy-aware counterparts with control signal generator under different input precisions. "Ripple", "Lookahead", "Select" and "Skip" refer to the simulation results of ripple-carry adder, carry-lookahead adder, carry-select adder, and carry-skip adder, respectively.

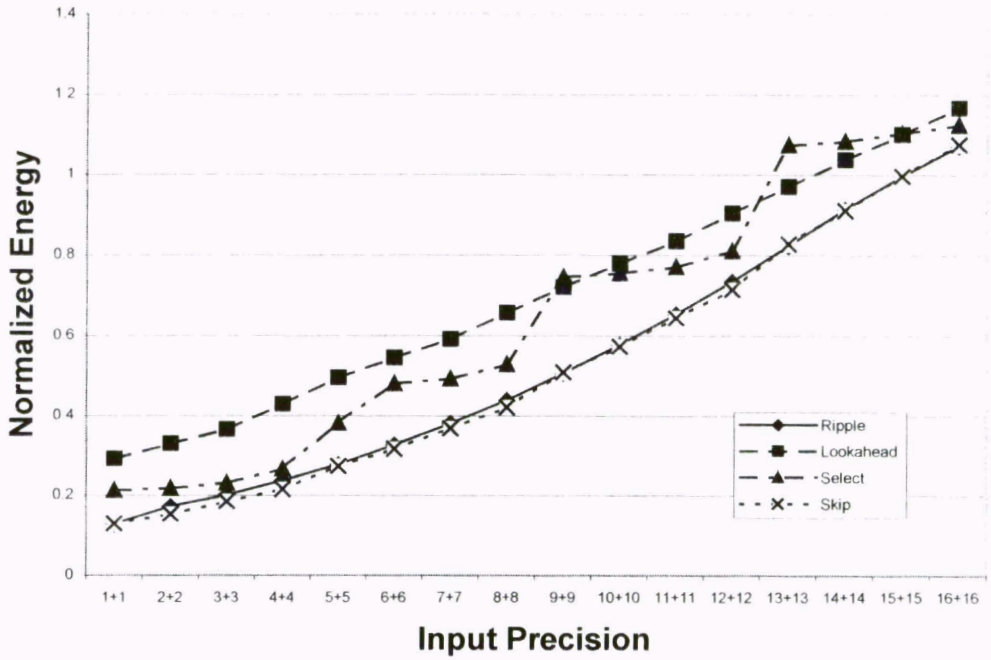


Figure 55. Normalized energy dissipation of parallel adders

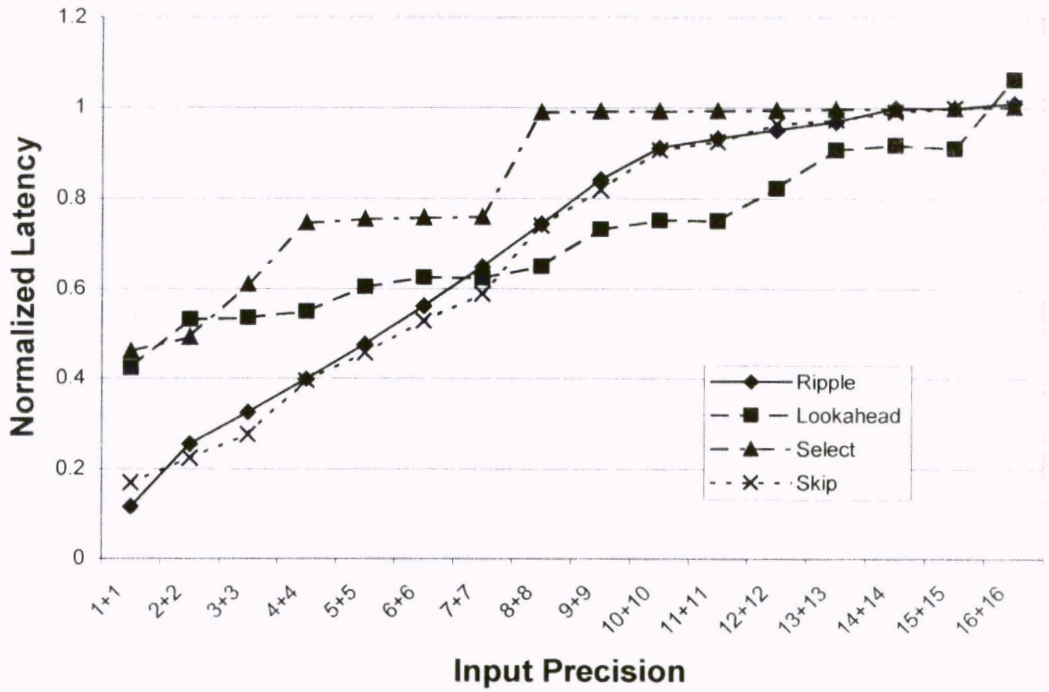


Figure 56. Normalized latency of parallel adders

From Figs. 55 and 56 several remarks are given as follows:

1) The normalized energy dissipation under most of the input precisions for all four types of adders is less than 1. A significant amount of energy saving can be achieved. Under an equal input precision probability (i.e., all input precisions have the same probability to occur), the overall energy savings are between 30% and 48%.

2) Among the four types of adders, the ripple-carry adder and carry-skip adder have the most energy saving rate, while the carry-lookahead adder saves the least. This is due to higher regularity of circuit structures for ripple-carry and carry-skip adders. Only a small amount of additional circuitry is needed to make them energy-aware. The structure of carry-lookahead adder is more complicated and needs more circuitry to improve energy-awareness.

3) The energy dissipation overheads of all four types of adders are very small. Only under highest input precisions (15-bit and 16-bit), E_{normal} is slightly greater than 1 due to 10% overhead from additional insertion circuitry and control signal generator.

4) The latency of all four types of adders is also less than 1 for most of the input precisions. The latency reduction is because when Null is used to replace Data 0's in high order bits, the number of bits in output is reduced. So there is no need to wait for the calculation of Data 0's in the high order bits of the output. The overall latency reductions of all four types of adders under an equal input precision probability are between 15% and 32%.

5) The latency overheads of these adders are also very small. The largest overhead also occurs in the carry-lookahead adder due to its additional circuitry. The carry-select adder has the smallest latency reduction and step-looking curve since each selection unit contains 4 bits.

4.4.2 Parallel Multiplier Design

A series of parallel-unsigned multipliers are designed using the Braun Array multiplication algorithm [47] (circuit architecture is recalled as Fig. 57) and Dadda Tree multiplication algorithm [54] (circuit architecture is shown in Fig. 58), respectively. The widths of multipliers are 4-bit, 6-bit, and 8-bit.

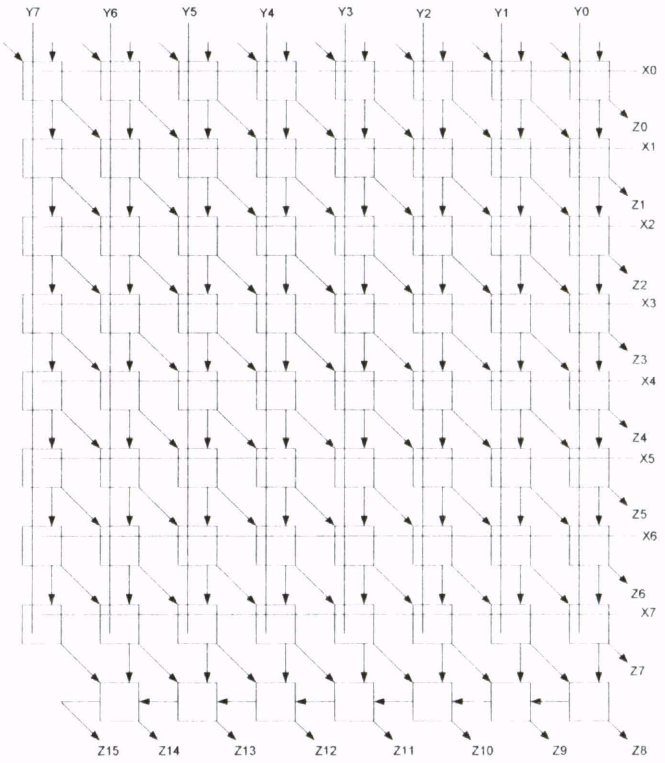


Figure 57. Circuit architecture diagram of Array multiplier

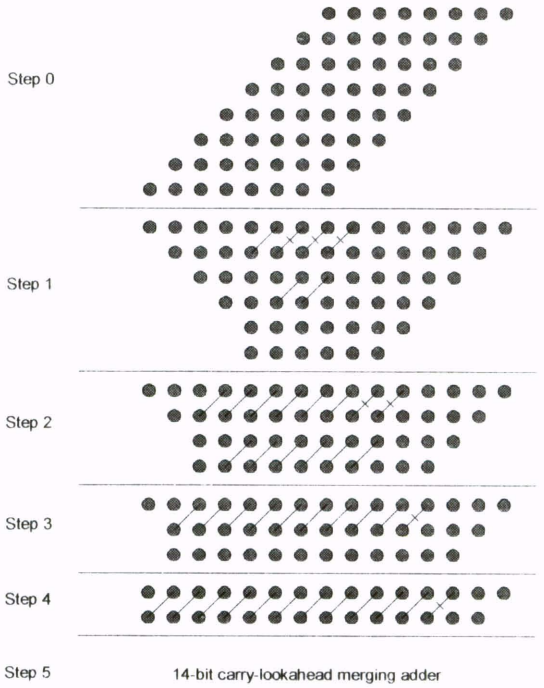


Figure 58. Circuit architecture diagram of Dadda Tree multiplier

In Fig. 57 each block represents a full-adder that includes AND gate. X and Y are inputs while Z is the output. A 14-bit ripple-carry adder is used as final merging adder in the bottom row of Array multiplier. In Fig. 58 each dot represents a bit product. Short lines represent full-adders while "crossed" short lines represent half-adders. A 14-bit carry-lookahead adder is used as merging adder in the Dadda Tree multiplier. Each multiplier is pipelined using bit-wise completion and each stage has a 2-gate delay. To show the trend of energy saving and latency reduction with changing of multiplier width, 4×4 and 6×6 Array multipliers and their energy-aware counterparts were also designed and simulated. Control signal generators were included in all energy-aware designs. The simulation results of these three multipliers are shown in Figs. 59 and 60.

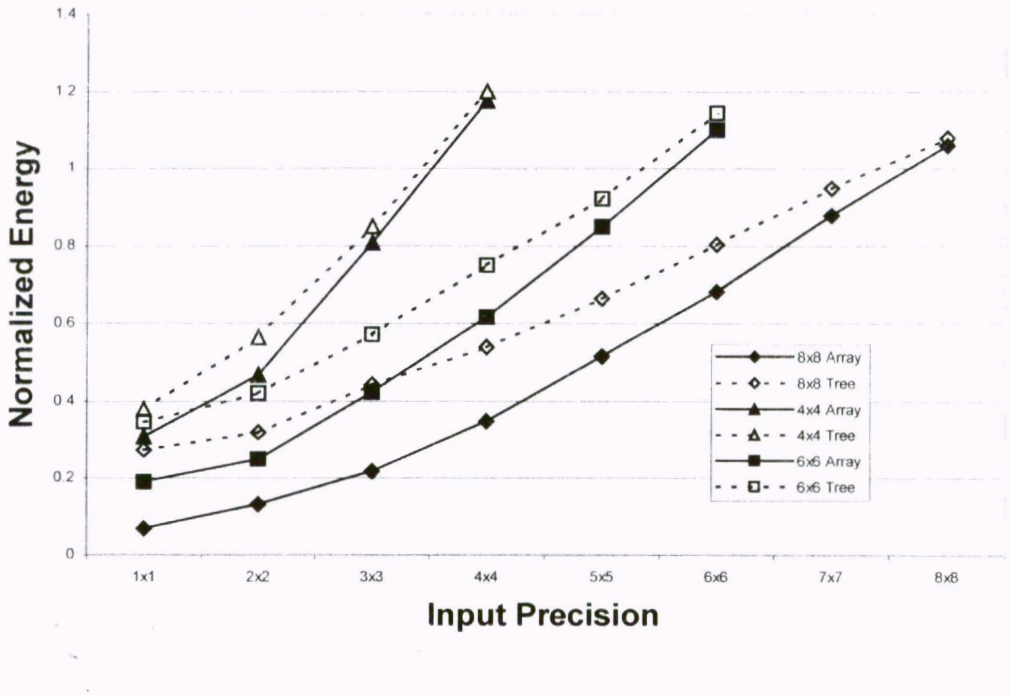


Figure 59. Normalized energy dissipation of multipliers

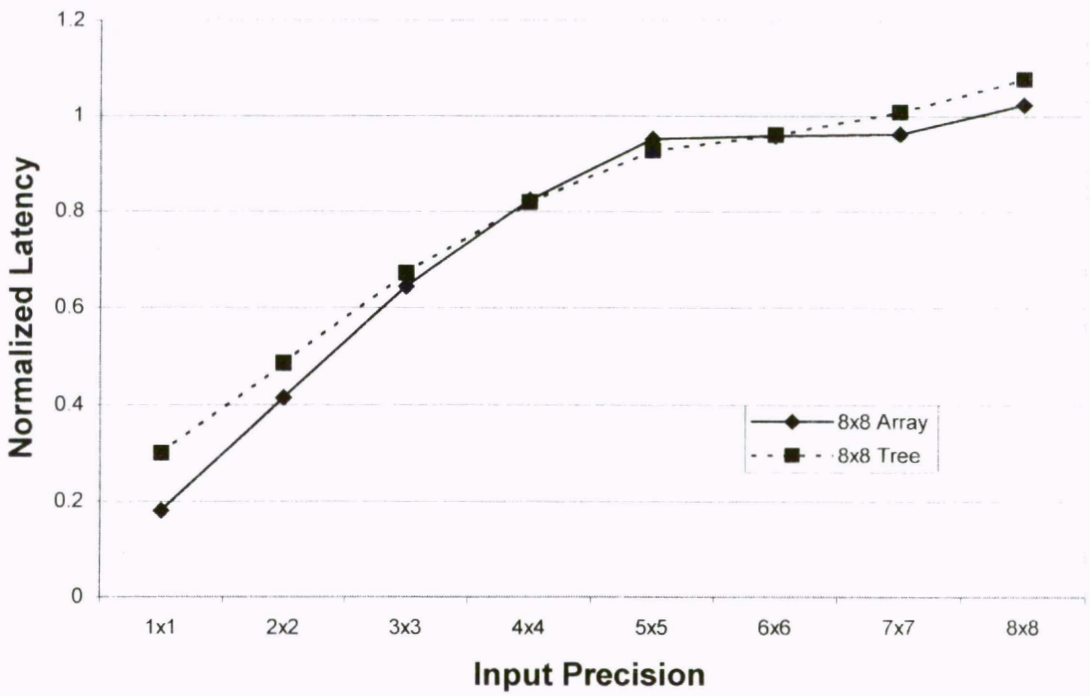


Figure 60. Normalized latency of multipliers

The following remarks are derived from these figures:

1) The energy saving in 8×8 multiplication, the Array multiplier is better than the Dadda Tree multiplier for all eight input precisions. This is due to different types of merging adder. Since the ripple-carry adder has more energy reduction than the carry-lookahead adder in implementing the Zero Insertion technique as seen in Fig. 55, the designed Array multiplier, which uses the ripple-carry adder as a merging adder, has better energy-awareness. But both multipliers have significant energy savings. The average energy saving under an equal input precision probability is 52% for the Array multiplier, and 37% for the Dadda Tree multiplier.

2) The energy overheads of both 8×8 multipliers are very small. The use of the carry-lookahead adder makes the overhead a little larger for the Dadda Tree multiplier, which is 7%. For the Array multiplier, the overhead is only 6.3%.

3) For the latency of 8×8 multipliers, both multipliers show good reduction rate under most of the input precisions. The average latency reduction rate under an equal input precision probability is 24% for the Array multiplier and 21% for the Dadda Tree multiplier. Due to the use of carry-lookahead adder, the Dadda Tree multiplier has a larger latency overhead of 7%.

The simulation results also show energy savings and latency reduction at lower input precisions for 4×4 and 6×6 multiplications. The 4×4 and 6×6 multipliers exhibit similar characteristics compared with the 8×8 multiplier. For each multiplier width, significant energy saving and latency reduction can be achieved even that longer width has more benefits overall. It is worthy mentioning that the area of control signal generator

increases linearly, while the area of the whole multiplier increases exponentially with multiplier width. Relatively, the larger the multiplier width, the lower the overhead cost.

4.4.3 8×8 rank-order filter

Rank-order filters are nonlinear filters that choose an output based on its rank within a window of sample inputs determined by sorting the inputs [55]. The ability to detect and remove nonlinear noise like impulse noise makes rank order filters widely used in image processing. The applications include picture smoothing, noise reduction, and edge detection [56]. The input output relationship of a rank-order filter is given by (14), where $x(n)$ is the input series and $y(n)$ is the output series.

$$y_r(n) = r^{th}\ rank\ [x(n-N), x(n-N+1), \cdots, x(n), \cdots, x(n+N-1), x(n+N)] \tag{14}$$

4.3.3.1 Rank-order filter architecture

There are various architectures developed for rank-order filter. A very good survey of these architectures was given in [55]. The architecture chosen in this section is based on Batcher's odd/even merge-sort algorithm. The basic element of this architecture is Compare & Swap (C&S) unit shown in Fig. 61.

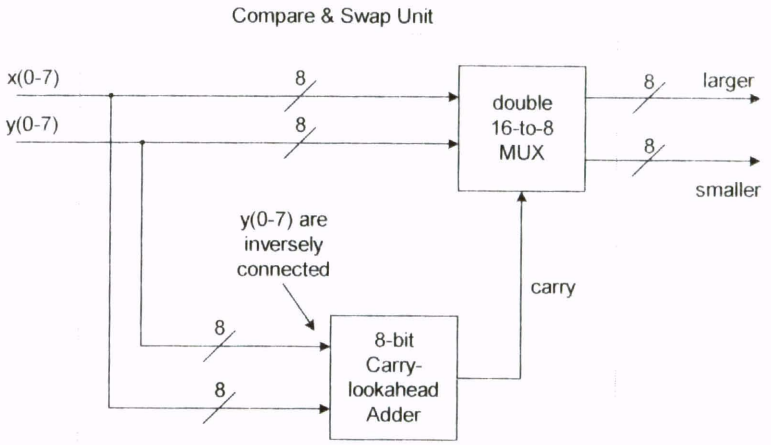


Figure 61. C&S unit structure

The function of C&S unit is to compare the two inputs, swap if necessary and let the larger one connected to the "H" output and the smaller one to the "L" input. This two-input C&S unit (also called Merge 1×1) is used to build four-input (Merge 2×2) and eight-input (Merge 4×4) C&S units, which are circuit blocks to implement Batcher's algorithm on rank-order filter. The architecture used in this section is a merge-sort rank-order filter with window size $W = 8$, which is shown in Fig. 62.

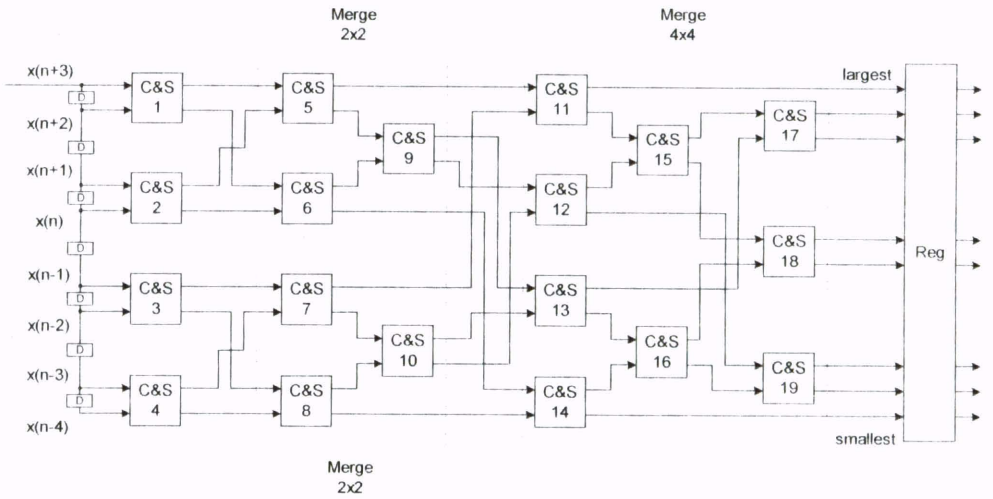


Figure 62. A rank-order filter with window size W=8

There are two variables that could change during the operation of rank-order filter: one is the window size; the other is the input sample wordlength. Window size determines the number of sample data to be sorted at the same time. Sample input wordlength determines the accuracy of image representation. Due to different quality and battery life requirements, both variables may be changed. Based on this observation, the Zero Insertion technique is used to design the energy-aware rank-order filter.

4.3.3.2 Implementing the Zero Insertion technique

The original bit-wise completion pipelined NCL design of the rank-order filter is 8-bit, with window size of 8. After implementing the Zero Insertion technique, this filter is able to work with different window sizes from 8 to 1, and different wordlengths of 8-, 4-, and 1-bit, with improved energy-awareness.

When the rank-order filter in Fig. 62 is used with reduced window size, additional circuit must be applied to connect input to the corresponding delay element stage and to insert redundant Data 0's to other delay elements at the top. These adjustments not only increase the circuit complexity, but results in more energy dissipation than reduction. The proposed solution in this section is to connect the *KO* of the output stage to the corresponding delay element stage, bypass the rest of the delay elements at the bottom. These bypassed delay elements will produce an output Null forever because there is no *KO* signal connected to them. The Null replacing the redundant Data 0's will act as inputs to the merge units. Since there is no transition between two Null states, significant energy saving can be achieved. Both solutions are illustrated in Fig. 63.

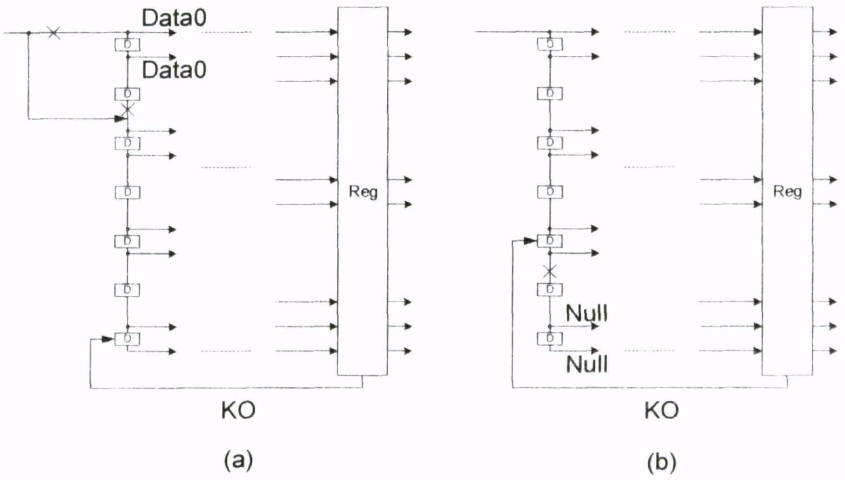


Figure 63. Solutions for filters with reduced window size

(a) Original design

(b) Energy-aware design

Also, as the sample input wordlength reduced, Null is used to replace the redundant Data 0's in the high order bits of each data. To make the filter work properly under these mixed Data-Null inputs, modifications are needed for circuit components.

For the C&S unit, on the changing window size, there are three operation modes for the C&S units. If both two inputs of the C&S unit are enabled, that is, no forced constant Null, the C&S unit functions normally, so called the normal mode; if one of the two inputs is disabled and the other one is enabled, the C&S unit should put the enabled input to the larger output and put the other to the smaller output, so called the bypass mode; if both inputs are disabled, the C&S unit is not functioning, so called the disable mode. There is no modification for normal and disable modes. For the bypass mode, a signal named the Bypass EN is connected into the unit to force a control signal to let the multiplexer select the corresponding input. Since one of the inputs is Null, the output of

the carry-lookahead adder will be Null and therefore the multiplexer stops functioning. This Bypass EN signal comes from the central control unit of the system. When the user selects the current window size, this control unit is able to know the operation modes of all C&S units and set up the control. The structure of the revised C&S unit is shown in Fig. 64.

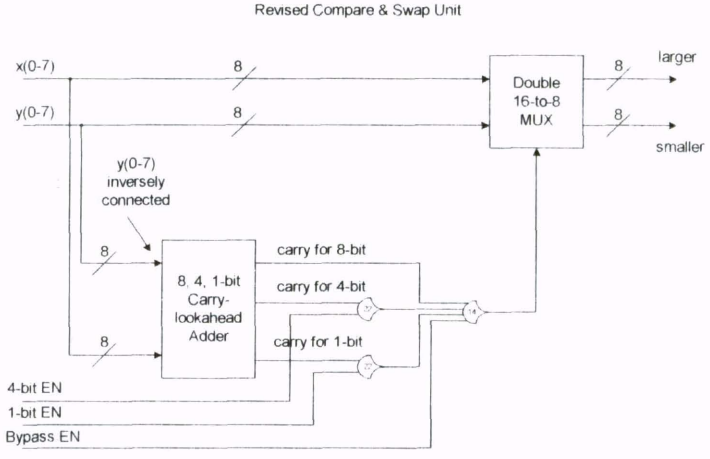


Figure 64. Revised C&S unit structure

For delay elements and output registers, due to the changing wordlength, NCL registers must be able to generate corresponding completion signals. This can be easily accomplished by combining completion signals from different groups of 1-bit registers. The block diagrams of the original and revised 1-bit NCL register are shown in Fig. 65. For different window sizes, each delay element and output register stage should be able to choose the completion signal from the next stage (in normal mode), the output stage (when all other delay element stages after this stage are disabled), and no completion signal at all (when this stage is disabled). Window size enable signals are added to the

filter to control the working mode. These signals, coming from the central control unit, also act as the Bypass EN signals for all C&S units.

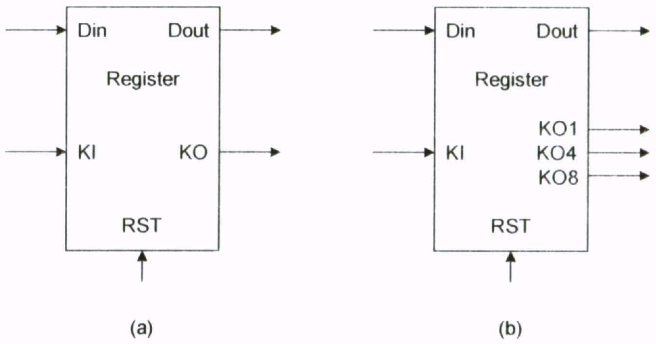


Figure 65. Revised NCL register

The original and modified rank-order filters are simulated in Cadence IC Design Environment for different window sizes and input wordlengths. Both energy dissipation and latency are measured. The results are shown in Figs. 66 and 67.

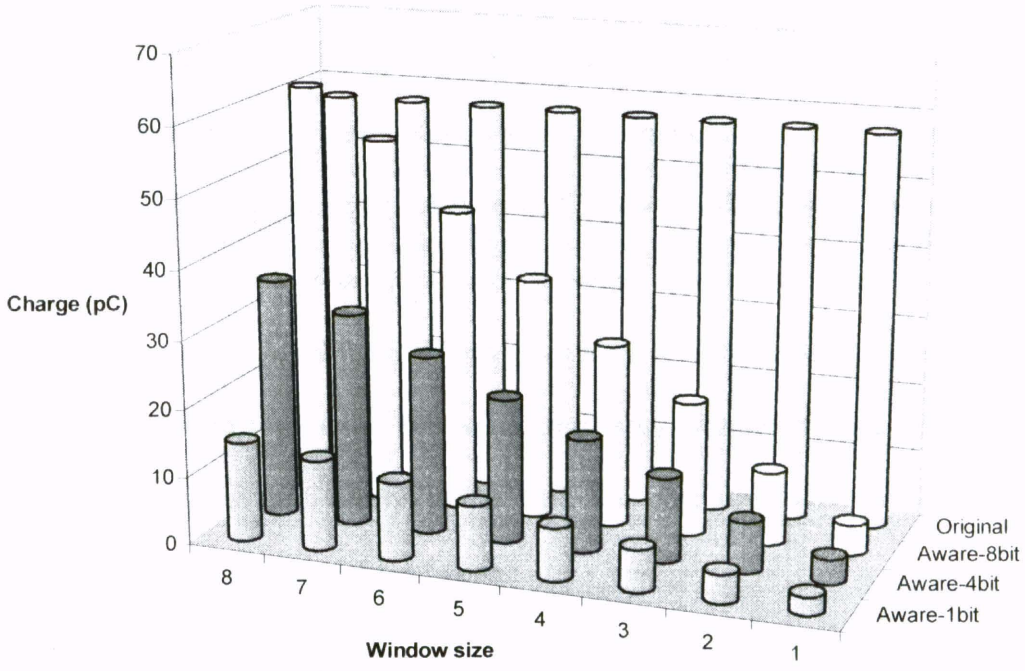


Figure 66. Energy dissipation comparison of rank-order filters

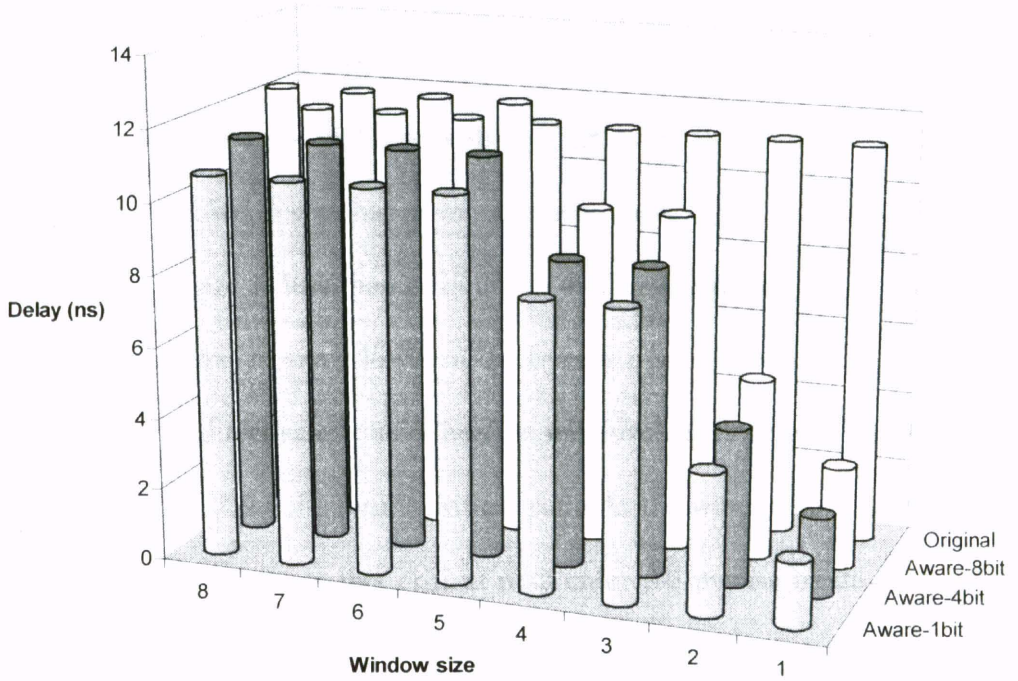


Figure 67. Latency comparison of rank-order filters

In Fig. 66, the *y*-axis is the total charge dissipated over simulation period measured directly by the simulation tool. So the unit is Coulomb. If timed by the supply voltage (3.3V), this figure represents the energy dissipation. In these figures, "Original" refers to data of the original filter; "Aware-8bit", "Aware-4bit", and "Aware-1bit" refer to data of the modified filter with control signal generator under different input wordlengths. The *x*-axis in all three figures is window size. Since the original design has redundant Data 0's in high order bits while the input wordlength changes, its energy dissipation and latency are nearly stable with the changing of wordlength. Several observations can be made from these figures:

1. The modified design has significant energy advantage over the original filter with changing conditions. This is because during the changing of operation mode, the modified design is able to use Null to replace the redundant Data 0's. The average energy saving is 65.4% under an equal input precision probability. Since the additional control circuit is quite simple, the energy dissipation overhead due to additional area cost is very low, only 6.3%.
2. The modified design has advantage over the original filter in latency when the window size is less than or equal to 4. This is because the propagation delay of C&S unit is much lower at the bypass mode than that at the normal mode owing to direct selection. When the window size is greater than 4, the normal-working C&S units form a critical path. As the window size is reduced, one or more C&S units in this critical path enter the bypass mode thus reduces the delay. The overhead of delay is mainly caused by the completion signal generation and selection circuits in delay elements and the output stage. The

average latency reduction is 22.4% under equal probabilities of all conditions, while the largest latency overhead is less than 9%.

4.4 An Energy Macromodel for Designing Energy-aware Multiplier Based On Dynamic Active-bit Detection and Operands Exchange

4.4.1 Overall Scheme

The structure of multipliers implementing Signal Bypassing & Insertion and Zero Insertion technique need a central control unit to detect and set up control. For data sequences with frequently changing input precisions, this structure is not efficient. A dynamic detection scheme should be used to detect the current input precision and send the control signal to the multiplier.

Due to the fact that the detecting circuit will cause additional energy and area, there are two things that need to be determined before the actual designing process begins. The first one is the "resolution" of the detection. In other word, which group of input patterns needs to be detected and controlled? For example, for a 3×3 multiplier, shall 3×2, 3×1, 2×2, 2×1, and 1×1 all be detected, or just detect some of them? The second one is whether to exchange the operands. For example, for pattern 3×2 and 2×3, whether to use an operand exchange circuit to swap the operands and treat both patterns as 3×2, or use separate detecting circuits for each of them?

For the first question, since energy saved and area increased by detecting different patterns are different, a macro model is built to calculate the potential energy saving and area overhead in advance. Based on given energy and area overhead constraints, a group of input patterns are selected, starting from the one has the best energy saving. The

energy/area overhead of each pattern selected is added to the total energy/area overhead. When the total energy/area overhead exceeds the energy/area constraint, the input patterns selected before the current one are detected in the actual energy-aware design.

For the second question, refer to the pipelined array multiplier architecture shown in the Fig. 68 below [47], all bits of the left operand are needed by parallel adder blocks in all rows; but only two bits of the right operand are needed by adder blocks in each row. That means blocking certain unused bits of the right operand brings more energy savings due to the reduction of registers propagating these bits along pipeline stages. Also, the overhead caused by operand exchange circuitry is comparable to the overhead caused by the detecting and control signal generating/propagating circuitries. So the operand exchange scheme is used.

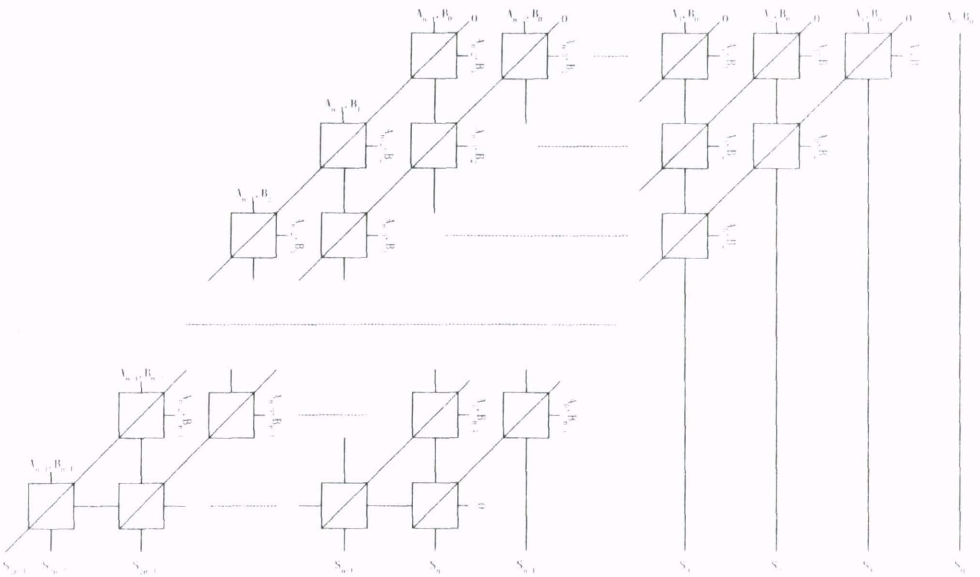


Figure 68. Pipeline array multiplier architecture

The energy-aware multiplier architecture is shown in the Fig. 69 and the overall design flow chart is shown in the Fig. 70.

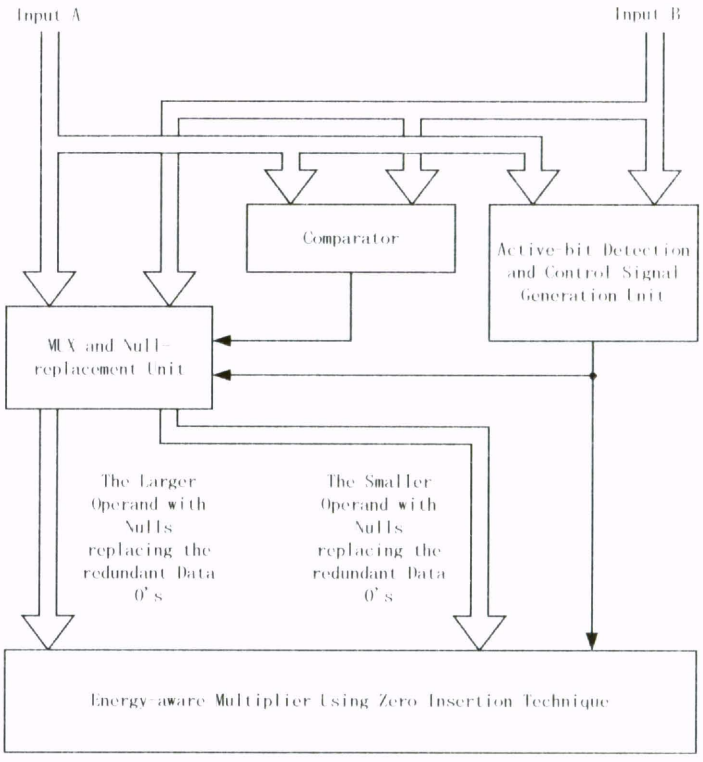


Figure 69. Energy-aware multiplier architecture

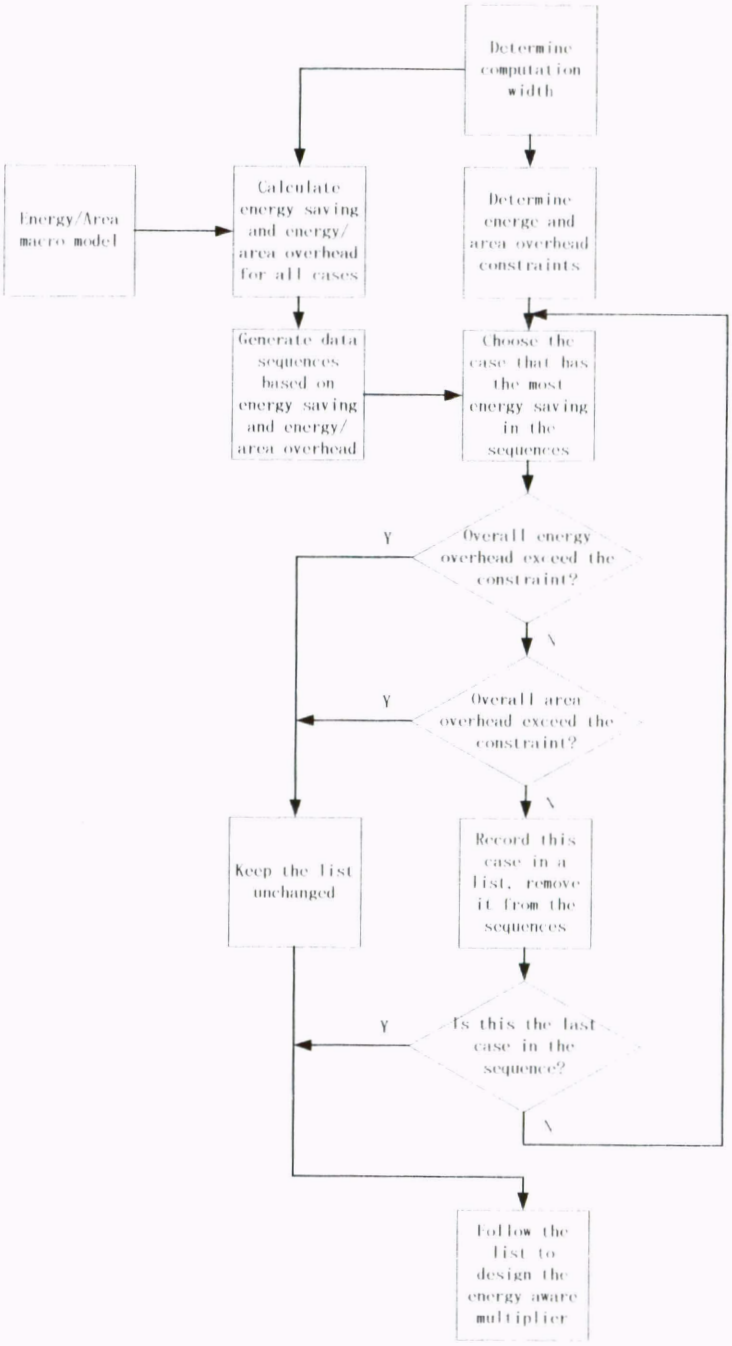


Figure 70. Energy-aware multiplier design flowchart

4.4.2 Energy Macromodel

Building a high-level energy macro model is an essential step in the designing process. Since there are only four types of elements, adder blocks with AND gates (due to the different number of actual inputs, there are totally 6 types of adder blocks), registers, completion detection, and an AND2 gate, in the multiplier, a LUT (Lookup Table) based method is used to build the model. Each element is simulated in all data patterns to calculate the average energy dissipation to build the LUT. Then for each multiplier, the number of each element is calculated. Due to the regular structure of pipeline array multipliers, this calculation can be accurate. Then these numbers are multiplied with the corresponding energy dissipation in the LUT, added up altogether to get the total energy dissipation. All six types of the adder blocks are shown in the Fig. 71.

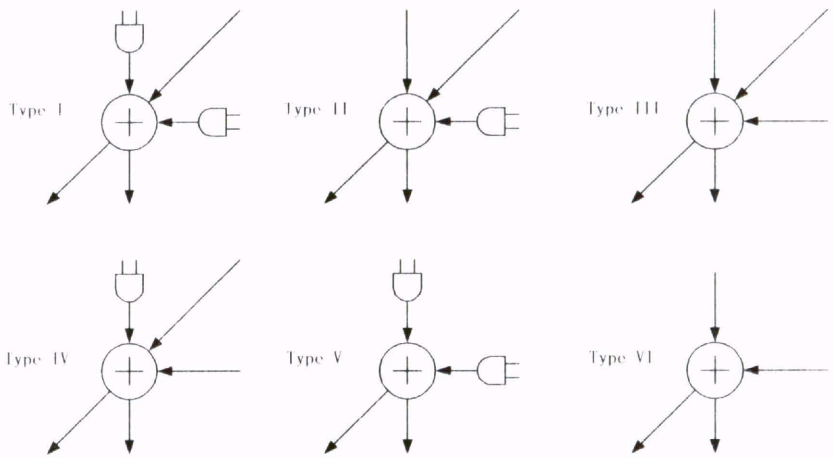


Figure 71. All types of adder blocks used in the multiplier

The LUT is shown in the Table 5. For an $n \times n$ pipeline array multiplier ($n \geq 4$), the numbers of all elements are shown in the Table 6.

Table 5. The lookup table

Element	Average Energy (pJ)	Number of transistors
Adder Block I	218.4188	116
Adder Block II	170.5399	98
Adder Block III	121.3065	80
Adder Block IV	169.7018	98
Adder Block V	180.4163	94
Adder Block VI	72.4455	58
AND2	25.83	18
Register	78.822	28
OR2	21.168	18
XOR	65.898	54
4-bit Comparator	677.07	414
Mux2	65.97	30

Table 6. Count of blocks in an $n \times n$ pipeline array multiplier

Element	Number
Adder Block I	n-2
Adder Block II	$(n-2)^2$
Adder Block I	n-3
Adder Block I	1
Adder Block I	n-1
Adder Block I	1
Register	$\frac{9}{2}n^2 - \frac{9}{2}n + 2$
Completion Detection 2	$2n^2-4n+3$
Completion Detection 3	n(n-1)
Completion Detection 2n-1	n

The detailed equations to calculate the number of each element under different input patterns are developed. These equations can be found in Appendix at the end of the dissertation.

4.4.3 Results And Analysis

The calculation result of normalized energy for all input pattern of a 16×16 multiplier is shown in the Fig. 72. The calculation result of area overhead for detecting

each input precision of a 16×16 multiplier is shown in the Fig. 73. As stated before, an operand exchange scheme is used in the design, so there should be only half parts of the figures above. These parts are just copied to the other side to form a symmetrical picture.

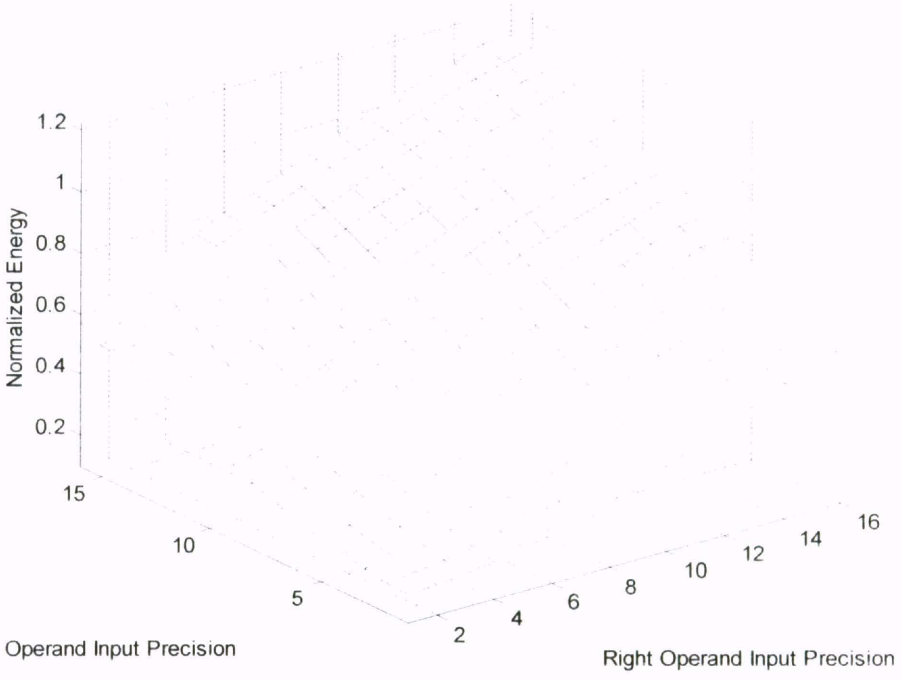


Figure 72. Normalized energy dissipation of a 16×16 energy-aware multiplier

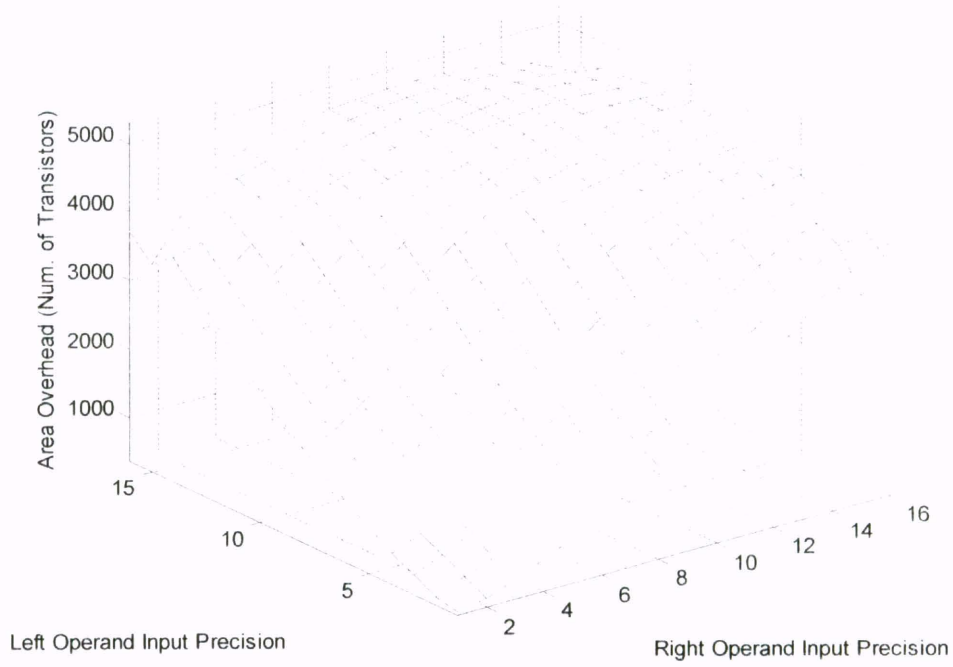


Figure 73. Area occupation of a 16×16 multiplier

From these tables generated from data sequences, the designer is able to choose to detect input patterns starting from the most energy efficient one which also has less area overhead. Then the designer needs to choose between energy efficiency and area overhead, although sometime they are consistent. The model calculates all the information from analytical equations so that the designer is able to make decision in advance based on the energy/area constraints.

A 4×4 multiplier and an 8×8 multiplier were designed and simulated under all input patterns. The energy dissipation and area were recorded and compared to the analytical predictions. The results are shown in the Table 7.

The errors of area are very small for both multipliers. The errors of energy dissipation are larger. These errors mainly come from the short-circuit dissipation of

internal nodes cause by slower rising/falling time of node voltages. Since in a real circuit, the output of a gate needs to drive a number of gates, the parallel capacitance makes the output voltage change slower. Then the short-circuit period of the CMOS gate becomes longer, causing more energy dissipation. In the model, all elements were simulated under a certain input rising/falling time. So there are some differences between the results. Since the designers only need the comparison results for detecting each input precision, the accuracy is good enough.

Table 7. The comparison of model and simulation results

Left Operand Precision	**Right Operand Precision**	**Energy Error**	**Area Error**
4×4			
1	1	10.19%	4.45%
2	1	10.39%	3.66%
2	2	13.75%	3.96%
3	1	10.32%	3.10%
3	2	14.41%	3.85%
3	3	9.90%	2.03%
4	1	13.41%	3.44%
4	2	10.36%	3.02%
4	3	8.72%	2.56%
Maximum Error		14.41%	3.96%
Average Error		11.27%	3.34%
8×8			
1	1	10.98%	1.19%
2	1	13.67%	1.46%
2	2	15.43%	1.60%
3	1	13.76%	1.47%
3	2	18.99%	2.10%
3	3	20.54%	1.73%
4	1	12.93%	2.22%
4	2	15.07%	2.13%
4	3	16.34%	1.86%
4	4	18.17%	1.44%
5	1	13.30%	2.60%

5	2	17.71%	2.19%
5	3	18.76%	1.94%
5	4	19.60%	2.02%
5	5	21.65%	1.48%
6	1	14.52%	2.56%
6	2	17.67%	2.39%
6	3	20.49%	2.14%
6	4	20.63%	1.66%
6	5	20.76%	1.52%
6	6	19.40%	1.10%
7	1	15.41%	2.45%
7	2	19.12%	1.96%
7	3	20.20%	1.55%
7	4	19.33%	1.55%
7	5	17.99%	1.36%
7	6	14.93%	1.42%
7	7	13.22%	1.10%
8	1	16.69%	2.28%
8	2	16.31%	1.99%
8	3	17.26%	1.79%
8	4	15.53%	1.22%
8	5	15.26%	1.23%
8	6	12.57%	1.27%
8	7	8.90%	0.95%
Maximum Error		21.65%	2.56%
Average Error		16.66%	1.59%

CHAPTER FIVE: ESTIMATING ENERGY DISSIPATION

5.1 Analytical Input Mapping for Modeling Energy Dissipation of Complex CMOS Gates

Power estimation could be at the logic level, at the architecture (register-transfer) level and at the behavior level. High-level power estimation techniques are fast, but produce less accurate results. On the other hand, power estimation at the transistor or gate level is more accurate, but computation time is significant. Recently, a power modeling technique using the equivalent inverter to evaluate power and switching delay was presented [59]. It uses an equivalent inverter to replace the whole CMOS gate. By carefully selecting the input of this inverter due to the current input pattern of the original gate, this inverter could dissipate the same amount of energy as the CMOS gate does. This technique is computation efficient, while maintaining high accuracy as mathematical analysis. The technique in [59], however, only accounts for input mapping for a serial MOSFET chain. Parallel structure was not analyzed. Jun et al., [60] developed an input mapping algorithm for a parallel structure, but they did not account for all possible inputs.

In this section, all possible input patterns on parallel CMOS structures are analyzed. The input patterns including slow, fast and very fast input ramps are considered. The input mapping and calculation of equivalent transistor size are correlated. The equivalent inverter modeling technique for complex CMOS gates including feedback is

also presented. All modeling results are compared by Cadence SPICE simulation to validate the model utility and accuracy.

5.1.1 Notation

m: The ratio of the channel width between M_b and M_a

$t_{ha,b,c}$: The starting-to-conduct time of M_a, M_b and M_c

$t_{a,b,c}$: The time when $V_{ina,b,c}$ arrives

$t_{la,b,c}$: The time when M_a, M_b and M_c, respectively, goes to linear region

$t_{sa,b,c}$: The time when M_a, M_b and M_c, respectively, goes to saturation region

$\tau_{a,b,c}$: Rising time of $V_{ina,b,c}$, respectively

V_{dd}: Supply voltage

$V_{ina,b,c}$: Input ramps for M_a, M_b and M_c, respectively

V_{TH}: Threshold voltage

$W_{a,b,c}$: The channel width of M_a, M_b and M_c, respectively

5.1.2 Input mapping for slow inputs on parallel structure

The MOS transistor model used in this section is the α-power law model [61]. For deep-submicron MOS transistor (channel length is less than 0.5μm), the value of α is around 1 [61]. In the calculations below, α is assumed to be 1. Figure 74 illustrates the mapping of the input for two parallel transistors to an equivalent single transistor.

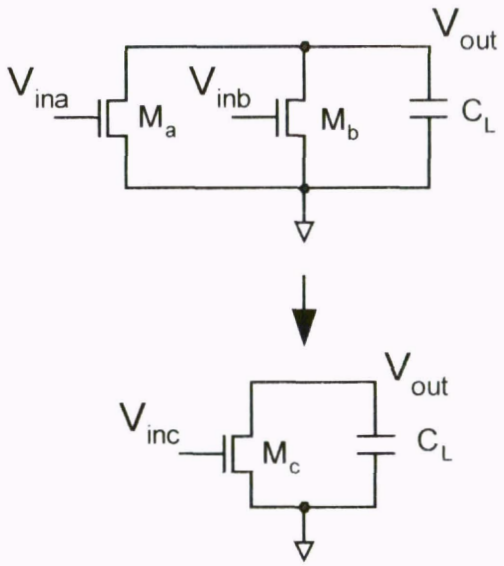


Figure 74. Input mapping example

The energy dissipation of the equivalent transistor M_c must equal to that of M_a and M_b together. The integration of current flowing through those transistors over the whole input changing time gives the total charge dissipation as

$$\int_0^{\tau_a} I_{DS1} dt + \int_0^{\tau_b} I_{DS2} dt = \int_0^{\tau_c} I_{DS3} dt \qquad (15)$$

where τ_a, τ_b, and τ_c are the rising times for the input to M_a, M_b, and M_c, respectively. Note that in this section, only charging/discharging energy is considered.

Three input conditions, slow input, fast input, and very fast input, are discussed during the calculation. The definitions are based on the status of M_a and V_{out} at the time the input ramp ends. Under each input condition, the division of the calculation time windows is different. The definition of slow input is that the rising ramp of input is slower than the falling ramp of V_{out}. If one uses t_0 to represent the time when V_{out} falls to

zero, then $t_1 < t_0 < \tau$, where t_1 is the time when transistor goes into the linear region and τ is the input rising time.

5.1.2.1 Case 1: V_{ina} and V_{inb} are both rising ramps, $t_a = t_b$, and $\tau_a = \tau_b$

$V_{ina,b,c}$ and $t_{a,b,c}$ are the input ramps and the time when $V_{ina,b,c}$ arrives for M_a, M_b and M_c, respectively. In this case, V_{inc} is the same as V_{ina} and V_{inb}. W_c is equal to the sum of W_a and W_b or $(1+m)$ times W_a, where m is the ratio of the channel width between M_b and M_a, and $W_{a,b,c}$ are the channel width of M_a, M_b and M_c, respectively.

5.1.2.2 Case 2: V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{ha} < t_{hb} < t_{la}$

Parameters t_{ha}, t_{hb} and t_{hc} are the starting-to-conduct times of M_a, M_b and M_c, and t_{la} is the time when M_a goes into linear region. The transient response of voltages and currents are shown in Figure 75.

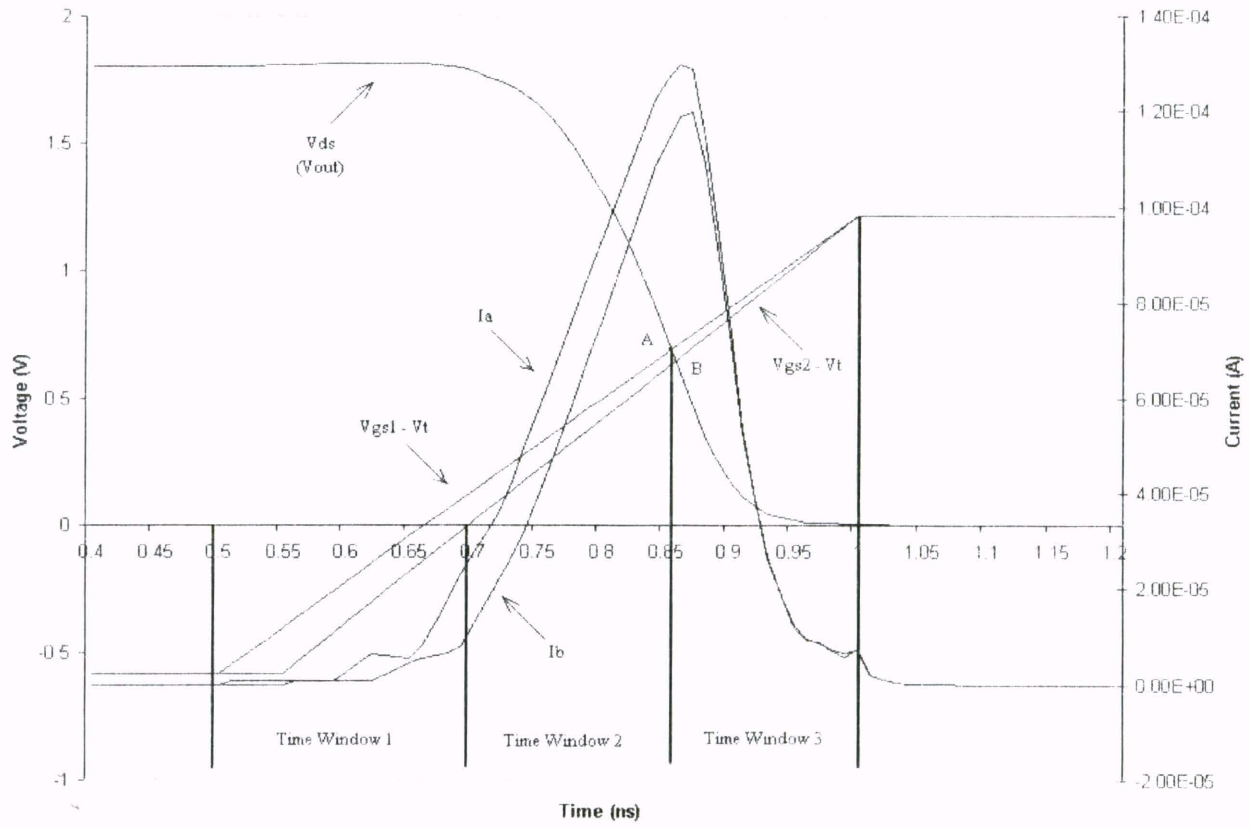


Figure 75. Waveforms of voltages and currents in Case 2

It can be divided into three time windows as follows:

A. Time window 1: $0 \le t \le t_{hb}$

In calculating t_{ha} and t_{hb}, since all inputs are linear and when a transistor starts to conduct current, $V_{GS} = V_{TH}$, t_{ha} and t_{hb} are given by $t_{ha} = V_{TH}\tau_a / V_{dd}$ and $t_{hb} = V_{TH}\tau_b / V_{dd} + t_b$. In this time window, M_b is in the cutoff region and M_a is in the saturation region. The total charge Q_1 dissipated is calculated as $\int_{t_{ha}}^{t_{hb}} I_{sa}\,dt$, where I_{sa} stands for the saturation current in M_a.

B. Time window 2: $t_{hb} \le t \le t_{la}$

t_{la}, t_{lb} and t_{lc} are the time when M_a, M_b and M_c go to the linear region, respectively. To calculate t_{la}, using Kirchoff's law, V_{out} is obtained from

$$C_L \frac{dV_{out}}{dt} = -(I_{sa} + I_{sb}) \tag{16}$$

For the linear input, V_{out} is given by

$$V_{out} = A_1 t^2 + B_1 t + C_1 \tag{17}$$

where $A_1 = -\frac{k_s}{C_L}\frac{V_{dd}}{2}\frac{\tau_b + m\tau_a}{\tau_a \tau_b}$, $B_1 = \frac{k_s}{C_L}\left[(m+1)V_{TH} - \frac{mV_{dd}}{\tau_b}t_b\right]$, and C_1 is solved using

$$C_L \frac{dV_{out}}{dt} = -I_{sa} = -k_s\left(\frac{V_{dd}}{\tau_a}t - V_{TH}\right) \tag{18}$$

From simulation data in Figure 2, an approximation can be made that the relationship between $V_{GS} - V_{TH}$ and V_{out} can still be used as the division of linear region and saturation region. So at $t = t_{la}$

$$V_{out} = V_{ma} - V_{TH} = \frac{V_{dd}}{\tau_a}t - V_{TH} \tag{19}$$

Equations (17) and (19) together yield t_{la}. The total charge Q_2 dissipated in this time window is $\int_{t_{hb}}^{t_{la}} (I_{sa} + I_{sb})dt$, where I_{sa} and I_{sb} stands for the saturation current in M_a and M_b, respectively.

C. Time window 3: $t_{la} \le t \le t_b + \tau_b$

In this window, as it can be seen from Figure 75, the time point Ma exits the saturation region (denoted by point A) and the time point Mb exits the saturation region (denoted by point B) are very close. The reason is in Case 2, from the criteria $t_{ha} < t_{hb} < t_{la}$, V_{inb} is not very far behind V_{ina}. That means at each time point, V_{inb} is only slightly smaller than V_{ina}. So when $V_{ina} - V_{TH} = V_{out}$ and M_a exits the saturation region, $V_{inb} - V_{TH}$ is also very close to V_{out}. After a very short time, $V_{inb} - V_{TH}$ will become equal to V_{out} and M_b will exit the saturation region. Since M_a and M_b almost exit the saturation region simultaneously, one can assume that $t_{la} = t_{lb}$. When V_{out} goes to zero at time t_0, the drain voltage of M_a and M_b is zero. No current flows through them afterward. Q_3 is calculated by integrating the linear currents from t_{la} to t_0.

In this time window M_a and M_b are in the linear region. Since V_{out} is given by $C_l \frac{dV_{out}}{dt} = -(I_{la} + I_{lb})$ and $V_{out} = \frac{V_{dd}}{\tau_a} t_{la} - V_{TH}$ at $t = t_{la}$, letting $V_{out} = 0$ one gets t_0. But as V_{out} approaching zero, it becomes more and more nonlinear and logarithmetic. After V_{out} goes down to a small value above zero, the current flowing through transistors is very small and doesn't affect the estimation accuracy much. So depending on the accuracy required, a small positive value can be used instead of zero as the end of this time window. In this shortened time window, as one can see from Figure 75, I_{la} and I_{lb} can be assumed as linear to simplify the calculation. This linear approximation was also used in [59]. The simulation results show that it doesn't affect the accuracy much. I_{la} and I_{lb} are given by $I_{la} = (t - t_0) I_{la0} / (t_{la} - t_0)$ and $I_{lb} = (t - t_0) I_{lb0} / (t_{la} - t_0)$. The charge Q_3 dissipated in this

window is $\int_{t_{la}}^{t_0}(I_{la}+I_{lb})dt$. The total charge in these three time windows is $Q_{ab}=Q_1+Q_2+Q_3$.

We now proceed to calculate the equivalent input. The size of the equivalent transistor M_c is the sum of M_a and M_b. i.e., $W_c = W_a + W_b = (1+m)W_a$. For calculation of Q_c, the rise time τ_c is used as a variable. Letting $Q_c = Q_{ab}$, one obtains τ_c.

In the saturation region

$$C_L\frac{dV_{out}}{dt} = -I_{sc} = -(1+m)k_s\left(\frac{V_{dd}}{\tau_c}t - V_{TH}\right) \tag{20}$$

At $t = t_{lc}$ M_c enters the linear region. Inserting V_{out} at t_{lc} into (20), one solves for t_{lc} as a function of τ_c and $t_{hc} = V_{TH}\tau_c / V_{dd}$.

In the linear region, $C_L\frac{dV_{out}}{dt} = -I_{lc}$. At $t = t_{lc}$, $V_{out} = \frac{V_{dd}}{\tau_c}t_{lc} - V_{TH}$. Since V_{out} is a function of τ_c, the method used to calculate M_a and M_b needs to be modified. The Taylor series of V_{out} at $t = t_{lc}$ is useful

$$V_{out} = V_{out}\Big|_{t=t_{lc}} + V'_{out}\Big|_{t=t_{lc}}(t - t_{lc}) \tag{21}$$

Setting V_{out} in (21) at zero gives t_{0c}. Now M_c is in the linear region and I_{lc} is approximated from the linear extrapolation. Using the current value at $t = t_{lc}$ as the initial value, I_{lc} is expressed as $I_{lc} = (t - t_{0c})I_{lc0}/(t_{lc} - t_{0c})$. The total charge Q_c dissipated in M_c is calculated as $\int_{t_{hc}}^{t_{lc}} I_{sc}\,dt + \int_{t_{lc}}^{t_{0c}} I_{lc}\,dt$.

5.1.2.3 Case 3: V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{la} < t_{hb} < \tau_a$

The calculation for this case is similar to that for Case 2. The only major difference is the division of the time windows because of the changing of the criteria. Since V_{inb} is further lagged behind V_{ina}, the time window 1 should be $0 \leq t \leq t_{la}$. In this window M_a is in the saturation region and M_b is in the cutoff region. The time window 2 should be $t_{la} \leq t \leq t_{hb}$. In this window M_a is in the linear region and M_b is in the cutoff region. The time window 3 should be $t_{hb} \leq t \leq \tau_a$. In this window M_a is still in the linear region. When M_b slightly conducts, M_b is in the saturation region. Same calculation steps in Case 2 can be used in this case. Calculate the total charge Q'_{ab} in M_a and M_b first. Let $W_c = W_a + W_b = (1+m)W_a$ and V_{inc} has the same starting time as V_{ina}. Equate the total charge Q'_c in the equivalent transistor M_c to Q'_{ab} to calculate τ_c, the rising time of the mapped input V_{inc}.

5.1.2.4 Case 4: V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{hb} > \tau_a$

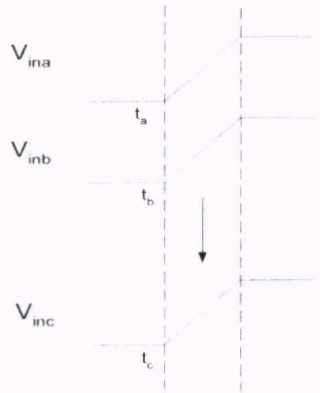
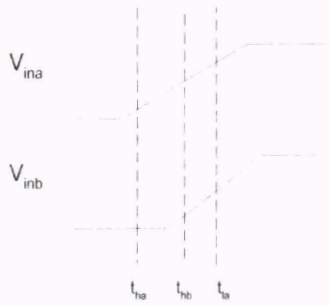
In this case when V_{inb} comes, M_a has been fully conducting, V_{out} is close to zero. So there is not much current flowing through M_b during its input rising process. The current flowing through M_a is considered. Therefore, $W_c = W_a$ and $V_{inc} = V_{ina}$.

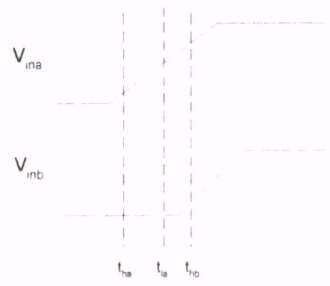
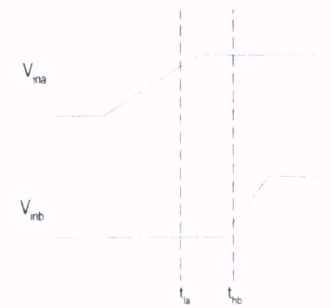
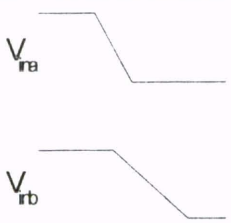
5.1.2.5 Case 5: V_{ina} and V_{inb} are falling ramps

In this case, M_a and M_b are all shutting down. CMOS structure is symmetric of pMOS side and nMOS side. When nMOS transistors are shutting down, there must be some pMOS transistors are starting to conduct. Comparing these two processes, conducting process dominates the total charge due to charge/discharge the load capacitance C_L. Therefore, only pMOS transistors are used in analysis.

The summary of the input mapping of all the cases is in Table 8.

Table 8. Summary of the input mapping of all possible cases

	Definition	Input Mapping	Transistor Size Calculating
Case 1	 ***V_{ina} and V_{inb} are both rising ramps, t_a=t_b, and τ_a=τ_b***	V_{inc} is the same as V_{ina} and V_{inb}	$W_c = W_a + W_b$
Case 2	 ***V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{ha}<t_{hb}<t_{la}$***	Use the algorithm in the text to calculate V_{inc}	$W_c = W_a + W_b$

Case 3	 ***V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{la}<t_{hb}<\tau_a$***	Use the algorithm in the text to calculate V_{inc}	$W_c = W_a + W_b$
Case 4	 ***V_{ina} and V_{inb} are rising ramps, $\tau_a \neq \tau_b$, and $t_{hb}>\tau_a$***	V_{inc} is the same as V_{ina}	$W_c = W_a$
Case 5	 ***V_{ina} and V_{inb} are falling ramps***	Calculate pMOS side	Calculate pMOS side

5.1.3 Input mapping for fast and very fast inputs on parallel structures

The definition of fast input is that when V_{ina} reaches V_{dd}, M_a is already in the linear region, i.e., $t_{ha} < t_{la} < \tau_a < t_{0a}$ (t_{0a} is the time when V_{out} reaches zero). In this situation, M_a enters the saturation region, and then the linear region. For fast input mapping the same method in time windows 1 and 2 is used to calculate Q_1 and Q_2 in Case 2. During the time window 3 V_{ina} and V_{inb} remain at V_{dd} after τ_a and τ_b, respectively. V_{out} is modified to calculate t_0. I_{la} and I_{lb} are extrapolated linearly during $\tau_a < t < \tau_b$ and $\tau_b < t < t_0$ to calculate Q_3. In Case 3 if $t_{hb} < \tau_a$, the same method during the time window 2 is used to calculate Q_2. If $t_{hb} > \tau_a$, the upper limit of the integration is modified

The definition of very fast input is when $t = \tau_a$, M_a is still in the saturation region. i.e., $\tau_a < t_{la} < t_0$. In Case 2, there are two sub-cases: when $t_{hb} < \tau_a$, the same method in slow input is used to calculate Q_1. After getting V_{out} in (17), Q_2 is computed by changing the upper limit of the integration to τ_a. Then use $I_{sa} = ks(V_{dd} - V_{TH})$ in (18) to calculate t_{la}. If $t_{la} < \tau_b$, change I_{sb} in (16) to $I_{sb} = mks(V_{dd} - V_{TH})$ to refine t_{la}. After getting t_{la}, if $t_{la} < \tau_b$ use $\int_{\tau_a}^{t_{la}} (I_{sa} + I_{sb})dt$ to calculate Q_3. Note that $I_{sa} = ks(V_{dd} - V_{TH})$. Use the linear approximation to determine I_{la} and I_{lb} and $\int_{t_{la}}^{\tau_b} (I_{la1} + I_{lb1})dt + \int_{\tau_b}^{t_0} (I_{la2} + I_{lb2})dt$ to calculate Q_4. If $t_{la} > \tau_b$, use $\int_{\tau_a}^{\tau_b} (I_{sa} + I_{sb})dt$ to calculate Q_3 and $\int_{t_{la}}^{t_0} (I_{la} + I_{lb})dt$ to calculate Q_4. The total mapped charges equal $Q_1 + Q_2 + Q_3 + Q_4$. For the other sub-case, $t_{hb} > \tau_a$, change the

upper integration limit to τ_a while calculating Q_1. Then calculate t_{la} considering the value of V_{out} at $t = t_{hb}$.

5.1.4 Modeling complex CMOS gates including feedback

In modeling CMOS gates virtually all publications in the past focused on inverter, NAND and NOR gates. Circuit schematics such as flip-flops or latches have feedback from output to input. Here, the NCL threshold gates are selectedas an example. The NCL gate shown in Fig. 76 is Th23x0. It contains a feedback structure.

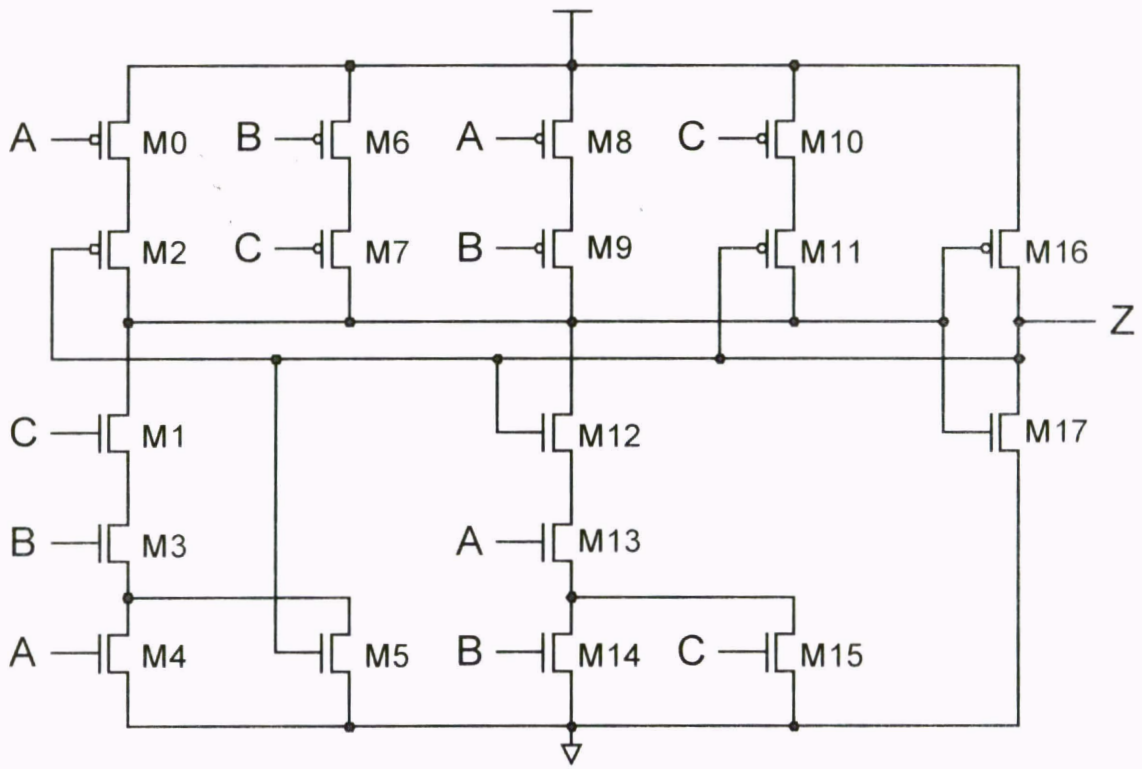


Figure 76. Circuit structure of a threshold gate

To break the feedback from the output Z to internal nodes, a new input Z from the previous output is added to each input node as displayed in Fig. 77. The circuit in Fig. 77

is then modeled as two cascaded inverters as shown in Fig. 78. The first inverter in Fig. 78 represents all transistors in Fig. 77 except the output transistors M16 and M17. The second inverter in Fig. 78 represents the output inverter in Fig. 77.

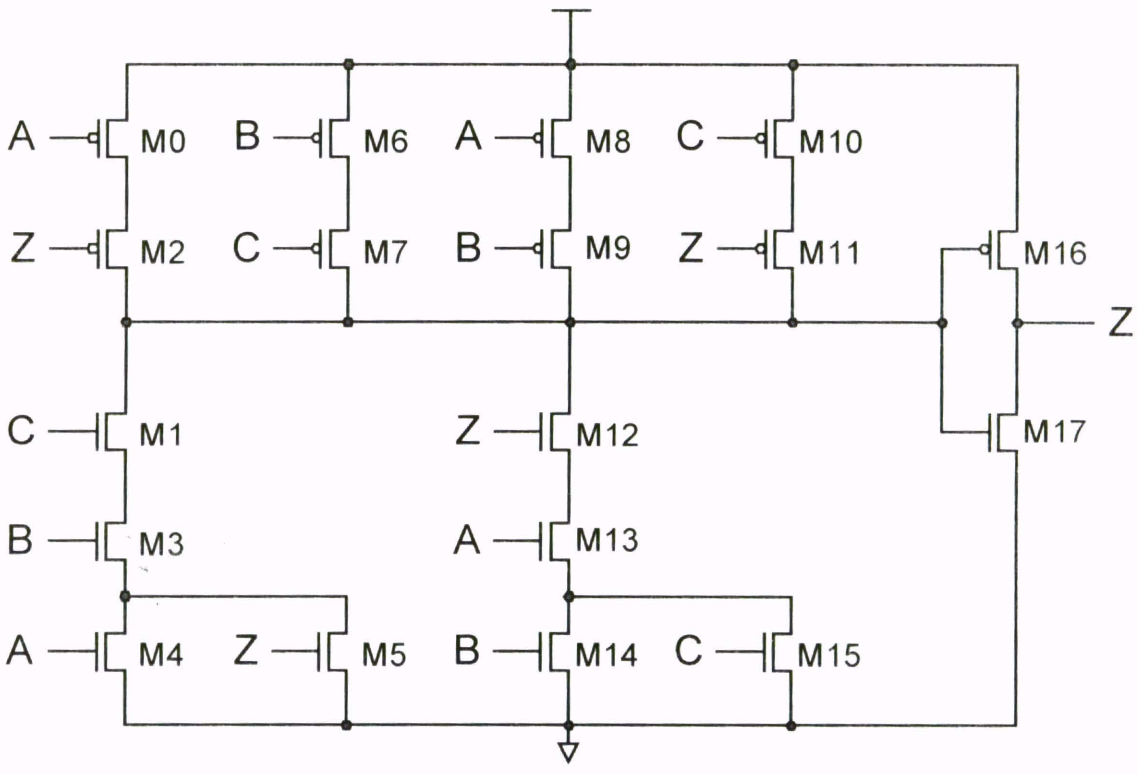


Figure 77. Equivalent structure of Fig. 3 after breaking the feedback

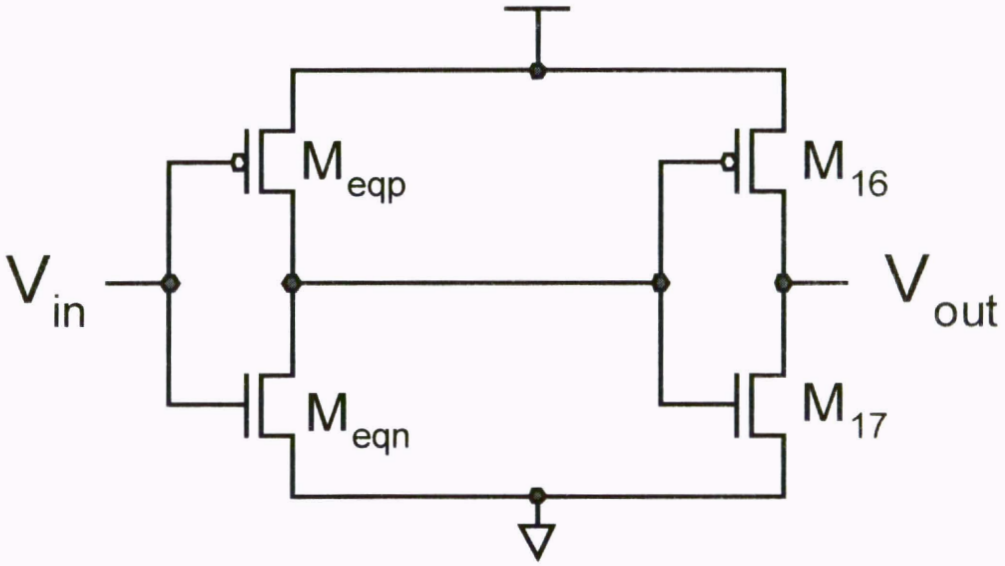


Figure 78. Two-stage equivalent inverters

When inputs *A*, *B* and *C* are rising from zero to V_{dd}, the output *Z* goes up from zero to V_{dd} also. M1, M3, M4 and M5 are conducting. There are three transistors vertically to form signal paths. The effective channel resistance or delay is roughly three times that of one transistor. Since *Z* is still zero when the transition begins, M12 is in the cutoff region. Transistors M12, M13, M14, and M15 do not make any contribution. Transistor M16 at the last stage is also included. This results in the delay four times that of a single transistor. Accounting for the effect from the pMOS transistors side, we may add another delay for estimation. Measuring delay due to rising *A*, *B* and *C* inputs and multiplying by five give the final effective gate delay. Though the delay estimation is rough, it is sufficient for the equivalent inverter to estimate power dissipation due to feedback. The reason is for the two inputs with different arriving times in Figure 74, the contribution of the later one to the total charge dissipation is much less than the former one when it arrives late enough.

5.1.5 Simulation results

The modeling techniques presented in Section 5.1.3 and 5.1.4 are evaluated by Cadence SPICE simulation. The transistor parameters used include 0.24μm channel length and some BSIM parameters used in simulation are shown in Table 9.

Table 9. BSIM parameters used in simulation

nMOS				pMOS			
t_{ox}	4.08E-09	x_j	1.6E-07	t_{ox}	4.23E-09	x_j	1.7E-07
V_{TH0}	0.5187622	N_{CH}	2.2205E+17	V_{TH0}	-0.435230	N_{CH}	4.9898E+17
U_0	0.0374658	T_{NOM}	25	U_0	0.01	T_{NOM}	25

Besides NAND, NOR gates and AOI, OAI gates, several threshold gates with feedback structure are also tested. The simulation result of Th23x0 gate with the changing of the load capacitance is shown in Figure 79 with inputs selected in Case 2. The data from original inputs (solid lines with transparent squares) and mapped equivalent input (discrete solid triangles) are compared. Figure 79 demonstrates good agreement between the results using the original inputs and the mapped input. Simulation results of other gates are listed in Table 10. Arbitrary input combinations from the first four cases are selected and simulated. The comparison in Table 10 also shows good modeling accuracy with small error percentage.

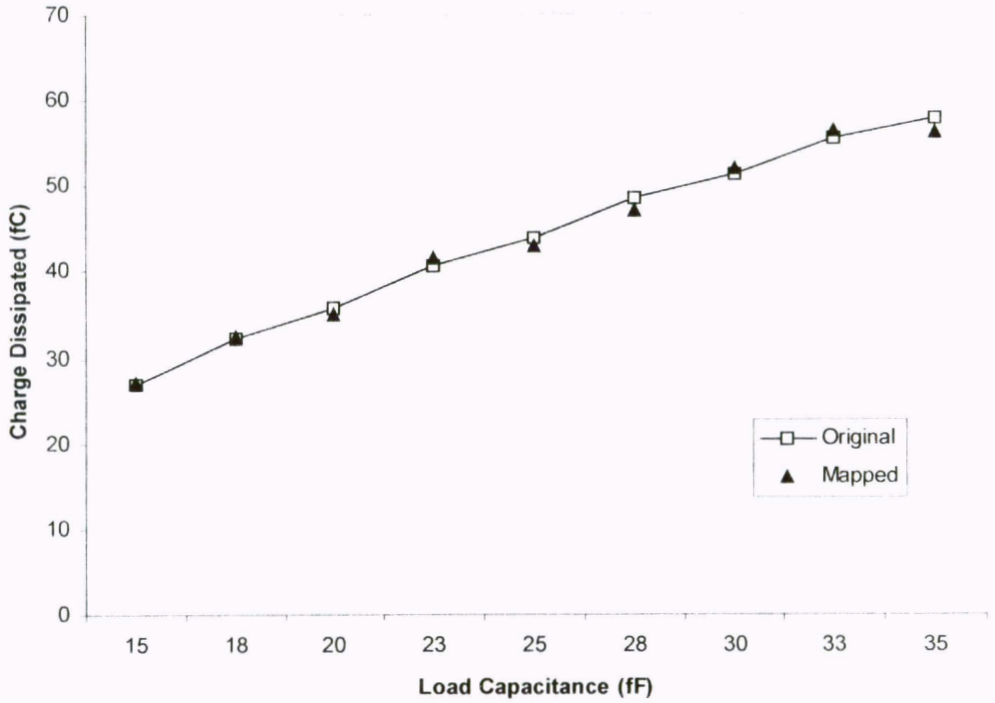


Figure 79. Simulation results of Th23x0 with changing of load capacitance

Table 10. Simulation results comparison of chosen gates

Name	Charge dissipated in gate under original inputs (fC)	Charge dissipated in equivalent inverter under mapped input (fC)	Error percentage
NAND4	1.864	1.87	0.3%
NOR4	3.55	3.49	1.7%
AOI44	9.66	9.81	1.55%
AOI333	9.84	9.95	2.95%
AOI33	9.41	9.37	0.43%
AOI322	9.55	9.49	0.63%
AOI32	9.35	9.33	0.21%

OAI44	9.006	9.02	1.55%
OAI43	8.99	9.01	0.22%
OAI333	9.12	9.15	0.33%
OAI33	8.74	8.82	0.92%
OAI32	8.65	8.74	1.04%
Th22x0	37.37	36.98	1.04%
Th34w2x0	31.86	32.81	2.98%
Th33x0	33.65	34.08	1.28%
Th44x0	30.19	29.87	1.06%

5.2 Switching Activity Modeling of Multi-rail Speed-independent Circuits — A Probabilistic Approach

In gate level, circuit behavior is modeled by statistical information of the signal at each node, such as signal probability and transition density. When the circuit is large and only the overall power dissipation information is needed, the probabilistic power estimation technique in gate level is appropriate.

5.2.1 Modeling switching activity in speed-independent circuit

5.2.1.1 Zero-delay model

A major difficulty of modeling power dissipation in synchronous circuits is to model glitch power. Glitch power comes from the different arriving time of input signals to a gate. The existence of glitch power makes gate propagation delay play an important

role on circuit behavior. The modeling technique becomes very complex if real gate delays are considered. On the contrary, speed-independent circuits have the property that whatever the gate delays are; the behaviors of the circuits are the same. There is no glitch power. This property greatly simplifies the modeling technique. A zero-delay model can be used instead of real gate delay. This model assumes that the propagation delays of all gates are zero. That means all the inputs of a gate can be assumed to arrive at the same time. Under this assumption, the behavior of the circuit is the same as that is under real gate delay.

5.2.1.2 Occurrence probability

In synchronous circuit, signal probability at a node X is defined as the average fraction of clock cycles in which the steady state value of X is logic high [5]. This term is able to reflect the characteristics of the logic high occupation time of a signal. In speed-independent circuit, since the gate delays do not affect circuit behaviors, the logic high occupation time cannot describe the signal behavior. As shown in Figure 80, a speed-independent circuit has performed the same operation twice. Due to the different gate delays, it has two different transient waveforms. Although the signal probabilities are obviously different, the data sequences as well as the number of transitions are the same.

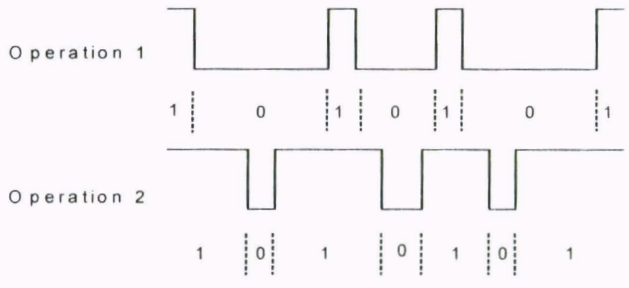


Figure 80. Same number of logic high occurrences in different occupation time

To describe the signal behavior in speed-independent circuits, a new statistical term is needed. It can be easily observed that the two waveforms in Figure 80 have the same number of logic high occurrence: four ones out of seven data. It is the number of occurrence instead of occupation time of logic high that is important to model the circuit behavior. A new term of signal statistical characteristics, Occurrence Probability, is defined as below.

Definition 1: The Occurrence Probability, denoted as OP, of a signal X is defined as

$$OP_X = \lim_{N\to\infty} \frac{N_H}{N} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{22}$$

where N is the total number of data states, x_i is the ith data value of node x, N_H is the number of logic high occurrence in N. Several properties of occurrence probability are discussed below.

1) Existence

The existence of occurrence probability can be illustrated as below:

$$\Delta_{N\to N+1} = \frac{\sum_{i=1}^{N+1} x_i}{N+1} - \frac{\sum_{i=1}^{N} x_i}{N} = \frac{\sum_{i=1}^{N} x_i + x_{N+1}}{N+1} - \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_{N+1}}{N+1} - \frac{\sum_{i=1}^{N} x_i}{N(N+1)}$$

Since x_i is bounded as 0 or 1, when N approaches infinity, the equation above equals to zero. So the limitation in Equation (22) exists.

2) Relationship with signal probability

As stated above, in synchronous circuit, signal probability at a node X is defined as the average fraction of clock cycles in which the steady state value of X is logic high [5]. Compared with the definition of occurrence probability, signal probability is a strict-

sense occurrence probability when the duration time period of each Data state is the same. Under this assumption the asynchronous circuit acts in the same way as synchronous circuits. So occurrence probability is more general. Actually, it can be used in synchronous circuits as well.

3) Relationship with transition probability

Transition probability at a node X is defined as the average fraction of clock cycles in which the steady state value of X is different from its initial value [5]. It seems that occurrence probability and transition probability are not related. But as shown later in this section, because of the Data-Null sequence behavior of multi-rail speed-independent circuits, direct relationship between these two terms is established. Since in Null state all nodes are in logic low so that no two Data 1 states can be adjacent to each other, transition probability is equal to occurrence probability if Data-Null sequence is considered and the duration time period of each Data and Null state is the same.

4) Relationship with transition density

Transition density is defined as the average number of transitions at a node X during unit time period [5]. This term characterizes circuit transition behavior in continuous system. Occurrence probability characterized circuit behavior in discrete system. As shown later in this section, if Data-Null sequence is considered and the duration time period of each Data and Null state is the same, occurrence probability is twice of transition density.

In this section, occurrence probability is used to model the switching activities of the signals in speed-independent circuits.

5.2.2 Modeling multi-rail signal

Multi-rail encoding is widely used in gate-level speed-independent circuits. As introduced before in this dissertation, multi-rail logic uses at least two wires to interpret one signal value. The simplest multi-rail encoding is dual-rail logic, in which two wires are used for one signal. There are two valid states of a signal in dual-rail encoding: One is Data, including 0 and 1; the other is called Null or Spatial. The truth table of dual-rail logic was shown in Table 1 in Chapter 2. From Table 1 it is easy to see that wire0 and wire1 are complementary when the signal is Data. If the occurrence probability of the signal during Data state is known as OP_S, the occurrence probability of wire0 and wire1 during Data state, denoted as OP_0 and OP_1, can be achieved as

$$\begin{aligned} OP_1 &= OP_S \\ OP_0 &= 1 - OP_1 = 1 - OP_S \end{aligned} \tag{23}$$

Other multi-rail encoding can be modeled similarly. For simplicity, in this section, only dual-rail logic is analyzed. In the rest of this section, the word "signal" means the logic value interpreted by both rails, and "wire" means the value of one rail.

5.2.3 Modeling Data-Null cycle

The operation of multi-rail logic circuits contains Data-Null cycles. After a Data state, all signals go to a Null state, and then go to next Data state. For a long operating sequence, the number of Data states is equal to the number of Null states. For dual-rail logic, both wires will become 0 in a Null state, and then one of them will become 1 in the next Data state. So the OP of any signal/wire in total operating states including Data and Null states are simply one half of the OP in Data states. For simplicity, in this section, the

OP of a signal/wire means its OP in Data states only. To calculate switching activity from OP, a theorem is given as below.

Theorem1: The switching activity of both wires of a signal can be calculated directly from their occurrence probabilities:

$$\begin{aligned} SW_1 &= 2 \times OP_1 \times N = 2 \times OP_S \times N \\ SW_0 &= 2 \times OP_0 \times N = 2 \times (1 - OP_S) \times N \end{aligned} \tag{24}$$

where SW_1 and SW_0 are the number of switching of wire1 and wire0, respectively, N is the total number of Data applied.

The proof of this theorem is simple. Since After a Null state, a Data state comes, if one of the two wires is logic high in this Data state, it will transit from 0 to 1 because in previous Null state it must be logic low. Also since there is a Null state following this Data state, this wire will transit from 1 to 0 while entering the Null state. For each logic high occurrence, there are two transitions. So the total number of switching can be calculated using Equations (24) above.

5.2.4 Occurrence Probability Propagation

To derive the OP propagation algorithm, an example of speed-independent circuit, NCL is used as logic example. The hysteresis property of threshold gates makes them behave as finite state machines (FSM). For example, Th23x0 is a threshold gate, its output will become logic high when at least two of the three inputs are logic high, and will become logic low when all three inputs are logic low. Otherwise, the output will remain unchanged from its previous value. Its FSM description is shown in Figure 81.

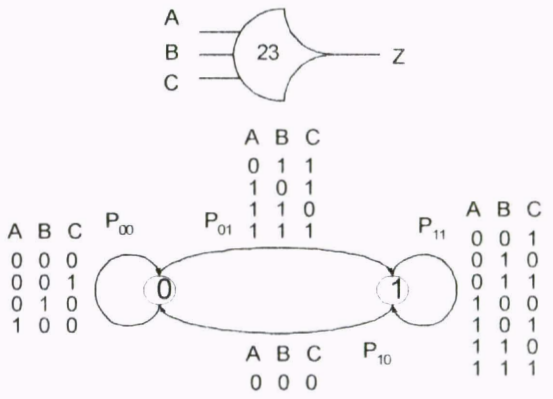


Figure 81. FSM description of Th23x0

In Figure 81, all the four lines are wires. Given the OPs of inputs, to calculate the OP and SW of output Z, consider in a Null state, all inputs are logic low, from Figure 81 it is clear that no matter what the value of Z in previous Data state is, Z must be zero. When next Data state comes, Z could become logic high or remain zero. From Figure 81, if the independence between inputs is assumed, the probability of Z to become logic high, P_{01}, and the probability to remain logic low, P_{00}, can be achieved using the OPs of inputs:

$$\begin{aligned} P_{01} &= (1-OP_A)\times OP_B \times OP_C + OP_A \times (1-OP_B)\times OP_C + OP_A \times OP_B \times (1-OP_C) + OP_A \times OP_B \times OP_C \\ P_{00} &= 1 - P_{01} \end{aligned} \tag{25}$$

When next Null state comes, all inputs and output will definitely become logic low. That means the value of Z is determined by P_{01} and P_{00} only. Because OP_Z is how many logic highs in a certain amount of input data, OP_Z is equal to P_{01}:

$$OP_Z = P_{01} \tag{26}$$

The switching activity SW_Z can be calculated using Equation (24):

$$SW_Z = 2 \times OP_Z \times N \tag{27}$$

The propagation of OP and SW through other threshold gates can be achieved in the similar way.

5.3 Modeling signal correlation

5.3.1 Correlated signals

To calculate the propagation of OP using Equation (25), the independence between inputs must be assumed. But in real circuits, the inputs of a gate are most likely to be correlated rather than independent. There are two kinds of correlations: temporal and spatial correlation. Temporal correlation indicates the current state of a signal is correlated with its previous state. Spatial correlation means two or more input signals to a gate are correlated to each other due to reconvergent fan-out (RFO) or pattern dependencies [62]. How to model these correlations is the key to propagate OP and SW through circuits.

5.3.2 The elimination of temporal correlation

In single-rail encoding synchronous circuits, there is no direct relationship between signal probability and switching activity. Although signal probability can be modeled accurately in most of the cases, switching activity is still hard to calculate. But in speed-independent circuit, from (24) it is clear that in dual-rail logic, SW can be calculated using OP ONLY (even the number of input data is unknown, (24) gives the switching probability). This is very important. For memoryless circuit, the existence of temporal correlation makes the signal behave as a Markov Chain [62]. From Markov Chain theory, the steady state probabilities can always be calculated using Equation (28),

$$\pi = \pi \cdot P$$
$$\sum_i \pi_i = 1 \tag{28}$$

where π is the steady state probability vector, P is the transition probability matrix [63]. So in dual-rail logic, the SW of signal under temporal correlation can be exactly modeled so that the effect of temporal correlation is eliminated. Figure 82 shows an example of this elimination: this is a 3-bit counter; during counting operation, the signals are strongly temporal correlated to themselves. For non-Data-Null operation, the signal probabilities of all three signals are all 0.5, but the switching activities are quite different. For Data-Null operation, same OP leads to same SW.

	Non-Data-Null			Data-Null		
	0	0	0	0	0	0
				0	0	0
	0	0	1	0	0	1
				0	0	0
	0	1	0	0	1	0
				0	0	0
	0	1	1	0	1	1
				0	0	0
	1	0	0	1	0	0
				0	0	0
	1	0	1	1	0	1
				0	0	0
	1	1	0	1	1	0
				0	0	0
	1	1	1	1	1	1
				0	0	0
	0	0	0	0	0	0
Number of switching	2	4	8	8	8	8

Figure 82. Example of the elimination of temporal correlation

5.3.3 A simplified method to model spatial correlation

During Data-Null cycles in speed-independent circuits, the logic highs are important. Unlike single-rail encoding, in dual-rail logic, two logic highs cannot be adjacent to each other. So Figure 81 can be redrawn as Figure 83.

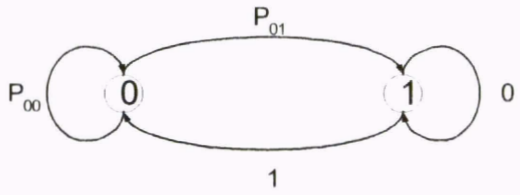


Figure 83. FSM in dual-rail logic

In Figure 83, there is only one variable, P_{01}, because P_{00} can be calculate by P_{01} subtracted by one. But in Figure 81, there are two variables, P_{01} and P_{10}. So dual-rail encoding weakens the uncertainty in calculating OP. This effect also reduces the error in approximating spatial correlations.

Again use NCL as an example of speed-independent circuits. NCL circuits can be divided to "stages". There is only one "gate depth" between two adjacent stages. Figure 84 shows some examples.

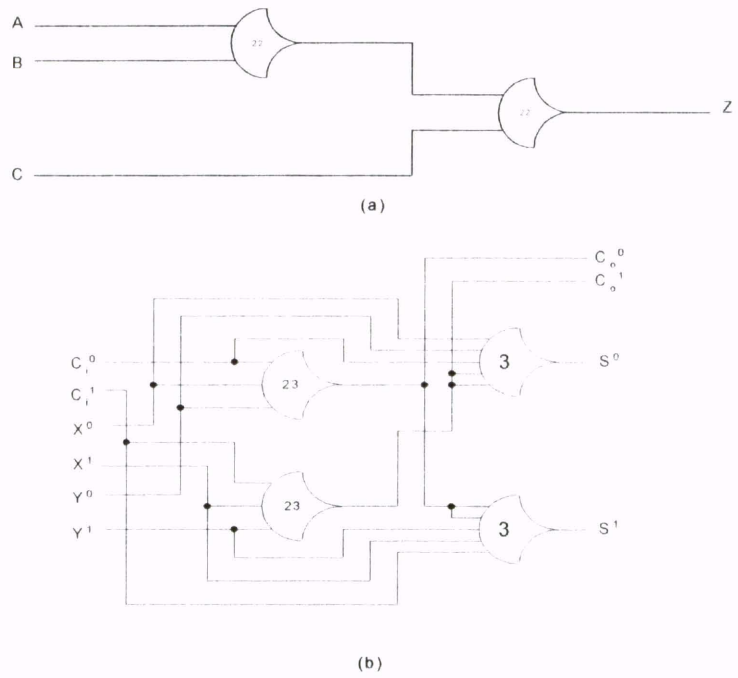


Figure 84. Examples of stage dividing

In Figure 84, circuit (a) can be divided to two stages. But circuit (b), which is a full adder, contains only one stage because the gates in right side share some inputs with the gates in left side. A group of simulations are done to simulate circuits in which the spatial correlated signals are connected to different stages. The results show that the larger the stage difference between the correlated signals, the less the error of output OP can be achieved by treating these signals as independent. When the stage difference is above three, under most combinations of the input OPs, the average error reduces to less than five percent. So a simple method to model spatial correlation in dual-rail circuit is proposed here. It is called Three-stage rule.

Three-stage rule: To calculate OP of the output of a stage, rather than tracing backward again and again to find independent source signals, find a group of signals that

are at least three stages away from any input to this stage, then use their OPs as if they are independent of each other.

This rule assumes that after three stages, the effect of original signal correlations can be neglected. Although this is an approximation, good results are shown in experiments. The importance of this rule is that it makes it possible to propagate OP throughout the whole circuit without huge calculation complexity.

5.4 Working flow of occurrence probability based switching activity analysis method

The working flow of this occurrence probability based switching activity analysis method is shown in Figure 85.

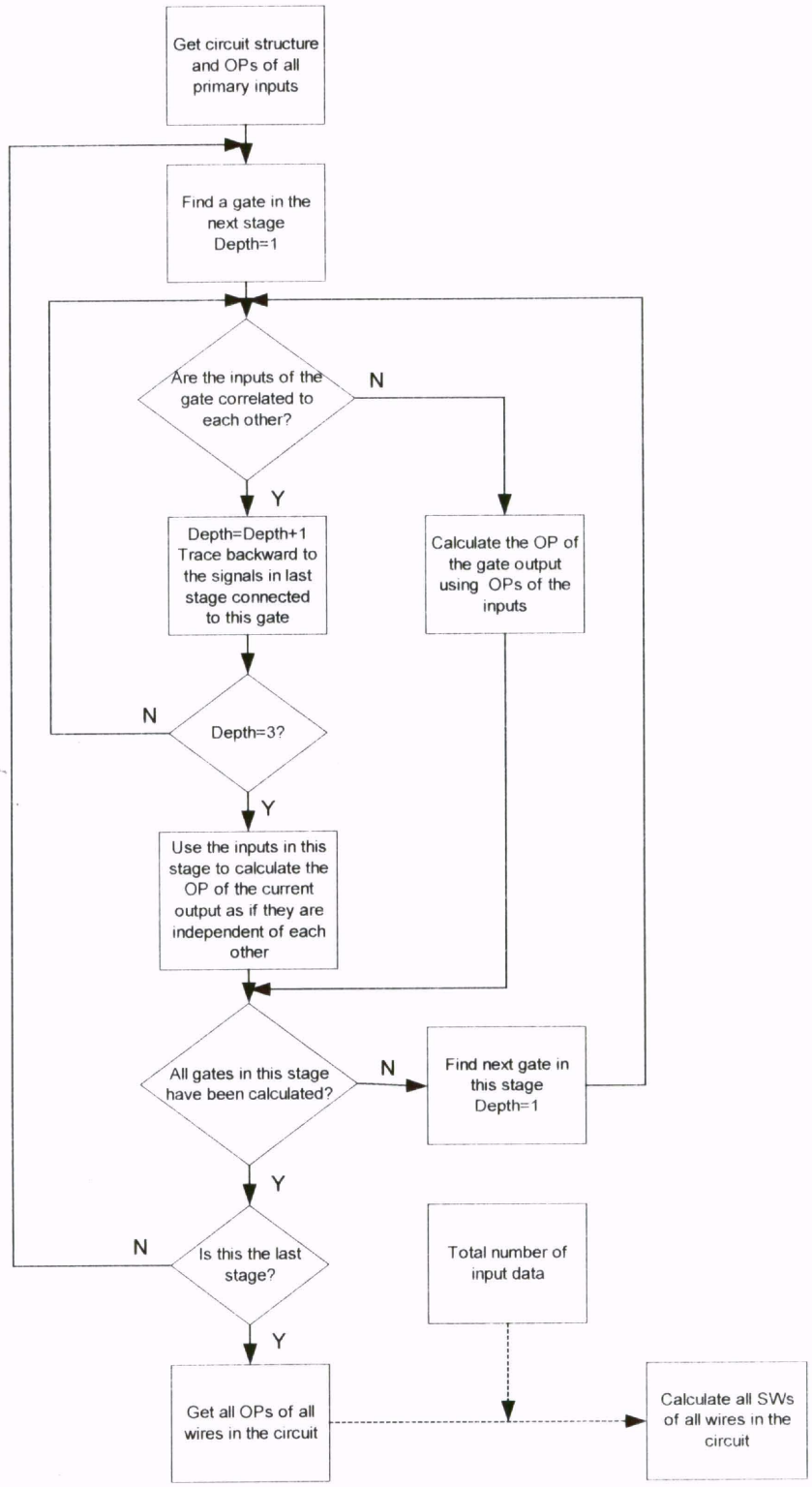


Figure 85. Working flow of switching analysis method

5.5 Case study — NCL 4x4 multiplier

This switching analysis method can be used to calculate SWs of any wire in a multi-rail speed-independent circuit. The example here is a NCL 4×4 multiplier as shown in Figure 86 [64].

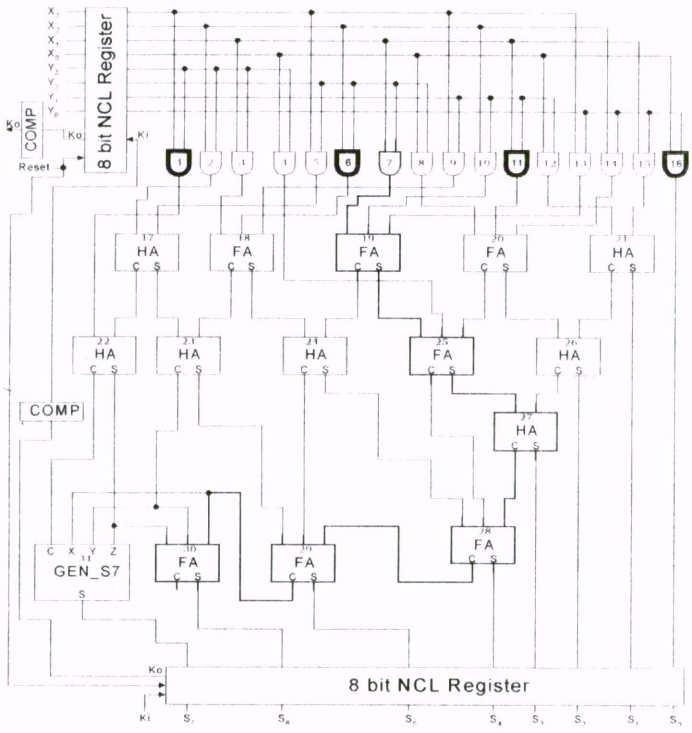


Figure 86. NCL 4×4 multiplier

In Figure 86, all lines shown are signals. Besides data paths, there are some control signals. RESET switches only once at the system start-up. KI and KO are handshaking signals. The number of logic highs of KI and KO is equal to the total number of input data. The NCL registers are used for pipelining. The OP and SW of data signals at the input are the same as that of the corresponding signals at the output.

This multiplier can be divided into eight stages. The first stage contains block 1 to 16. The second contains block 17 to 21. The third contains block 22 to 26. The fourth contains block 27. The fifth contains block 28. The sixth contains block 29. The seventh stage contains block 30 and the eighth stage contains block 31. Given the OPs of the eight primary inputs, starting from the first stage, the OPs of all signals are calculated one by one. Due to the three-stage rule, while calculating the OPs of the outputs of stage 4 to 8, the OPs of the outputs of stage 1 are used instead of that of the primary inputs. Then all SWs are calculated from OPs.

These SWs need to be compared with real switching data. Synopsys software is used to simulate the synthesized multiplier. Firstly the OPs of the primary inputs are given. Then a random input pattern is generated based on those OP values. This pattern is provided to the multiplier to simulate. During the simulation, the switching behaviors of all signals are recorded. When this simulation is over, another random input pattern is generated. After many simulation cycles (over 8000), the average SWs are calculated. Because real gate delays are used in the simulation, the results are accurate. They can be used as exact SWs to measure the correctness of the theoretical results.

Three experiments of different input OPs are done. The original OPs of primary inputs of these experiments are given in Table 11. Among these experiments, the experiment 3 is the worst case because when OPs are close to 0 and 1, there are always some wires that rarely switch. The error percentage can easily become high based on small number of switching. For the same reason, the experiment 1 is the best case. The error percentages between real and theoretical SWs of all signals in Figure 86 are shown in the Figure 87, 88, and 89. From these figures it is clear that this switching activity

analysis method leads to accurate results. Most of the error percentages are below 5%. Note that in Experiment 3, the number of switching of B27c is less than 10. That is very small compared to the total over 8000 input data. So although the error percentage looks large, the absolute error is so small that it will not affect the accuracy of total switching analysis.

Table 11. *OP*s of primary inputs in experiments

OP	X3	X2	X1	X0	Y3	Y2	Y1	Y0
Experiment 1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Experiment 2	0.7	0.6	0.5	0.4	0.4	0.5	0.6	0.7
Experiment 3	0.8	0.2	0.8	0.2	0.2	0.8	0.2	0.8

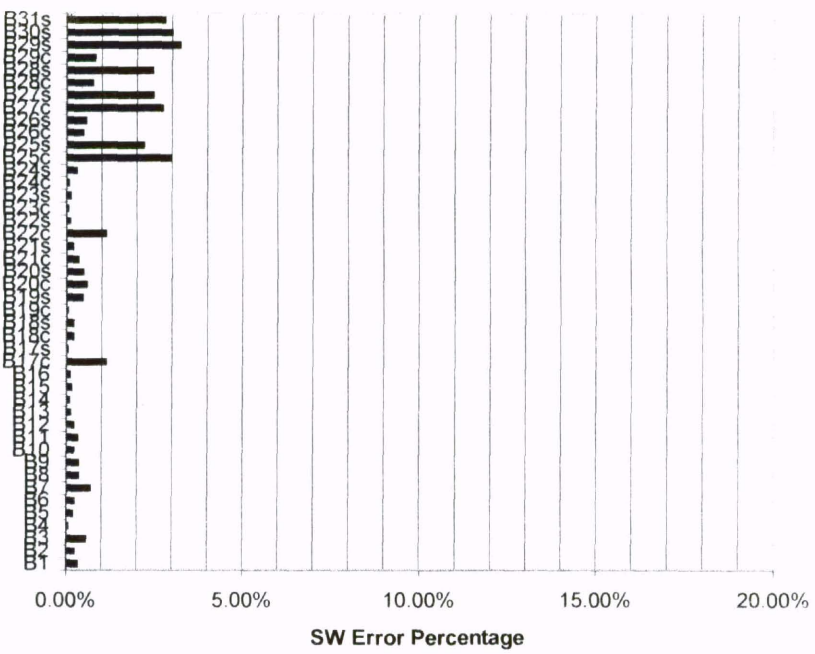


Figure 87. SW error percentage of experiment 1

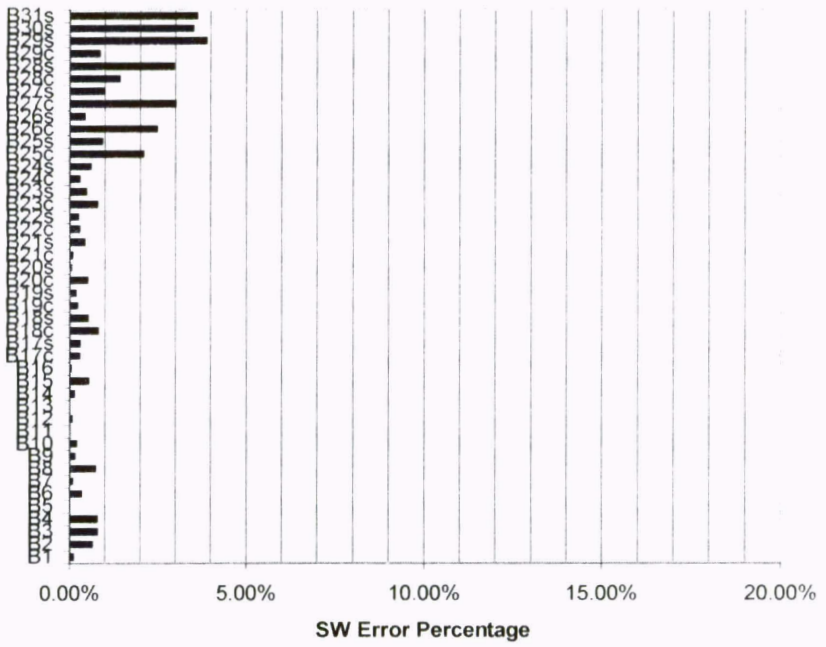


Figure 88. SW error percentage of experiment 2

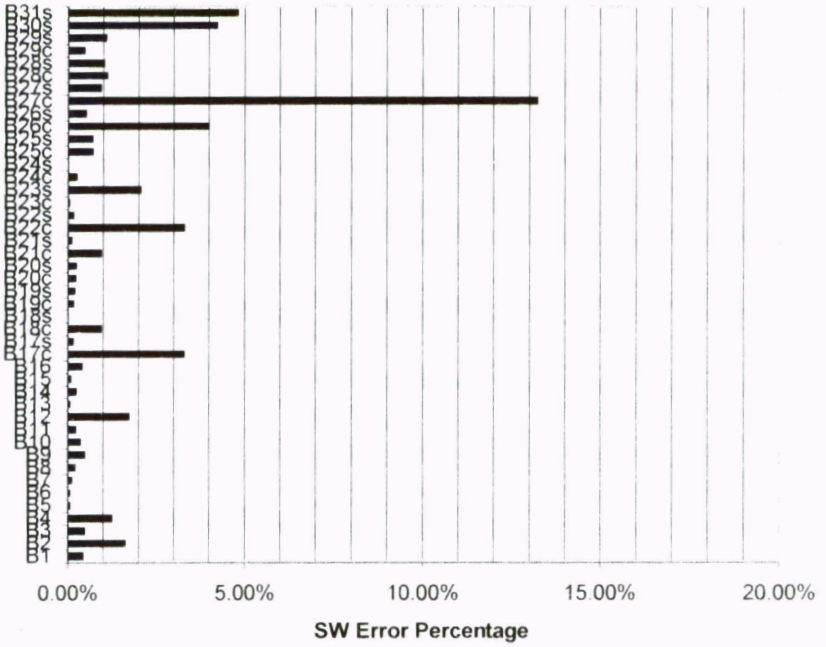


Figure 89. SW error percentage of experiment 3

CHAPTER SIX: CONCLUSION

6.1 Summary

In this dissertation, some power/energy reduction techniques regarding to power/energy awareness have been discussed. Two energy estimation methods are also presented. The achievements are summarized as following:

1) A power optimization technique, 2-Dimensional Pipeline Gating, has been developed for synchronous pipeline circuits. By gating clock signals to registers in both vertical direction (along data flow direction) and horizontal direction (within each pipeline stage), the switching activities of clock signals connected to unused registers can be significantly reduced with the changes of current input precisions, thus improves the power awareness.
2) For unpipelined asynchronous circuits, a technique named Signal Bypassing & Insertion has been proposed for NCL. When the current input precision changes, Nulls are used to replace the redundant Data 0's during the calculation. The correct result is selected by multiplexers. The energy awareness is improved due to the reduction of switching activities.
3) For pipelined asynchronous circuits, another technique named Zero Insertion has been developed for NCL. The basic idea of Zero Insertion is the same as Signal Bypassing & Insertion. But the presence of pipeline brings several new

difficulties like control signal propagation. On the other hand, it is relatively easier to modify the structure of NCL registers and to insert Data 0. So the overhead of Zero Insertion is much smaller.

4) Both Signal Bypassing & Insertion and Zero Insertion need a central control unit to detect the changing of input precision and to set up the global control signals. There are two potential problems. Firstly, this unit is not always available. Secondly, when the input precision changes frequently, both schemes are not efficient enough. A dynamic active-bit detection scheme is developed to solve this problem. A pre-processing unit is added to detect the current input precision dynamically. To calculate the overhead beforehand, an energy macromodel for array multipliers has been developed.

5) A transistor-level energy estimation technique, Input Mapping, has been developed for Equivalent Inverter modeling method. This technique is to use an inverter to replace the original CMOS gate, and to map the multiple inputs into one input connected to the equivalent inverter. Then the effort of energy calculation can be significantly reduced.

6) For multi-rail encoded asynchronous circuit, a switching activity modeling method has been proposed for dual-rail NCL. Based on Occurrence Probability, the number of switching on each nodes inside the circuits can be accurately calculated.

6.2 Future Work

To extend the research from the previous works, building a real chip is needed. All my previous works are in gate-level and below. But the real applications are in RTL

and system-level. Designing and fabricating a real chip will be very helpful not only in testing the existing techniques, but also in triggering new ideas in a different angle.

APPENDIX

An Analytical Energy Macromodel For Dynamic Active-bit Detection Scheme To Design Energy-aware NCL Multiplier

For completion detection circuitry, an assumption is needed.

Assumption 1: Since it is difficult to calculate the exact number of transistors increased when the $2n$-input completion detection circuit (C_{2n}) is upgraded to ($2n$-1)-input detection circuit (C_{2n-1}), an average value, denoted as C^*, is used instead. This value is calculated by averaging the number of transistors increased from C_1 to C_{16}.

All notations used in the equations are summarized in Table 12.

Table 12 Notations used in the equations

Notation	Meaning
DB	Disabled adder Block
DR	Disabled Register
A	Group of Disabled Register due to the changing of input A
B	Group of Disabled Register due to the changing of input B
AnBn	Group of Disabled Register due to the changing of input bit A_n and B_n
S	Group of Disabled Register due to the changing of output *S*
IL	Low part of group *I* of Disabled Register
IH	High part of group *I* of Disabled Register
IIL	Low part of group *II* of Disabled Register
IIH	High part of group *II* of Disabled Register
III	Group *III* of Disabled Register
IR	Inserted Register
CD	Control Depth. The number of stages the control signal needs to go through
CA	Completion Adjustment Circuitry
CI	Completion Increment
CS	Completion Subtraction
CU	Completion Upgrade
NC	New Completion circuitry
TC	Total Completion circuitry of the original multiplier
n	Computational length of the multiplier. *n* must be greater than 3.
r	Input precision of input A
m	Input precision of input B
C#	Transistor increment for the completion circuitry of inserted registers and edged adders
Cx	Transistor count of *x*-input completion detection circuitry

Based on the structure shown in Fig. 68, the total calculation is divided into 7 regions. Equations for each region are shown in the Table 13 as below.

Table 13 Model equations

Region 1. $r = n, m = k, n \geq k$			
$k = n-1$	DB		0
	DR		$n-1$
	IR		1
	CD		$n-2$
	CA		0
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$(n-2)\times C_1$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$k = n-2$	DB	$Type\ 1$	0
		$Type\ 2$	0
		$Type\ 3$	0
		$Type\ 4$	1
		$Type\ 5$	0
		$Type\ 6$	0
	DR	A	0
		B	$(n-1)+(n-2)$
		$AnBn$	$2\times(n-1)$
		S	2
		IL	0
		IH	0
		IIL	$n-1$
		IIH	0
		III	1
	IR		2
	CD		$2n-3$
	CA		3
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$(B-2)\times C_1 + A_nB_n \times C_1 + 2\times C_2 + (n-2)\times C_1$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$k = n-3$	DB	$Type\ 1$	1
		$Type\ 2$	0
		$Type\ 3$	1

		Type 4	*1*
		Type 5	*0*
		Type 6	*0*
	DR	*A*	*1*
		B	$(n-1)+(n-1)+(n-3)$
		AnBn	$2\times(n-1)$
		S	*3*
		IL	$n-2$
		IH	*0*
		IIL	$(n-1)+(n-2)$
		IIH	*1*
		III	*2*
	IR		*3*
	CD		$2n-4$
	CA		*4*
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-3)\times C_1+A_nB_n\times C_1+(I_L-1)\times C_1+C_2+(II_L-2)\times C_1+2\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$2\le k\le n-4$	*DB*	*Type 1*	$n-k-2$
		Type 2	$\frac{[1+(n-k-3)](n-k-3)}{2}$
		Type 3	$n-k-2$
		Type 4	*1*
		Type 5	*0*
		Type 6	*0*
	DR	*A*	$\frac{[(n-k-1)+2](n-k-2)}{2}$
		B	$(n-1)+(n-1)+k+\frac{(n+k)(n-k-3)}{2}$
		AnBn	$2\times(n-1)$
		S	$n-k$
		IL	$\frac{[(k+1)+(n-2)](n-k-2)}{2}$

		IH	$\frac{[1+(n-k-3)](n-k-3)}{2}$
		IIL	$\frac{[(k+1)+(n-1)](n-k-1)}{2}$
		IIH	$\frac{[1+(n-k-2)](n-k-2)}{2}$
		III	$n-k-1$
	IR		$n-k$
	CD		$n+k-1$
	CA		$2n-2k-1$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-3-(n-k-3))\times C_1+A_nB_n\times C_1+$ $(I_L-(n-k-2))\times C_1+(n-k-2)\times C_2+I_H\times C_2+$ $(II_L-(n-k-1))\times C_1+(n-k-1)\times C_2+$ $II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$k=1$	*DB*	*Type 1*	6
		Type 2	$\frac{[1+(n-3)](n-3)}{2}$
		Type 3	$n-3$
		Type 4	1
		Type 5	0
		Type 6	1
	DR	*A*	$\frac{[(n-1)+2](n-2)}{2}$
		B	$(n-1)+\frac{(n-2)(n+1)}{2}$
		AnBn	$2\times(n-1)$
		S	n
		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-3)](n-3)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$\frac{[1+(n-2)](n-2)}{2}$
		III	$n-2$
	IR		$n-1$

	CD		$n-2$
	CA		$2n-2$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-2))\times C_1+A_nB_n\times C_1+$ $(I_L-(n-2))\times C_1+(n-2)\times C_2+I_H\times C_2+$ $(II_L-(n-1))\times C_1+(n-1)\times C_2+II_H\times C_2+$ $III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
Region 2. $r=n-1, m=k, n\geq k+1$			
$k=n-1$	DB	*Type 1*	*0*
		Type 2	*0*
		Type 3	*0*
		Type 4	*1*
		Type 5	*0*
		Type 6	*0*
	DR	*A*	*0*
		B	$(n-1)$
		AnBn	$2\times(n-1)$
		S	*2*
		IL	*0*
		IH	*0*
		IIL	$n-1$
		IIH	*0*
		III	*1*
	IR		*2*
	CD		$2n-3$
	CA		*3*
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$(B-1)\times C_1+A_nB_n\times C_1+(II_L-1)\times C_1+C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$k=n-2$	DB	*Type 1*	*1*
		Type 2	*0*
		Type 3	*1*
		Type 4	*1*
		Type 5	*0*
		Type 6	*0*
	DR	*A*	*2*
		B	$(n-1)+(n-1)$
		AnBn	$2\times(n-1)$

		S	3
		IL	$n-2$
		IH	0
		IIL	$2n-3$
		IIH	1
		III	2
	IR		3
	CD		$2n-4$
	CA		5
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-2)\times C_1+A_nB_n\times C_1+(I_L-1)\times C_1+$ $C_2+(II_L-2)\times C_1+2\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$2\le k\le n-4$		$Type\ 1$	$n-k-1$
		$Type\ 2$	$\dfrac{[1+(n-k-2)](n-k-2)}{2}$
	DB	$Type\ 3$	$n-k-1$
		$Type\ 4$	1
		$Type\ 5$	0
		$Type\ 6$	0
		A	$\dfrac{[(n-k)+2](n-k-1)}{2}$
		B	$(n-1)+\dfrac{(n+k)(n-k-1)}{2}$
		$AnBn$	$2\times(n-1)$
		S	$n-k-1$
		IL	$\dfrac{[(k+1)+(n-1)](n-k-1)}{2}$
	DR	IH	$\dfrac{[1+(n-k-2)](n-k-2)}{2}$
		IIL	$\dfrac{[k+(n-1)](n-k)}{2}$
		IIH	$\dfrac{[1+(n-k-1)](n-k-1)}{2}$
		III	$n-k$
	IR		$n-k+1$
	CD		$n+k-2$
	CA		$2n-2k+1$

	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-k-1))\times C_1+A_nB_n\times C_1+$ $(I_L-(n-k-1))\times C_1+(n-k-1)\times C_2+I_H\times C_2+$ $(II_L-(n-k))\times C_1+(n-k)\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$k=1$	DB	$Type\ 1$	$n-2$
		$Type\ 2$	$\frac{[1+(n-2)](n-2)}{2}$
		$Type\ 3$	$n-3$
		$Type\ 4$	1
		$Type\ 5$	1
		$Type\ 6$	1
	DR	A	$n-1+\frac{n\times(n-1)}{2}-1$
		B	$(n-1)+\frac{(n-2)(n+1)}{2}$
		$AnBn$	$2\times(n-1)$
		S	$n+1$
		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-2)](n-2)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$\frac{[1+(n-2)](n-2)}{2}+n-2$
		III	$n-2$
	IR		$n-2$
	CD		$n-2$
	CA		$2n-3$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-2))\times C_1+A_nB_n\times C_1+$ $(I_L-(n-2))\times C_1+(n-2)\times C_2+I_H\times C_2+$ $(II_L-(n-1))\times C_1+(n-1)\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
$Region\ 3.\ r=x, m=y, n-1\geq x\geq y\geq 2, x+y\geq n+1$			
	DB	$Type\ 1$	$2n-x-y-2$

		Type 2	$\frac{[1+(2n-x-y-3)](2n-x-y-3)}{2}$
		Type 3	$2n-x-y-2$
		Type 4	*1*
		Type 5	*0*
		Type 6	*0*
	DR	*A*	$\frac{[(2n-x-y-1)+2](2n-x-y-2)}{2}$
		B	$(n-1)+\frac{(n+y)(n-y-1)}{2}+(2n-x-y-2)-(n-y)+1$
		AnBn	$2\times(n-1)$
		S	$2n-x-y$
		IL	$\frac{[(x+y-n+1)+(n-2)](2n-x-y-2)}{2}$
		IH	$\frac{[1+(2n-x-y-3)](2n-x-y-3)}{2}$
		IIL	$\frac{[x+y-n+1+(n-1)](2n-x-y-1)}{2}$
		IIH	$\frac{[1+(2n-x-y-2)](2n-x-y-2)}{2}$
		III	$2n-2-x-y+1$
	IR		$2n-x-y$
	CD		$x+y-1$
	CA		$2n-x-y+(n-1-x-y+n)+(2n-x-y-2)-n+y+1$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-y-1))\times C_1+A_nB_n\times C_1+$ $(I_L-(2n-x-y-2))\times C_1+(2n-x-y-2)\times C_2+$ $I_H\times C_2+(II_L-(2n-x-y-1))\times C_1+$ $(2n-x-y-1)\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
Region 4. $r=x, m=y, n-1\geq x\geq y\geq 2, x+y=n$			
	DB	*Type 1*	$n-2$
		Type 2	$\frac{[1+(n-3)](n-3)}{2}$
		Type 3	$n-3$
		Type 4	*1*

		Type 5	*0*
		Type 6	*1*
	DR	*A*	$\frac{[(n-1)+2](n-2)}{2}$
		B	$(n-1)+\frac{(n+y)(n-y-1)}{2}+(2n-x-y-2)-(n-y)+1$
		AnBn	$2\times(n-1)$
		S	n
		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-3)](n-3)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$\frac{[1+(n-2)](n-2)}{2}$
		III	$n-2$
	IR		n
	CD		$n-1$
	CA		$2n-4+(2n-x-y-2)-(n-y)+1$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-y-1))\times C_1+A_nB_n\times C_1+$ $(I_L-(n-2))\times C_1+(n-2)\times C_2+I_H\times C_2+$ $(II_L-(n-1))\times C_1+(n-1)\times C_2+II_H\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
Region 5. $r=x, m=y, n-1\geq x\geq y\geq 2, x+y\leq n-1$			
	DB	*Type 1*	$n-2$
		Type 2	$\frac{[1+(n-3)](n-3)}{2}+\frac{(n+x+y-3)(n-x-y)}{2}$
		Type 3	$n-3$
		Type 4	*1*
		Type 5	$n-x-y$
		Type 6	*1*

	DR	A	$n\times(n-x-y)+\frac{[(n-1)+(n-x-y)](x+y)}{2}$
		B	$(n-1)+\frac{(n+y)(n-y-1)}{2}+y-1$
		AnBn	$2\times(n-1)$
		S	$2n-x-y$
		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-2)](n-2)}{2}+\frac{(x+y+n-4)(n-x-y-1)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$\frac{[1+(n-2)](n-2)}{2}+\frac{(x+y+n-3)(n-x-y)}{2}$
		III	$n-2$
	IR		$x+y$
	CD		$x+y-1$
	CA		$2x+3y-3$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-y-1))\times C_1+(n-x-y)\times C_{2n-1}+$ $A_nB_n\times C_1+(I_L-(n-2))\times C_1+(n-2)\times C_2+$ $I_H\times C_2+(II_L-(n-1))\times C_1+(n-1)\times C_2+II_H\times C_2+$ $III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
Region 6. $r=x, m=y, n-1\geq x\geq y, y=1$			
	DB	Type 1	$n-2$
		Type 2	$\frac{[1+(n-3)](n-3)}{2}+\frac{(n+x-3)(n-x-2)}{2}$
		Type 3	$n-3$
		Type 4	1
		Type 5	$n-x$
		Type 6	1
	DR	A	$n\times(n-x)+\frac{[(n-1)+(n-x+1)](x-1)}{2}-1$
		B	$(n-1)+\frac{(n+1)(n-2)}{2}$
		AnBn	$2\times(n-1)$
		S	$2n-x$

		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-2)](n-2)}{2}+\frac{(x+n-4)(n-x-1)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$\frac{[1+(n-2)](n-2)}{2}+\frac{(x+n-3)(n-x)}{2}$
		III	$n-2$
	IR		$x-1$
	CD		$x-1$
	CA		$2x$
	CI		$(IR+CA-1)\times C^{\#}$
	CS		$A\times C_3+(B-1-(n-2))\times C_1+(n-x)\times C_{2n-1}+$ $A_nB_n\times C_1+(I_L-(n-2))\times C_1+(n-2)\times C_2+I_{II}\times C_2+$ $(II_L-(n-1))\times C_1+(n-1)\times C_2+II_{II}\times C_2+III\times C_2$
	CU		$(TC-CS)\times C^{*}$
	NC		$CU+CI$
Region 7. $r=x, m=y, x=y=1$			
	DB	*Type 1*	$n-2$
		Type 2	$(n-2)^2$
		Type 3	$n-3$
		Type 4	*1*
		Type 5	$n-1$
		Type 6	*1*
	DR	*A*	$n\times(n-2)+(n-1)$
		B	$(n-1)+\frac{(n+2)(n-2)}{2}$
		AnBn	$2\times(n-1)$
		S	$2n-1$
		IL	$\frac{[1+(n-2)](n-2)}{2}$
		IH	$\frac{[1+(n-2)](n-2)}{2}+\frac{(1+(n-3))(n-3)}{2}$
		IIL	$\frac{[1+(n-1)](n-1)}{2}$
		IIH	$(1+(n-2))(n-2)$
		III	$n-2$
	IR		*0*
	CD		*1*

	CA	2
	CI	$(IR+CA-1)\times C^{\#}$
	CS	$A\times C_3+(B-1-(n-y-1))\times C_1+(n-1)\times C_{2n-1}+$ $A_nB_n\times C_1+(I_L-(n-2))\times C_1+(n-2)\times C_2+I_{II}\times C_2+$ $(II_I-(n-1))\times C_1+(n-1)\times C_2+II_{II}\times C_2+III\times C_2$
	CU	$(TC-CS)\times C^{*}$
	NC	$CU+CI$

LIST OF REFERENCES

[1] F. N. Najm, "Feedback, correlation, and delay concerns in the power estimation of VLSI circuits," *ACM/IEEE Design Automation Conference*, pp.612-617, 1995.

[2] Neil H. E. Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design, A systems perspective" second edition, Addison-Wesley Publishing Company, 1993

[3] Bernard Cole, "Asynchronous design gets a second look", *EE Times*, June 9th, 2003

[4] E. Macii, M. Pedram and F. Somenzi, "High-level Power Modeling, Estimation, and Optimization," *IEEE Trans. on Computer Aided Design*, Volume: 17, no. 11, Nov. 1998, Page(s): 1061-1079.

[5] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI System*, Volume 2, no. 4, pp. 446-455, 1994.

[6] T. Sato, Y. Ootaguro, M. Nagamatsu, and H. Tago, "Evaluation of archyitectural-level power estimation for CMOS RISC processors," in *proceedings of ISLPE-95, IEEE Int. Symp. Low Power Electronics*, San Jose, CA, Oct. 1995, pp. 44-45.

[7] C. –L. Su, C. –Y. Tsui, and A. M. Despain, "Power analysis of embedded software: A first step toward software power minimization" *IEEE Trans. VLSI Syst.*, vol. 2, no. 4, pp. 437–445, 1994.

[8] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step toward software power minimization," *IEEE Trans. VLSI Syst.*, vol. 2, no. 4, pp. 437–445, 1994

[9] C.-T. Hsieh, M. Pedram, H. Mehta, and F. Rastgar, "Profile-driven program synthesis for evaluation of system power dissipation," in *Proc. DAC-34: ACM/IEEE Design Automation Conf.*, Anaheim, CA, June 1997, pp. 576–581.

[10] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for power analysis," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 6, pp. 599–610, 1996.

[11] M. Nemani and F. Najm, "Toward a high-level power estimation capability," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 6, pp. 588–598, 1996.

[12] K. T. Cheng and V. D. Agrawal, "An entropy measure for the complexity of multi-output Boolean functions," in *Proc. DAC-27: ACM/IEEE Design Automation Conf.*, Orlando, FL, June 1990, pp. 302–305.

[13] F. Ferrandi, F. Fummi, E. Macii, M. Poncino, and D. Sciuto, "Power estimation of behavioral descriptions," in *Proc. DATE-98: IEEE Design Automation and Test in Europe*, Paris, France, Feb. 1998, pp. 762–766.

[14] A. Tyagi, "Entropic bounds on FSM switching," *IEEE Trans. VLSI Syst.*, vol. 5, no. 4, pp. 456–464, 1997.

[15] K. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating essential design characteristics to support project planning for ASIC design management," in *Proc. ICCAD-91: IEEE/ACM Int. Conf. Computer Aided Design*, Santa Clara, CA, Nov. 1991, pp. 148–151.

[16] M. Nemani and F. Najm, "High-level area prediction for power estimation," in *Proc. CICC-97: Custom Integrated Circuits Conf.*, Santa Clara, CA, May 1997, pp. 483-486.

[17] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 1, pp. 12–31, 1995.

[18] J. M. Chang and M. Pedram, "Module assignment for low power," in *Proc. EuroDAC-96: IEEE Eur. Design Automation Conf.*, Geneva, Switzerland, Sept. 1996, pp. 376–381.

[19] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low power VLSI systems," *IEEE Design Test Comput. Mag.*, vol. 12, no. 3, pp. 70–84, 1995.

[20] R. San Martin and J. Knight, "Optimizing power in ASIC behavioral synthesis," *IEEE Design Test Comput. Mag.*, vol. 13, no. 2, pp. 58–70, 1996.

[21] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, "Regression models for behavioral power estimation," in *Proc. PATMOS-96: Int. Workshop on Power and Timing Modeling, Optimization and Simulation*, Bologna, Italy, Sept. 1996, pp. 179–186.

[22] L. Benini, A. Bogliolo, and G. De Micheli, "Characterization-free behavioral power modeling," in *Proc. DATE-98: IEEE Design Automation and Test in Europe*, Paris, France, Feb. 1998, pp. 767–773.

[23] A. Boglioio, L. Benini, G. De Micheli, "Adaptive least mean square behavioral power modeling," in *Proc. EDTC-97: IEEE Eur. Design and Test Conf.*, Paris, France, Mar. 1997, pp. 404–410.

[24] C. M. Huizer, "Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation," in *Proc. IEEE Eur. Solid State Circuits Conf.*, 1990, pp. 61–64.

[25] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of powermill," in *Proc. ISLPD-95: ACM/IEEE Int. Symp. Low Power Design*, Dana Point, CA, Apr. 1995, pp. 105–110.

[26] F. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 4, pp. 439–450, 1990.

[27] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power dissipation under a real delay model," in *Proc. ICCAD-93: IEEE/ACM Int. Conf. Computer Aided Design*, Santa Clara, CA, Nov. 1993, pp. 224–228.

[28] M. Borah, R.M. Owens, M.J. Irwin, "Transistor sizing for low power CMOS circuits ", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume: 15 Issue: 6, Jun 1996, Page(s): 665 -671

[29] 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, pp. 468-483, Aug. 1984

[30] U. Ko and P.T. Balsara, "Short-circuit power driven gate sizing technique for reducing power dissipation", *IEEE Trans. VLSI Syst.*, vol. 3, No. 3, Sep. 1995

[31] R. Hossain, M. Zheng, and A. Albicki, "Reducing power dissipation in CMOS circuits by signal probability based transistor reordering", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Volume 15, NO. 3, March 1996

[32] W. Shen, J. Lin, and F. Wang, "Transistor Reordering Rules for Power Reduction in CMOS Gates", 1995. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95, IFIP International Conference on Hardware Description Languages; IFIP International Conference on Very Large Scale Integration, Asian and South Pacific Design Automation Conference, 29 Aug-1 Sep 1995, Page(s): 1 -6

[33] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold boltrage scaling for low power CMOS", *IEEE Journal of Solid-State Circuits*, vol. 32, No. 8, Aug. 1997, pp. 1210-1216

[34] R. Bai, S. Kulkami, W. Kwong, A. Srivastava, D. Sylvester, and D. Blaauw, "An implementation of a 32-bit ARM processor using dual power supplies and dual threshold voltages", *IEEE Computer Society Annual Symposium on VLSI*, Feb. 2003

[35] K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. chiba, Y. Watanabe, K. Matsuda, T. Maeda, and T. Kuroda, "A 300MIPS/W RISC core processor with variable supply-voltage scheme in variable threshold-voltage CMOS", *IEEE Custom Integrated Circuits Conference*, 1997

[36] S. Jung, K. Kim, and S. Kang, "Low-swing clock domino logic incorporating dual supply and dual threshold voltages", *39th Design Automation Conference*, 2002, Page(s): 467 -472

[37] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits", *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 47, NO. 103, March 2000, pp. 415-420

[38] M. Pedram and J. Rabaey, "Power Aware Design Methodologies", *Kluwer Academic Publishers*, 2002

[39] Manish Bhardwaj, et al. "Quantifying and Enhancing Power Awareness of VLSI Systems". *IEEE Transactions on VLSI Systems*. 2001, Volume 9, Issue 6, pages 757-772.

[40] S. H. Nawab, et al. "Approximate signal processing". *J. VLSI Signal Processing Syst. Signal, Image, and Video Technol,* Volume 15, no. 1/2, Jan. 1997, pages 177-200

[41] M. Dean, T. Williams and D. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)", *MIT Conference on Advanced Research in VLSI*, March 1991, pp. 55-70

[42] I. David, R. Ginosaur and M. Yoeli, "Implementing sequential machines as self-timed circuits", *IEEE Transactions on Computers*, Vol. 41, No. 1, January 1992, pp. 12-17

[43] I. E. Sutherland, "Mircopipelines", *Communications ACM*, June 1989, 32, (6), pp. 55-70

[44] Fant, K. and Scott A. Bradt. "Null Convention Logic™: a complete and consistent logic for asynchronous digital circuits synthesis". *Proceedings of International Conference on Application Specific Systems*, 1996, pages 261-273

[45] Peter A. Beerel, Kenneth Y. Yun, Steven M. Nowick, Pei-Chuan. Yeh. "Estimation and Bounding of Energy Consumption in Burst-Mode Control circuits". *IEEE/ACM International Conference on Computer-Aided Design*. 1995, pages 26-33.

[46] S. Kim, M. C. Papaefthymiou, Reconfigurable low energy multiplier for multimedia system design, Proceedings of IEEE Computer Society Workshop on VLSI, 2000

[47] B. Parhami, Computer arithmetic – algorithms and hardware designs, Oxford University Press, 1999

[48] K. K. Parhi, VLSI Digital Signal Processing Systems, John Willey & Sons Inc., 1999

[49] Peter A. Beerel, Kenneth Y. Yun, Steven M. Nowick, Pei-Chuan. Yeh. "Estimation and Bounding of Energy Consumption in Burst-Mode Control circuits". *IEEE/ACM International Conference on Computer-Aided Design*. 1995, pages 26-33.

[50] S. C. Smith. "Gate and Throughput Optimizations for Null Convention Self-timed Digital Circuits", *Ph.D. Thesis, University of Central Florida*, 2001

[51] Ted Williams, "Latency and Throughput Tradeoffs in Self-timed Speed-independent Pipelines and Rings", Technical report, Computer System Laboratory, Stanford University, Aug. 1990

[52] Chris J. Myers, "Asynchronous Circuit Design", John Wiley & Sons, Inc. 2001

[53] Jia Di, J. S. Yuan and M. Hagedorn, "Switching Activity Modeling of Multi-rail Speed-independent Circuits – A Probabilistic Approach". *IEEE 45th Midwest Symposium on Circuits and Systems*, Aug. 2002

[54] A. Chandrakasan, W. J. Bowhill, and F. Fox, "Design of High-performance Microprocessor Circuits", IEEE Press, 2001

[55] L. E. Lucke and K. K. Parhi. "Parallel processing architectures for rank order and stack filters", *IEEE Trans. On Signal Processing*, pp. 1178-1189, 1994

[56] K. K. Parhi. "VLSI Digital Signal Processing Systems", John Wiley & Sons, Inc. 1999

[57] F. Najm, "Towards a high-level power estimation capability," 1995 International Symposium on Low-Power Design, Apr. 1995, pp 87-92

[58] T. Sato, Y. Ootaguro, M. Nagamatsu, and H. Tago, "Evaluation of architecture-level power estimation for CMOS RISC processors," 1995 International Symposium on Low-Power Electronics, Oct. 1995, pp 44-45

[59] Alexander Chatzigeorgiou, Spiridon Nikolaidis and Ioannis Tsoukalas, "A modeling technique for CMOS gates," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 18, NO. 5, pp 557-575, May 1999

[60] Y. –H. Jun, K. Jun, and S. –B. Park, "An accurate and efficient delay time modeling for MOS logic circuits using polynomial approximation," *IEEE Trans. Computer-Aided Design*, vol. 8, pp 1027-1032, Sept. 1989

[61] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, pp 584-594, Apr. 1990

[62] R. Marculescu, D. Marculescu and Massoud Pedram. "Logic Level Power Estimation Considering Spatiotemporal Correlations". *USC CENG* 94-05

[63] Alberto Leon-garcia. "Probability and Random Processes for Electrical Engineering". *Addison-Wesley Pub Co*, 1993

[64] S. C. Smith, R. Demara, J.S. Yuan, M. Hagedorn and D. Ferguson. "Delay-insensitive gate-level pipelining". *VLSI Journal on Integration*, 2000

DATE DUE

APR 05 2005			
JUN 19 2005			
AUG 31 2007			
JAN 02 2008			
			Printed in USA