

# Chapter 1

## Introduction

Design techniques of low power devices, circuits, architectures, and algorithms in the complex system-on-chip (SoC) design are becoming important [1.1]. As technology move into deep submicron and nano-scaled feature size, leakage power will become comparable to dynamic powering in the future years [1.2]. Dynamic power is also increasing and is still the major source. The most effective way of low power is to reduce the supply voltage. The supply voltage which has been scaled aggressively is necessary to reduce the dynamic power and the leakage power in both the active and standby modes of operation.

In the increasing demand of portable devices such as cellular phone, laptops, and PDAs, battery-life is the major concern. Generally, the DSP processor is the key component of portable system, and thus consumes large amount of energy or power from battery. The Multiplier-Accumulator (MAC) unit is the major component of DSP processors, and it consumes nearly one half powers [2.10] of DSP core. Thus, the power minimization technique for datapath, the MAC unit, is our focus throughout this thesis.

In the area of fast VLSI arithmetic design, there is a significant disconnect between the algorithms and physical design. How do we estimate speed and efficiency of our algorithm? What criteria should we use when we develop a new algorithm? How does power enter into these criteria? We want to estimate as close as possible to the physical design results. Therefore, a simple tool which can evaluate speed and power of different design trade-off for a given technology is needed. Logic effort, a method for designing fast CMOS circuits, is a simple model can be used to estimate delay efficiently. Energy-delay sensitivity, a metric for trading power for speed, indicates that how much energy it would cost to increase speed. These two simple tools provide good estimation of speed and fine tradeoffs for low-power.

The overview of MACs will be presented in the beginning of chapter 2. The high-speed circuit design techniques and low-power methods will be discussed in the remainder of chapter 2. Logic effort design techniques will be discussed which including hand calculation estimation and calibrating method of the model for dedicated fabrication process. Then, the low power techniques focus on datapath structure will be introduced, it categorized which low power design technique is suitable for either design time or run time. The analytical formula of energy-delay sensitivity will be discussed for knowing the trend of power-speed curve on the power delay design space. In order to achieve

optimum point of power and delay, circuit level optimization based on this trend to adjust tuning variables to move the design point from normal to optimum.

High-speed multiplier-accumulator micro-architecture design will be implemented in chapter 3. Cell based IC design flow is taken as a platform to synthesize the efficient algorithm for comparison. Four recoding schemes, two partial product matrix topologies, as well as three high speed parallel adders will be described in Verilog-HDL gate level. These Verilog-HDL designs are optimized and mapped into TSMC 0.13 $\mu$ m standard cell library using Synopsys Design Compiler. The area, delay, and power consumption will be compared based on the report of the design compiler.

In chapter 4, the low power circuit level implementations of MACs are implemented. Low power Booth recoder design via transistor sizing is presented. Various XOR gate designs are characterized by logical effort and their power consumption are compared. Efficient computation kernels of column compression tree topology, the 5-2 compressors, are designed ranging from the circuit topology to the circuit style. The proposed power-speed tradeoffs method will be used in column compression stage and final addition stage to increase speed while save power.

In chapter 5, micro-architectural optimization techniques are discussed. A MAC functional unit for general DSP processors is described. Pipelining and parallelism are two major techniques for micro-architectural optimization. In the pipelining technique, the relation between pipeline stage number and supply voltage is very important. More pipeline stages results lower supply voltage and power saving under the same delay constrain. On the other hand, parallelism exploitation of MAC in DSP processors is a usual way to increases performance. VLIW architecture based DSP processors embeds SIMD feature instructions is a trend for state of the art DSP processors. Variable precision and variable pipelined reconfigurable MAC are discussed in this chapter.

## Chapter2

### Power-Speed Tradeoffs in Datapath Structures

In this chapter, we explored the tradeoffs between power and speed with a fixed architecture of the datapath. To effectively optimize the power consumption of a datapath while maintaining speed is crucial to first identify the critical path on it. Based on a simple delay model, named logical effort, which is suitable for designing fast CMOS circuits, the critical path of the datapath will be obtained. Under a fixed delay constrain, power consumption can be minimized through the choice of the supply voltage, transistor threshold, and device size.

At first, Sec. 2.1 is the overview of MAC. In Sec. 2.2, the method of designing fast CMOS circuits will be described. Next, the components of power consumption and the low-power technique of implementing datapath are discussed in Sec. 2.3. Finally, Sec. 2.4 presents theoretical studies of true power minimization for designing a datapath. For Sec. 2.5, we make some conclusions of this chapter.

#### 2.1 Overview of Multiplier-Accumulator Unit

Within either DSP processors or RISC processors, there is a MAC unit that function as computation kernel to perform multiplications and accumulations within given latencies. For portable applications, the primary design challenge of MAC in the DSP processors is the limited power consumption while maintaining acceptable latencies. There have been reported a number of works on low latency and low power MAC design and implementation, we categorized it into two abstractions levels, the architectural level and the circuit level:

In the architecture level, [2.1], [2.2], and [2.3] presented high speed MAC design. Reducing transition activities [2.4], programmable hardware to deactivate the unused cell of MAC [2.5], and redundant binary representation of MAC [2.6], are ways to minimize active power consumption of a MAC unit.

In the circuit level, [2.7], [2.8], and [2.9] use proposed XOR-XNOR gates to design high speed and low power MAC. [2.3], [2.10], and [2.11], proposed high speed and low power compressors as the kernels of MAC. New logic style, named SRPL [2.12], is used to design pass transistor logic to enhance its speed, to restore the voltage level and to

reduce the power consumption of MAC.

In this work, we attempt to develop a high speed micro-architecture MAC based on [2.3], and minimize its power consumption. According to this high speed MAC architecture, how to minimize the power consumption in this datapath structure is our ultimate objective. For high speed MAC unit design, logical effort [2.13], a method for designing fast CMOS circuits, will be used to achieve minimum delay under fixed micro-architecture. For designing low power, an effective methodology, named true-power minimization [2.14] [2.21], which is a sensitivity-based power-speed tradeoff methodology will be exploited to minimize the power consumption of high speed MAC unit.

## 2.2 Logical Effort

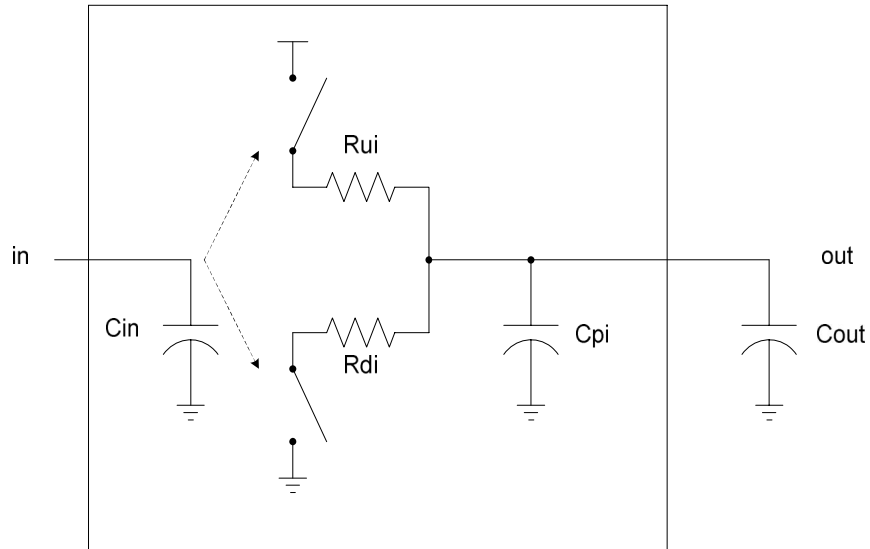
Typical datapath designs are either latency or throughput constrained. A latency-constrained design has to finish computation by a given deadline while throughput-constrained designs must maintain a required data throughput. The architectural optimization techniques, such as pipelining or parallelization, are works effectively for the throughput-constrained designs. However, with a fixed architecture, the circuit level optimization techniques, such as logical effort, are applicable to the latency-constrained designs.

### 2.2.1 Model of Logic Gate

Logical Effort is founded on a simple model of the delay through CMOS inverter. It describes delays caused by the capacitive load from which the logic gate drives and by the topology of the logic gate. This section derived an electrical model that approximated the behavior of a CMOS logic gate is shown in Figure 2.1. The logic gate is modeled by four quantities:  $C_{in}$ ,  $R_{ui}$ ,  $R_{di}$ , and  $C_{pi}$ , which are determined by the particular logic gate function, the performance of transistors in that particular CMOS process, and so on. In order to obtain the information of a particular logic gate, we scale the widths of all transistors in the template circuit by a factor  $a$  and leaving the transistor lengths unchanged. Figure 2.1 shows the expressions which are used to represent the relationship between the quantities of model and the properties of template circuit. Scaling a transistor's width of template circuit increases its gate capacitance by the scale factor  $a$ , but decreases its resistances by the scaled factor.

The delay in a logic gate modeled by logic effort is a RC delay which charging and discharging the capacitance to the output capacitance can be derived following.

$$d = \kappa R_i (C_{out} + C_{pi}) \quad (2.1)$$



$$C_{in} = \alpha C_t$$

$$R_i = R_{ui} = R_{di} = R_t/\alpha$$

$$C_{pi} = \alpha C_{pt}$$

- $\alpha$  scale factor  
 $C_t$  input capacitance of template circuit  
 $R_t$  pulldown and pullup resistances of template circuit  
 $C_{pt}$  parasitic capacitance of template circuit  
 $R_{ui}$  pullup resistance  
 $R_{di}$  pulldown resistance  
 $C_{pi}$  parasitic capacitance  
 $C_{out}$  load capacitance

Figure 2.1 Conceptual model of a CMOS logic gate

$$= \kappa \left( \frac{R_t}{\alpha} \right) C_{in} \left( \frac{C_{out}}{C_{in}} \right) + \kappa \left( \frac{R_t}{\alpha} \right) (\alpha C_{pt}) \quad (2.2)$$

$$= \kappa R_t C_t \left( \frac{C_{out}}{C_{in}} \right) + \kappa R_t C_{pt} \quad (2.3)$$

$$= \kappa R_{inv} C_{inv} \left( \frac{R_t C_t}{R_{inv} C_{inv}} \frac{C_{out}}{C_{in}} + \frac{R_t C_{pt}}{R_{inv} C_{inv}} \right) \quad (2.4)$$

where  $\kappa$  is a constant characteristic of the fabrication process

We can rewrite Equation 2.4 to obtain the key equations of logical effort:

$$d = \tau(gh + p) = \tau(f + p) \quad (2.5)$$

$$\tau = \kappa R_{inv} C_{inv}, \quad g = \frac{R_t C_t}{R_{inv} C_{inv}}, \quad h = \frac{C_{out}}{C_{in}}, \quad p = \frac{R_t C_{pt}}{R_{inv} C_{inv}} \quad (2.6)$$

where  $C_{in}$  is the input capacitance of the inverter template circuit, and  $R_{inv}$  is the resistance of the pullup or pulldown transistor in the inverter template circuit.

Equation 2.5 shows the delay of a logic gate is composed of three components, the first is called the parasitic delay  $p$ , the second is called the effort delay or stage effort  $f$ , the third is a unit delay  $t$ . The parasitic delay  $p$  represents the intrinsic delay of the gate due to its internal capacitance which is almost independent of the size of the transistors. The effort delay  $f$  depends on the output load and on the input driving properties of the gate which introduce two related terms for these effects: the electrical effort  $h$  characterizes the output load, while the logical  $g$  captures input driving properties of the logic gate. The effort delay of the logic gates is the product of these two factors. The process parameter  $t$  represents the speed of the basic transistors. Equation 2.6 presents these delay components of logic gate in terms of the delay quantity of CMOS inverter. According to this equation, if an inverter is served as template circuit for evaluating this delay model, the logical effort will be 1. Therefore, an inverter driving a copy of itself, the effort delay will also be 1.

The delay model of a logic gate, as presented in Equation 2.5, is linear model. Figure 2.2 shows this linear relationship graphically: the delay is plotted as a function of the electrical effort for an inverter and for a two input NAND gate. The slope of the line is the logical effort of gate; its intercept is the intrinsic delay. This graph shows that we can adjust the delay by adjusting the logical effort (by transistor sizing) or by choosing a logic gate with different logical effort. Once a gate type is fixed, the parasitic delay is then fixed, and later optimization procedure will not change this delay component.

### 2.2.2 Multistage Logic Networks

The total delay of a path through a combinatorial logic block can be expressed as

$$t_p = \sum_{j=1}^N t_{p,j} = t_{p0} \sum_{j=1}^N (g_j h_j + p_j) \quad (2.7)$$

By finding  $N-1$  partial derivatives and setting them to zero, we find that each stage should bear the same gate effort:

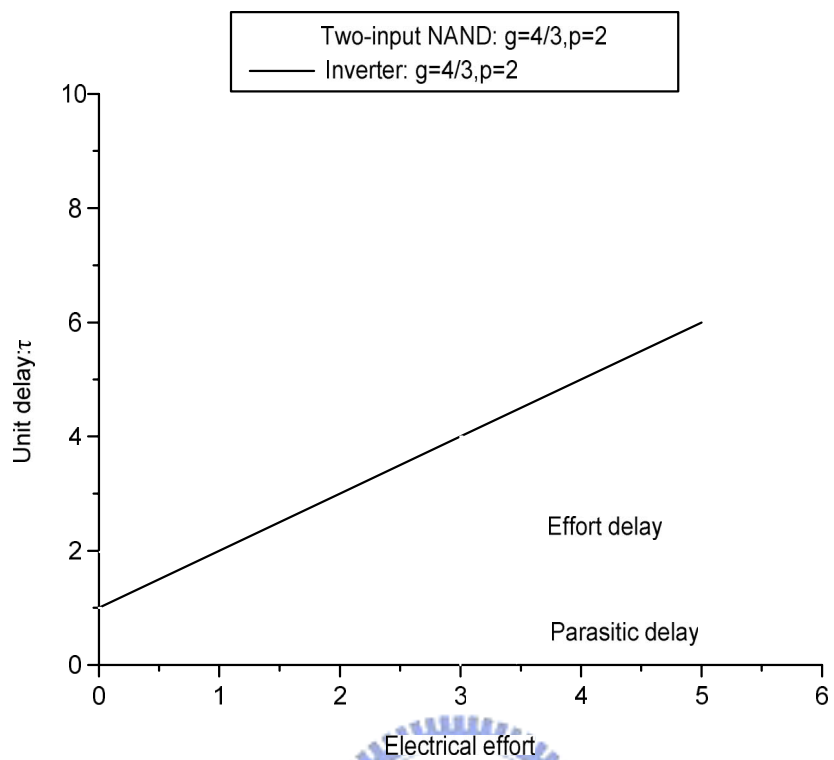
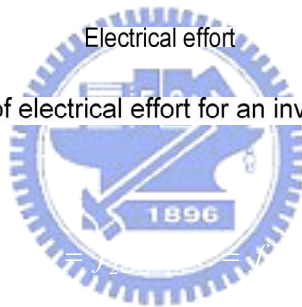


Figure 2.2 Delay as a function of electrical effort for an inverter and a two-input NAND gate



where Con-path is the load capacitance along the path we are analyzing and Coff-path is the capacitance off the path we are analyzing.

The branching effort along an entire path B is then defined as the product as the product of the branching effort at each stages along the path.

$$B = \Pi b_i \quad (2.12)$$

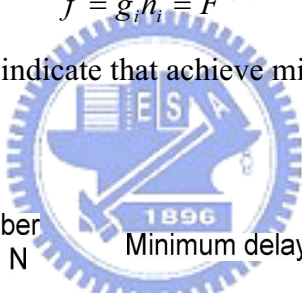
The total path effort H can be defined as follow:

$$F = GBH \quad (2.13)$$

According to the result of total path H, the best number of stages N can be determined by look up Table 2.1. The path delay will be minimum when each stage in the path bears the same stage effort. This minimum delay is achieved when the stage effort as follow:

$$\hat{f} = g_i h_i = F^{1/N} \quad (2.14)$$

where the hat over the symbol indicate that achieve minimum delay.



Path effort F	Best number of stage, N	Minimum delay $\hat{D}$	Stage effort, $\hat{f}$ , range
0	1	1.0	0-5.8
5.83	2	6.8	2.4-4.7
22.3	3	11.4	2.8-4.4
82.2	4	16	3.0-4.2
300	5	20.7	3.1-4.1
1090	6	25.3	3.2-4.0
3920	7	29.8	3.3-3.9
14200	8	34.4	3.3-3.9
51000	9	39.0	3.3-3.9
184000	10	43.6	3.4-3.8

Table 2.1 Best number of stages to use for various path efforts



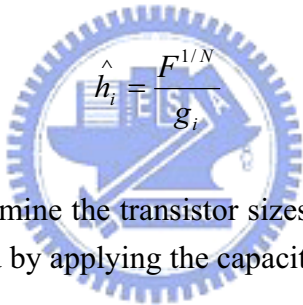
The delay D is the sum of the delays of each stage in the path can be represents as follow:

$$D = \sum d_i = \sum g_i h_i + \sum p_i = D_F + P \quad (2.15)$$

Combining these equations, from Equations 2.9 to Equations 2.15, we obtain a method of logical effort for designing fast combinational CMOS circuits, which is an expression for the minimum delay along a path:

$$\hat{D} = NF^{1/N} + P \quad (2.16)$$

From a simple computation of Equation 2.16, calculate its logical, branching, and electrical efforts, we can obtain an estimate of the minimum delay of a logic network. Equation 2.14 shows the best number of stages of a logic network, in order to achieve minimum delay along a path we must equalize the effort delay kept by each stage on a path, and choose appropriate transistor sizes for each stage of logic gate along the path. Equation 2.14 shows that each stage should be designed with electrical effort:



$$\hat{h}_i = \frac{F^{1/N}}{g_i} \quad (2.17)$$

From Equation 2.7 we can determine the transistor sizes of gate along a path. Start at the last stage and evaluate backward by applying the capacitance transformation:

$$C_{ini} = \frac{g_i C_{outi}}{\hat{f}} \quad (2.18)$$

Therefore, the input capacitance of each gate can be determined and achieve the goal by transistor sizing to optimize the speed of a logic network.

### 2.2.3 Calculating Logical Effort and Calibrating the Model

#### Logical Effort Calculation

Logical effort describes the driving capability of a gate relative to a reference CMOS inverter; it is defined as the ratio of its input capacitance to that of a CMOS inverter which delivers the same output current. This definition provides a convenient method for calculating the logical effort of a logic gate.

$$g_b = \frac{C_b}{C_{inv}} = \frac{\sum_b C_i}{C_{inv}} \quad (2.19)$$

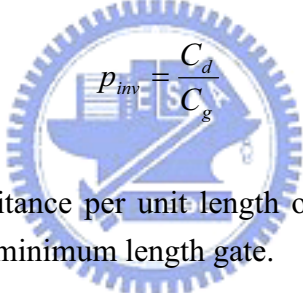
where b represents an input group of a gate, C<sub>b</sub> is the combined input capacitance of

every signal in the input group b,  $C_{inv}$  is the input capacitance of a CMOS inverter.

Figure 2.3 shows an example to calculate logical effort of logic gates. Each gate has the same driving characteristics as an inverter with a pulldown of width 1 and pullup width 2. We calculate the logical effort of the NAND gate in Figure 2.3 by counting gate capacitance from the circuit schematic. The input capacitance of one input signal, input a or input b, is the sum of the width of the pulldone transistors and pullup transistors,  $2 + 2 = 4$ . The input capacitance of the inverter is  $C_{inv} = 1 + 2 = 3$ . According to Equation 2.19, the logical effort for each input of two-input NAND gate is  $g = 4/3$ . Therefore, we applying this method to calculate the logical effort for each input of two-input NOR gate is  $g = 5/3$ .

### Parasitic Delay Estimation

The major contribution to the parasitic capacitance is the capacitance of the diffusion region of transistors connected to the output signal. We defined the parasitic delay as the ratio of the parasitic capacitance to the input capacitance of the inverter:

$$p_{inv} = \frac{C_d}{C_g} \quad (2.20)$$


where  $C_d$  is a diffused capacitance per unit length of contact diffusion,  $C_g$  is a gate capacitance per unit length of minimum length gate.

In general, we adopt a nominal value of parasitic delay of inverter  $p_{inv} = 1.0$  ( $C_d/C_g = 3/3$ ). Then we can estimate the parasitic delay of other logic gates from following equation:

$$p = \left( \frac{\sum w_d}{1 + \gamma} \right) \cdot p_{inv} \quad (2.21)$$

where  $w_d$  is the width of transistors connected to the logic gate's output,  $\gamma$  is the ratio of PMOS to NMOS width in an inverter.

This approximation can be used to estimate arbitrary logic gate. For example, a two-input NAND gate has one pulldown transistor of width 2 and two pullup transistors of width  $\gamma$  connected to output signal, assume that  $\gamma = 2$ , so we can get  $p = (2+2+2/1+2) = 2$ . Likewise, we can estimate the parasitic delay of a two-input NOR  $p = 2$ . This quantity of parasitic delay can also be measured from calibration method, will be discussed later.

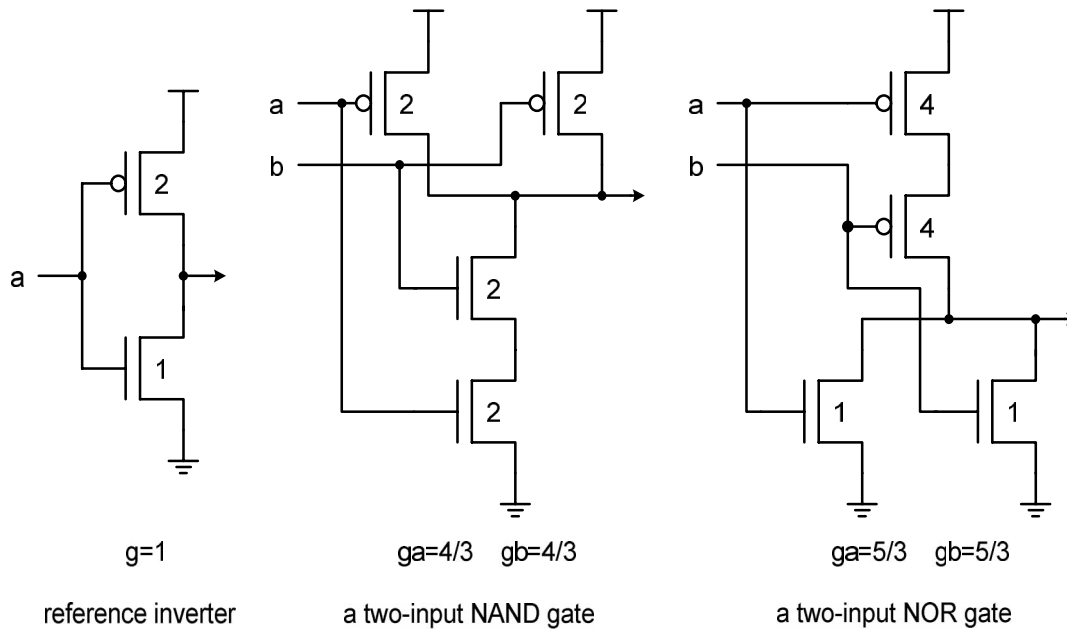


Figure 2.3 Calculating logical effort of simple gates

### Calibration Technique

Previously, we have discussed the method of calculating the logical effort and the parasitic delay of a logic gate. Now, we use more accurate method to measure these quantities by calibrating the model and show how to design a suitable test circuit to obtain these two quantities.

We calibrate by measuring the delay of a CMOS inverter as a function of its electrical effort. For different value of electrical effort, we simulate the average propagation delay of this inverter ( $t_p = (t_{p_{lh}} + t_{p_{hl}})/2$ ). Figure 2.4 shows these simulation data, plots these data, and connect them as line segments. Because the logical effort of inverter is defined as 1, from Equation 2.5 the delay will be  $d = t(h + p_{inv})$ . As Figure 2.4 shows, line segments which connect the points will have slightly different slope  $t$ . When  $h = 0$ , one of the line segments will intercept the y axis at  $d = t p_{inv} = 12.78 \text{ ps}$ . When  $h = 2$ ,  $d = t(2 + p_{inv}) = 2t + t p_{inv}$ , thus  $t = 11.34 \text{ ps}$ . Figure 2.4 also shows the evaluation of  $t$  at other electrical effort, the result of average  $t$ , and the result of parasitic delay.

Several suitable test circuits for measuring the logical effort and parasitic delay of an inverter are shown in Figure 2.5-1 ~ Figure 2.5-3. The test circuit has four stages. The first two stages are designed for considering the input slop. The third stage is the gate we want to characterize. The final stage serves as a load on the third stage.

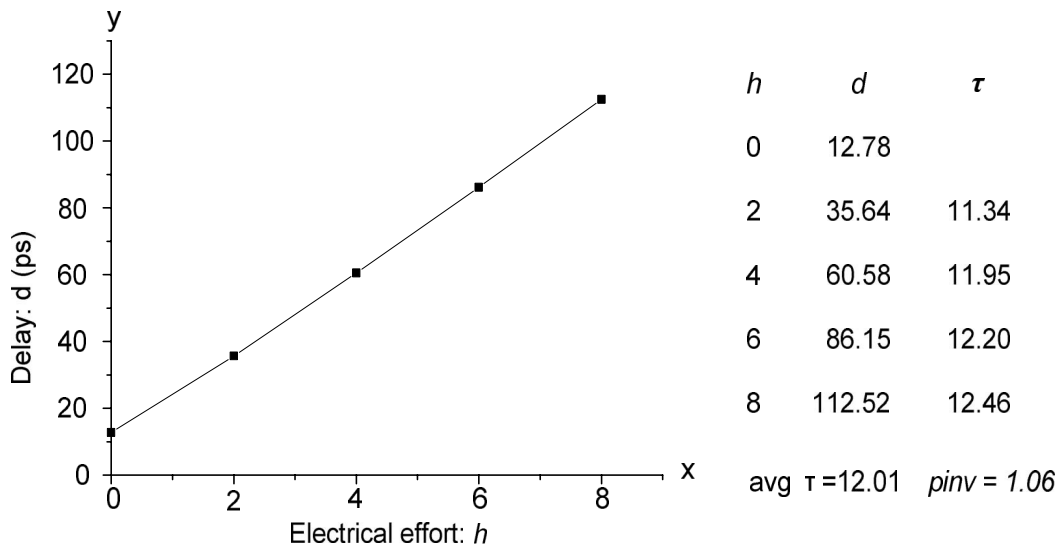


Figure 2.4 Simulated delay of inverters driving various loads. Results from tsmc 0.13 $\mu$ m 1.2v process.

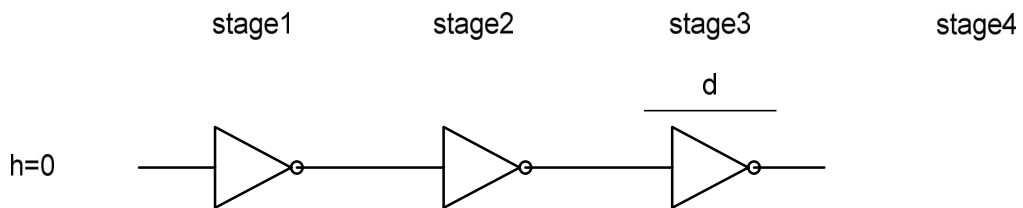


Figure 2.5-1 Test circuit of inverter calibration for electrical effort  $h=0$

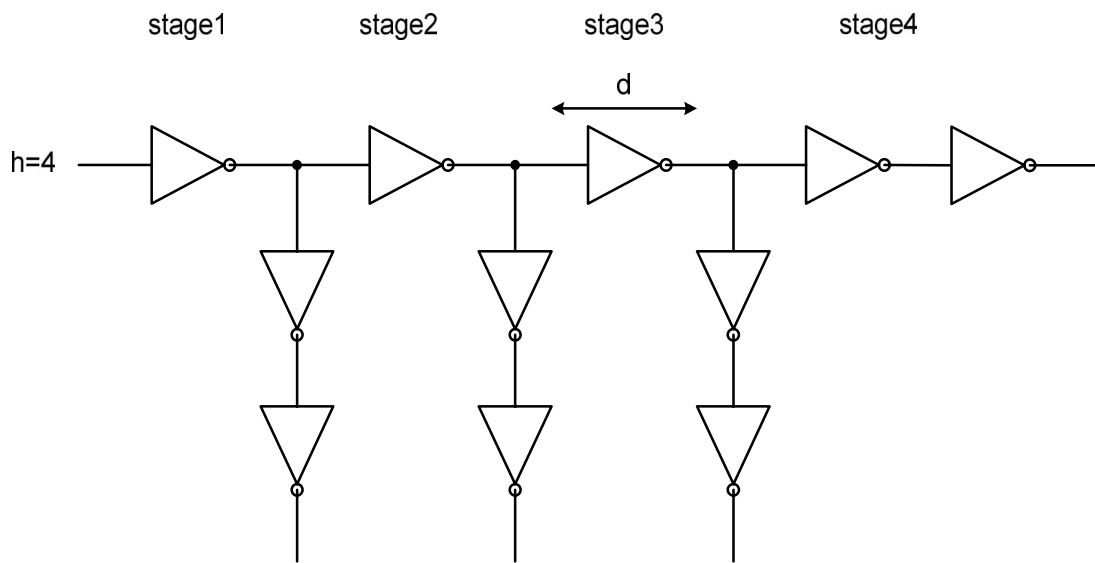


Figure 2.5-2 Test circuit of inverter calibration for electrical effort  $h=2$

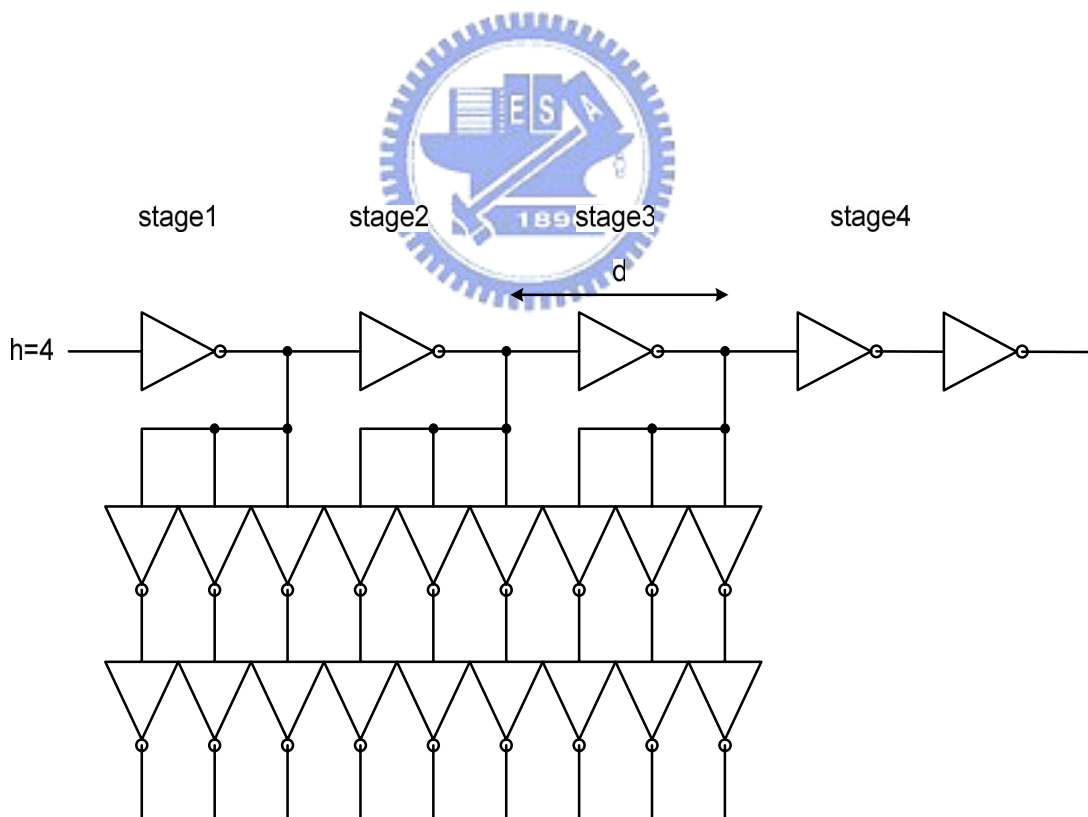


Figure 2.5-3 Test circuit of inverter calibration for electrical effort  $h=4$

## 2.3 Low Power Techniques in Datapath Structures

Energy and power are related. Power is the time rate of consumption of energy, while energy is consumed by a circuit to perform a designated computation. Both of them are useful to different applications. If we are concerned about battery life, energy is the more suitable metric to think about. However, if we are concerned with heat and reliability issue, power is the appropriate metric to consider. Throughout this thesis, evaluating a circuit by its power dissipation is equivalent to evaluate by its energy consumption. Because in here our focus of power or energy consumption is on a datapath structure, the achievable minimum delay time is fixed as a baseline of optimization.

In [2.15], a classification of low power techniques in datapath was presented to summarize adequate and effective power minimization methods for different operation mode of datapath. In this section at first, the power source of CMOS circuits will be discussed briefly and subsequently discuss the low power techniques in datapath structure.

### 2.3.1 Power Dissipation in CMOS Circuits

There are four main sources of power dissipation: dynamic switching power due to charging and discharging circuit capacitances, short circuit current power due to finite signal rise time and fall time, leakage current power from reverse biased diode and sub-threshold conduction, and static biasing power found in some types of logic style (For example, pseudo-NMOS).

CMOS has become the dominant process technology for digital circuits. And it is convenient and accurate to estimate the power consumption of CMOS circuits using simple equations. These equations which are used to model the power source of CMOS circuits can be found in many references [2.15].

### 2.3.2 Power Minimization Techniques

With a fixed datapath structure, speed, area, and power can be trade off through the choice of the supply voltage(s), threshold voltage(s), and transistor sizes. Table 2.2 categorized these optimization method. They are classified as follows:

#### *Enable Time*

Supply voltage and threshold voltage can be either assigned statically during the design time or changed dynamically at run time, while the transistor size are fixed at design phase. Other techniques address the time that a datapath enter into idle mode (sleep mode).

	<i>Constant latency</i>	<i>Variable latency</i>	
Enable Time Source	Design Time	Run Time	Sleep Mode
Active	<ul style="list-style-type: none"> <li>• Optimal VDD</li> <li>• Transistor Sizing</li> </ul>	<ul style="list-style-type: none"> <li>• Dynamic Voltage Scaling</li> </ul>	<ul style="list-style-type: none"> <li>• Clock gating</li> </ul>
Leakage	<ul style="list-style-type: none"> <li>• Optimal Vth</li> </ul>	<ul style="list-style-type: none"> <li>• Variable Vth</li> </ul>	<ul style="list-style-type: none"> <li>• Sleep transistors</li> </ul>

Table 2.2 Power-speed optimization techniques

### *Dissipation Source*

From the point of view of power dissipation source of CMOS circuits, the power consumption of a datapath can be classified as active power (dynamic) or leakage power (static).

In the Enable time classifications, the design complexity and power saving capability as well as power saving flexibility are emphasized. When we fixed the supply voltage and the threshold voltage, the power saving capability and flexibility are also fixed without hardware overhead. If the supply voltage and the threshold voltage are changed dynamically, the power saving capability and flexibility will have great improvement but result in complicated hardware design and may need software support. In designing a datapath, therefore, the design time techniques are more applicable for power reduction. When we are considering a microprocessor design, for example, the run time techniques can significantly improve processor energy-efficiency, but result in challenging dynamic voltage-scaled system design.

In this thesis, we would like to focus attention on the design time and sleep mode power reduction technique exploration, which are more effective methods for datapath structure.

### *2.3.3 Design Time Power Reduction Techniques*

There are three major knobs can be fine tuned to trade off power-speed at design phase: the supply voltage, the threshold voltage, and transistor sizing. We demonstrate these techniques as follow:

### *Reducing Supply voltage*

The optimization of supply voltage is the most effective way for power reduction, because it results in quadratic power saving. There are two major approaches for optimizing supply voltage, one is lowering the supply voltage directly, and the other is using multiple supply voltages. When we reduce the supply voltage, the delay of CMOS circuits will increase inversely with supply voltage. When designing a datapath, this loss of speed can be compensated by replacing the underlying algorithm or by arranging the micro-architecture. For example, a ripple-carry adder can be replaced by a faster, more advanced algorithm to increase its speed, although it also translates into a larger physical and switching capacitance, we can lower the supply voltage under a constrain of the same speed to reduce power consumption. From the point of view of micro-architecture, parallelism and pipelining are also the effective ways to compensate for the loss in speed. These approaches presented tradeoff area for power. When the design is not area constrained, we should optimize carefully, because parallelism and pipelining introduces extra routing overhead which might cause additional power dissipation.

### *Multiple Supply Voltages*

The alternative approach for optimizing supply voltage is to selectively decrease the supply voltage on some of the gates based on the path delay distributions: a fast paths but not a critical path which finish the computation early. There are two gradations of multiple supply voltage optimizations, one is gate-level voltage selection the other is module-level voltage selection. In gate-level voltage selection, employing only two supply voltages is a more practical implementation [2.16]. While in module-level voltage selection, a system architecture and chip implementation methodologies which is called “Voltage Island” is proposed by [2.17] . When combining multiple supply voltages on each gradation, level converters are required for a module (gate) at the lower supply voltage to drive a module (gate) at higher voltage. Without level converter, a module (gate) supplied by lower supply voltage drives a gate at higher supply voltage, the PMOS transistor never turns off. Figure 2.6 demonstrates this static current problem.



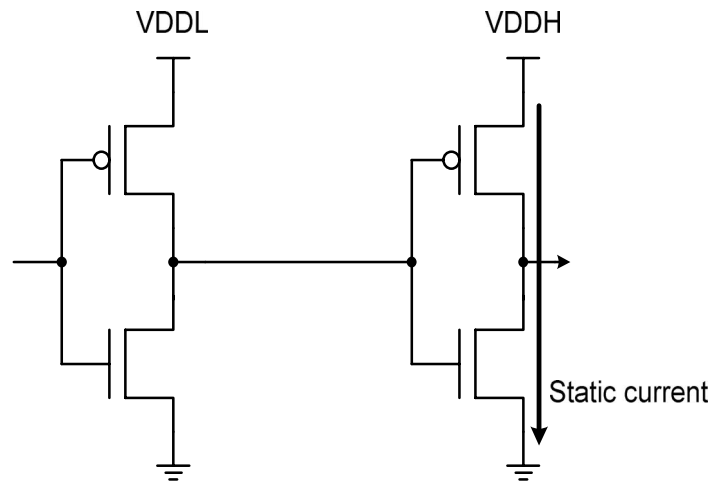


Figure 2.6 Static current from VDDL to VDDH

### *Multiple Threshold Voltages*

As technology scaling lower the supply voltage, the threshold voltages have been scaled down to maintain speed. Sub-threshold leakage currents increase exponentially as threshold voltage is reduced. In order to reduce leakage current without compromising speed, multiple threshold voltages are used. The use of multiple threshold voltages has been discussed in various ways, such as the use of low-V<sub>t</sub> devices only on critical paths.

### *Transistor Sizing*

Transistor sizing is an effective power-reduction method for datapaths, because the major power dissipation is consumed inside the block rather than in driving the external load capacitance. Using logical effort, the minimum delay of a datapath can be achieved, and the transistor size for minimum delay can be determined in Equation 2.18. Sizing was then used to bring each path to its maximum speed to achieve high performance. Under this circumstance, the main objective of transistor sizing is to downsize the gate off the critical path to save power.

#### *2.3.4 Run Time Power Management*

Dynamic voltage scaling (DVS) and Dynamic threshold scaling (DTS), techniques to dynamically vary microprocessor's performance and power consumption during the processor's run time. There are three major components for implementing DVS in a general-purpose microprocessor system: an operating system that can dynamically vary the processor's speed, a regulation loop that can generate the minimum voltage required for the desired speed, and a microprocessor that can operate over a wide voltage range.

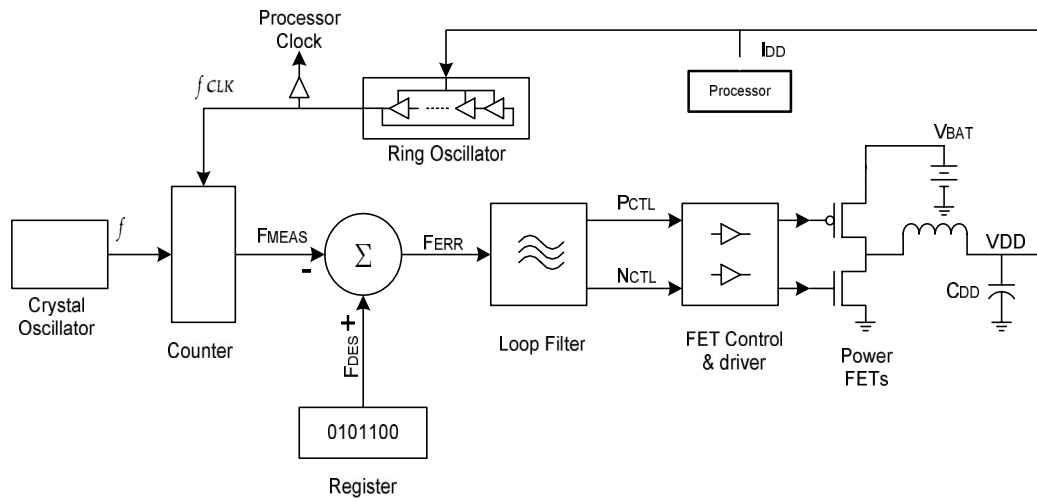


Figure 2.7 A dynamic voltage-scaled system



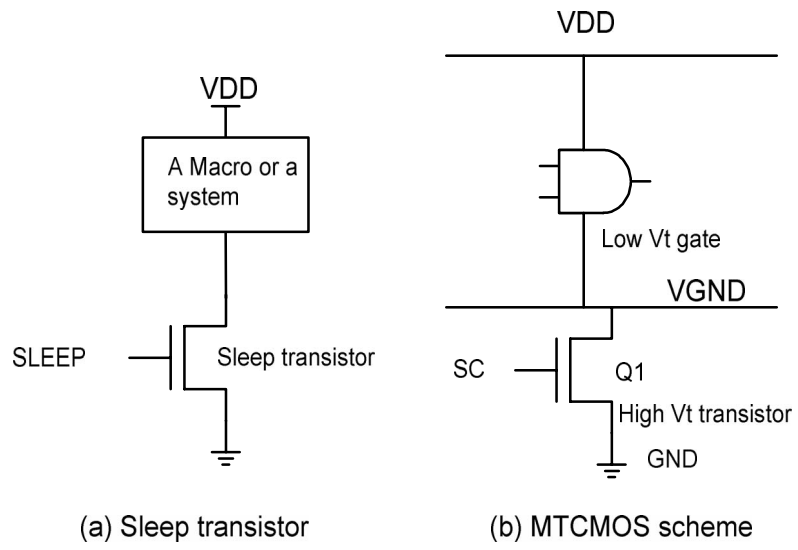


Figure 2.8 Sleep transistors and MTCMOS scheme



By combining these two variables, Var1+Var2, we can get the better result of lower energy consumption and higher speed, as Figure 2.9 (d) shows. But the question is how far are we from the most energy efficient solutions, Figure 2.9 (d) also demonstrates this problem.

Design points on the dotted line are the most energy efficient. Our goal is to achieve the highest speed while minimize the energy consumption as possible. When energy is constrained, we must use the most energy efficient solution to achieve highest speed, the best performance point, as indicated in Figure 2.9 (e).

Overall, as Figure 2.9 (f) shows, starting from an unoptimized design point, we either want to design the circuit in highest speed while bringing the circuit under the energy constrain, or vice versa. The true power minimization is achieved when the power reduction potentials of all tuning variables are balanced [2.14]. We will demonstrate the circuit sensitivity for each of tuning variables to explore its power saving potentials individually. This is helpful in designing a energy-speed efficient datapath.

#### 2.4.2 Energy-Delay Sensitivities

In energy-speed optimization, the objective is to utilize surplus time for maximum energy reduction. There are three major tuning variables, the supply voltage, the threshold voltage, and the transistor size, which can be trade off power and surplus time of a datapath at circuit level. We want to know how much energy it would cost to increase the speed by some amount, using each tuning variable named x [2.14]:

$$S(X) = \frac{\partial E / \partial x}{\partial D / \partial x} \Big|_{x=X} \quad (2.22)$$

We call this quantity the energy-delay sensitivity for tuning variable  $x = X$ . As pointed out by [2.18], the energy-efficient design is achieved when the marginal cost of all the tuning variables are balanced. In [2.14], the key idea is that: At optimal point, all sensitivities should be the same. Equation 2.23 shows this property.

$$\Delta E = S(A) \cdot (-\Delta D) + S(B) \cdot \Delta D \quad (2.23)$$

where A, B represent the tuning variables.

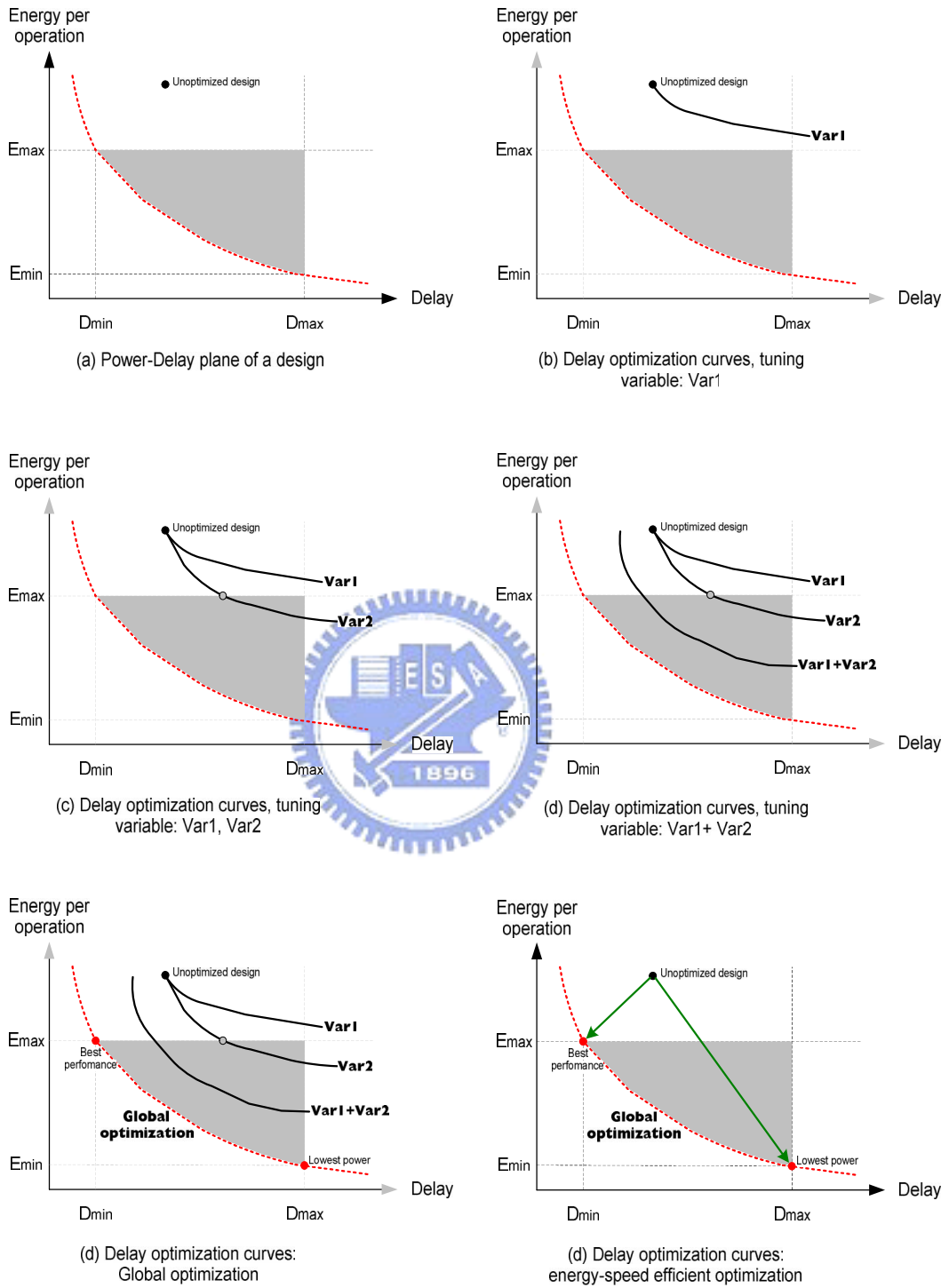


Figure 2.9 Energy-speed optimization process

### Power and Delay Model

In order to formulate the sensitivities of the supply voltage, the threshold voltage, and transistor sizing, we need to have models for power and delay. The alpha-power law model [2.20] is expressed as the logical effort formulation [2.13] for modeling delay:

$$t_p = \frac{K_d V_{dd}}{(V_{dd} - V_{on} - \Delta V_{th})^{\alpha_d}} \cdot \left( \frac{W_{out}}{W_{in}} + \frac{W_{par}}{W_{in}} \right) = \tau_{ref} \cdot g \cdot \left( h + \frac{p}{g} \right) = \tau_{ref} \cdot d \quad (2.24)$$

where  $V_{on}$  and  $\alpha_d$  are intrinsically related to, but not necessarily equal to the transistor threshold voltage and velocity saturation index,  $\Delta V_{th}$  is the change from the standard threshold voltage given by technology,  $K_d$  is a fitting parameter,  $h$  is the electrical effort of a gate;  $p$  is the parasitic delay of that gate.

For energy model, there are two components should be take into account. The switching power model, which consists of power consuming transition probability, supply voltage, parasitic and output load capacitances:

$$e_{sw} = \alpha \cdot K_e \cdot (W_{out} + W_{par}) \cdot V_{dd}^2 \quad (2.25)$$

where  $K_e$   $W_{out}$  is the load capacitance,  $K_e$   $W_{par}$  is the parasitic capacitance of the gate,  $\alpha$  is the probability of an power-consuming transition at the output of the gate.

The leakage energy model is using the standard input state-dependent exponential leakage current model with DIBL effect:

$$e_{sw} = D \cdot W_{in} \cdot I_0(S_{in}) \cdot e^{-\frac{(V_{th}^{ref} + \Delta V_{th} - \gamma \cdot V_{dd})}{V_0}} \cdot V_{dd} \quad (2.26)$$

where  $D = d_{critical-path} \cdot \tau_{ref}$  is the cycle time,  $I_0(S_{in})$  is the normalized leakage current of the gate with inputs in state  $S_{in}$ ,  $V_{th}^{ref}$  is the standard threshold voltage provided by technology,  $V_0 = n \cdot kT/q$  and  $\gamma$  account for the sub-threshold slope and DIBL factor, respectively.

### Sensitivity to Gate Sizing

The sensitivity of energy to delay due to a change in size of a gate in stage  $i$  is given by Equation 2.27 and 2.28:

$$\frac{\partial E_{sw} / \partial W_i}{\partial D / \partial W_i} = - \frac{e_{C,i}}{\tau_{ref} \cdot (h_{eff,i} - h_{eff,i-1})} \quad (2.27)$$

$$\frac{\partial E_{LK} / \partial W_i}{\partial D / \partial W_i} = - \frac{E_{LK}}{D} \frac{e_{LK,i}}{\tau_{ref} \cdot (h_{eff,i} - h_{eff,i-1})} \quad (2.28)$$

where  $e_{C,i} = \alpha \cdot K_e \cdot (W_{in,i} + W_{par,i}) \cdot V_{dd}^2$  represents the switching energy due to capacitances of stage i,  $e_{LK,i}$  is the leakage energy of stage i as given by Equation 2.26.

$h_{eff,i} = g_i \cdot h_i$  is the effective fanout (stage effort) of stage i.

Observe that, the minimum achievable delay is designed initially which is the starting unoptimized point. The design is sized for minimum delay with equal effective fanout  $h_{eff}$ , resulting in infinite sensitivity. Therefore, the largest potential for energy saving occurs at this point. This makes sense because at minimum delay no amount of energy added through sizing can be further improving the delay.

#### Sensitivity to Supply Voltage

The sensitivity of total energy to delay, due to global supply reduction is given by Equation 2.29 and Equation 2.30:

$$\frac{\partial E_{sw} / \partial V_{dd}}{\partial D / \partial V_{dd}} = - \frac{2E_{sw} \cdot (1 - V_{on} / V_{dd})}{D \cdot (\alpha_d - 1 + V_{on} / V_{dd})} \quad (2.29)$$

$$\frac{\partial E_{LK} / \partial V_{dd}}{\partial D / \partial V_{dd}} = - \frac{E_{LK}}{D} \cdot \left( \frac{(1 - V_{on} / V_{dd}) \cdot (1 + \gamma \cdot V_{dd} / V_0)}{\alpha_d - 1 + V_{on} / V_{dd}} - 1 \right) \quad (2.30)$$

Analogous with the gate sizing approach, the design sized for minimum delay at highest supply voltage offers the greatest potential for energy saving. This potential diminishes with the reduction in supply voltage. The sensitivity of supply voltage is finite at normal point but decrease to zero when supply voltage approaches threshold voltage, because the delay becomes infinity, Figure 2.10 shows this property.

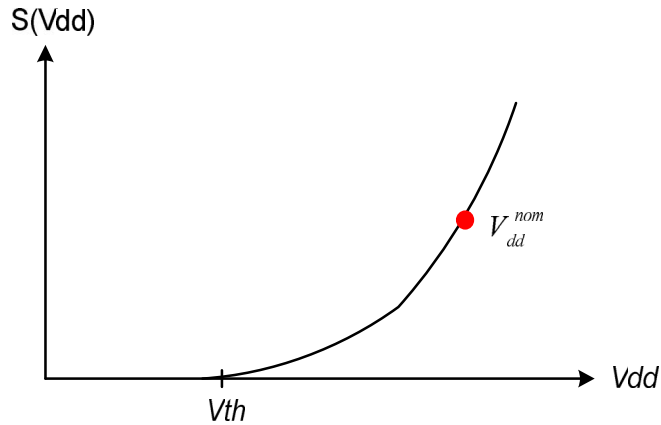


Figure 2.10 A sketch map of sensitivity to supply voltage

**Sensitivity to Threshold Voltage**

The sensitivity of total energy to delay, due to the change in threshold voltage is given by Equation 2.31:

$$\frac{\partial E / \partial(\Delta V_{th})}{\partial D / \partial(\Delta V_{th})} = -\frac{E_{LK}}{D} \cdot \left( \frac{V_{dd} - V_{on} - \Delta V_{th}}{\alpha_d \cdot V_0} - 1 \right) \quad (2.31)$$

As Figure 2.11 shows, the sensitivity to threshold voltage is opposite to that of supply voltage. As the threshold voltage is decreased, the circuit speed will be improved but resulting more leakage energy consumption.

Because the exponential dependence of the leakage energy on  $V_{th}$ . To decrease the threshold voltage becomes very expensive in terms of speed improvement.

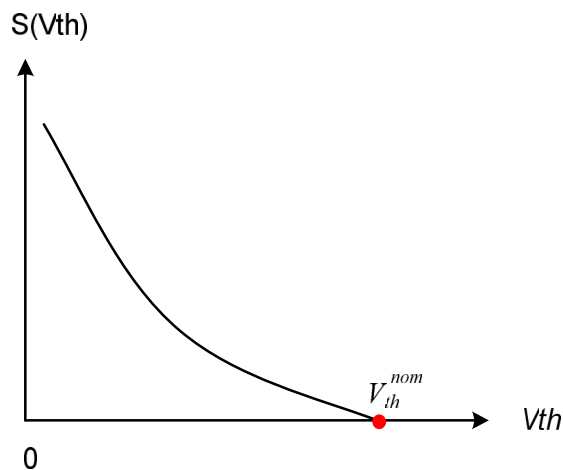


Figure 2.11 A sketch map of sensitivity to threshold voltage



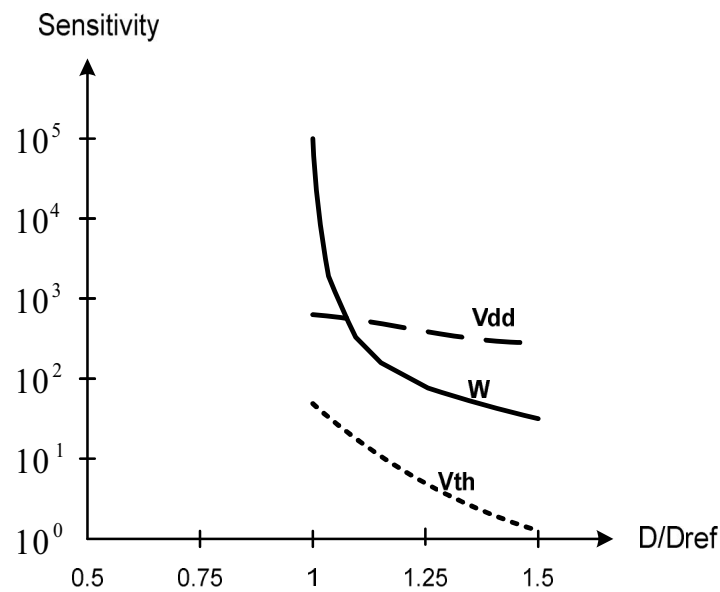


Figure 2.12 Energy-delay sensitivities of Vdd, W, and Vth

The analytical plots of energy-delay sensitivity of each tuning variable are shown in Figure 2.12. In the initial phase of tuning transistor size, the sensitivity of gate sizing is quite high which represents that it has great energy saving potential. As delay penalty growing to 10%, the sensitivities of supply voltage becomes the highest one, thus the energy saving potential goes beyond the gate sizing. The sensitivity of threshold voltage turned out far worse than gate sizing and supply voltage. Tuning these circuit level knobs to achieve balanced energy-delay sensitivity and turns out energy saving and speed improvement are the ultimate goal of circuit level optimization.

### 2.4.3 Circuit-Level Optimization

Assume that at initial phase of designing a circuit, all sensitivities were not the same (at unoptimized design point). Using the low sensitivity tuning variable to decrease the delay by delta, the energy increase is proportional to the low sensitivity. On the other hand, using the high sensitivity tuning variable to increase the delay by delta, the energy decrease is proportional to the high sensitivity resulting in net power saving with no delay penalty. Thus, the method of joint optimization of tuning variables affects the energy-speed efficiency profoundly.

Figure 2.13 shows a sketch map of joint optimization to transistor sizing and supply voltage scaling. The normal design point is optimized for minimum delay at normal supply voltage. Three curves  $f(V_{dd})$ , and  $f(W, V_{dd, new})$  represent optimization of different tuning variable, respectively. The first step is to increase supply voltage to make

the design a little faster, and then resize it bring it back to original minimum delay and save some energy.

#### 2.4.4 Micro-Architectural Optimization

In the circuit level, individual optimization results may be deceptive due to the analysis of different abstraction level. For example, if the energy of a MAC is a much smaller fraction of the total DSP processor energy than that of memory, than it might be more beneficial to lower the power of memory (make the memory slower) and increase the power of the MAC (make the MAC faster). To, this problem, a block diagram which illustrates various abstraction layers in the optimization is shown in Figure 2.14. The optimal energy-delay curves from the circuit level are used to hierarchically extend to larger blocks. These tradeoff curves conjoin with optimal  $W$ ,  $V_{dd}$ , and  $V_{th}$  are strategically combined to obtain the optimal energy-performance tradeoff for circuit macros. Energy-speed tradeoff is the objective at the circuit and micro-architectural layers. While for the macro-architectural layer, the objective is to achieve proper energy-area tradeoff. Each abstraction layer provides more degrees of freedom of knobs to optimize energy, performance, and area.

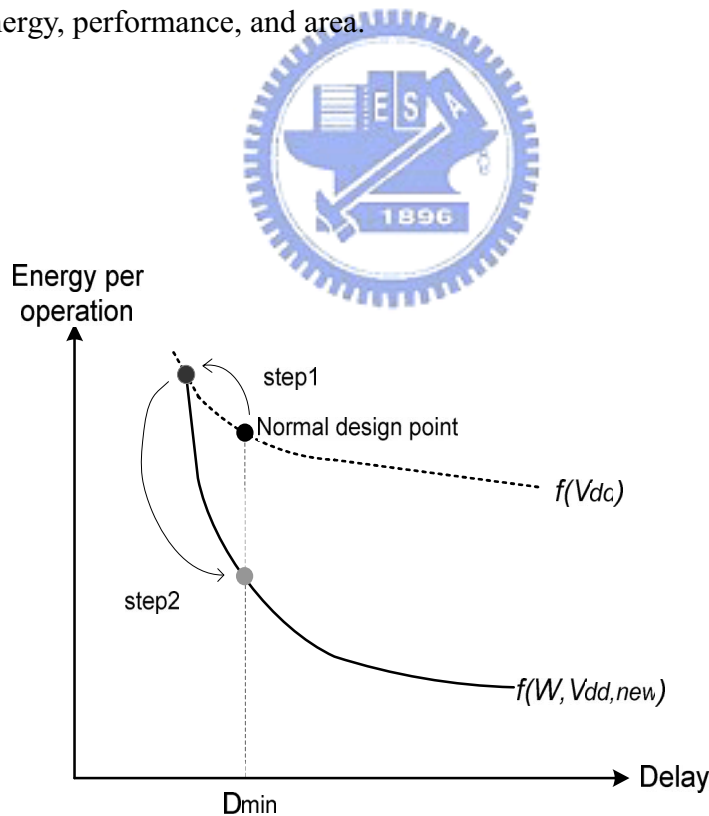


Figure 2.13 Joint optimization to transistor sizing and supply voltage

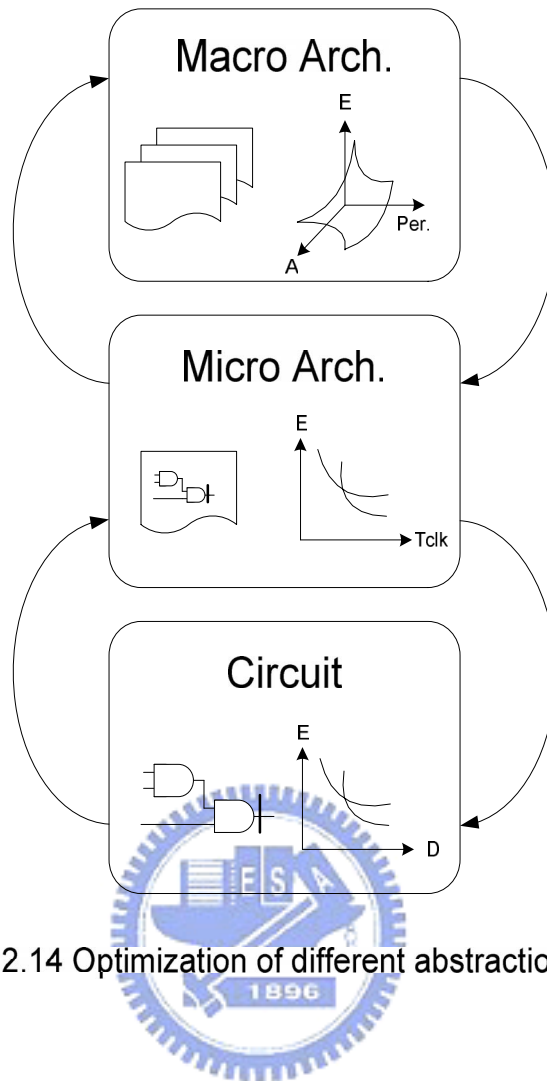


Figure 2.14 Optimization of different abstraction level

## 2.5 Conclusions

In this chapter we present basic concepts of low power circuit design techniques focus on datapath circuit block. It supports a guide for designing low power MAC. But we must take care of a point, MAC circuit design, which is one kind of VLSI computer arithmetic implementations, composed of various heterogeneous algorithms, these are Booth recoding schemes, partial product reduction topologies, and final adders. These algorithms determine the inherent speed of a MAC operation. Therefore, in designing MAC, first we must determine the underlying efficient algorithms.

# Chapter3

## High-Speed Multiplier-accumulator

### Micro-Architecture Design

#### 3.1 Background

High speed multiplier-accumulator units have become one of the essential building block in digital signal processor as well as in the general purpose processors [2.1] [2.10]. In order to increase the MAC speed, there are two stages design that need to be considered. The first one is from micro-architecture level optimization and subsequent is to complete MAC design in fast and low-power circuit style. This chapter is intended as an investigation of high speed MAC micro-architecture, implement and synthesize it in gate level in order to offers the key to an high performance MAC design.

##### 3.1.1 Basic Concepts

Figure 3.1 shows the general structure of parallel MAC for multiplying two numbers A and B and adding the number Z to accumulator. The general form of MAC operation can be presented by the equation (1) :

$$X = A B + Z \quad (3.1)$$

where A and B are two binary number of length N-bits, Z of 2N bit merged in to partial production matrix (PPM) while X has a length of at least 2N bits.

Figure 3.2 shows a conventional dot diagram, can provide information which usually be used as a guide for examining various multiplication algorithms. The micro-architecture development here will base on such kind of representation throughout this thesis. Each dot represents a binary number in specific order, if there is something wrong with the dot position, the final result may be incorrect. In fact , the number of dots in the partial product matrix is proportional to the amount of hardware cost to sum the partial products and form the final result. Thus fewer dots can be faster and require less hardware cost.

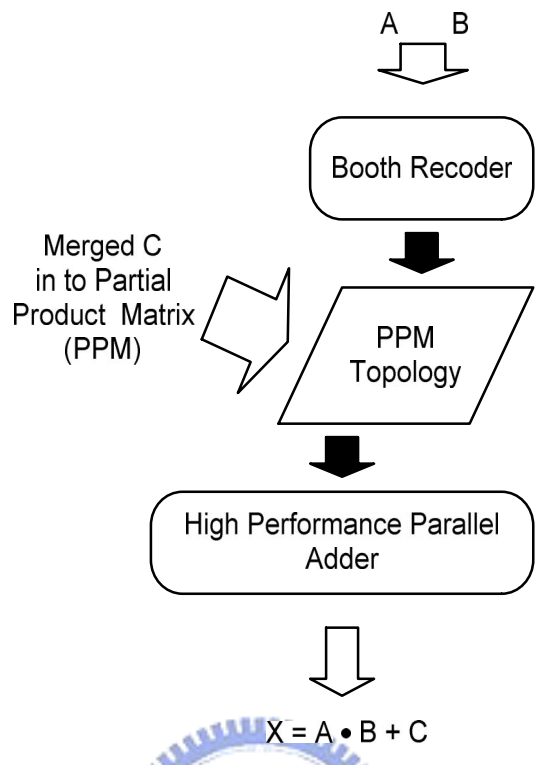


Figure 3.1 Parallel MAC structure

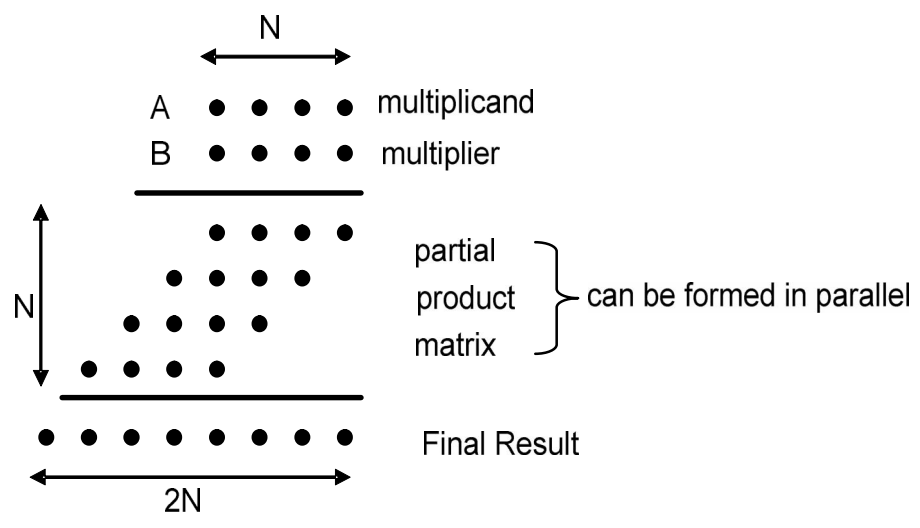


Figure 3.2 Dot diagram

The essence of the merged MAC operation is a parallel multiplication process. Therefore, designing the MAC almost the same as multipliers. From Wallace's suggestions [3.1], acceleration of the multiplication process base on following strategies: (1.) Fast partial product generation. (2.) Fast partial product reduction. and (3.) High speed final carry-propagate addition. To reduce the number of partial produces, several Booth's recoding scheme have been used extensively [3.2] [3.3] [3.4]. In the partial product reduction step, the multi-operand addition is usually accomplished using carry save adder array and tree topology to reduce partial product matrix into two rows. Eventually, the speed of multiplier is improved via optimization of the final-propagation adder, to add the non-uniform sum and carry in fastest fashion.

In thesis, Our focus is to develop a high speed micro-architecture in gate level. Thus, we examine several high speed Booth's recoding scheme, two efficient partial product reduction topologies, and several high performance parallel adders. We will implement these three major part of parallel MAC hard macro by using verilog HDL gate-level modeling and synthesize them in TSMC 0.13 $\mu$ m generic cell library. After evaluate these micro architecture, the later circuit level implementation will base on this high speed micro-architecture to do power and delay optimization.

### *3.1.2 Booth Algorithm*

A common method for reducing dots is to use Booth's algorithm [3.5] . From hardware implementation aspect, typically, multipliers are implemented using the modified Booth's algorithm (radix-4) [3.6] [3.7] [3.8]. As Figure 3.3 shows, the multiplier is partitioned into three – bit groups that overlap by one bit. According to Table 3.1, each group of three is recoded and forms one partial product. The number of dots has decreased after Booth recoding, but this reduction can be done further more using higher radix recoding scheme, such as radix-8. However, with radix-8 recoding all partial product term may be implemented with simple shifts and negation with the exception of  $\pm 3$  multiple. This multiple require an additional high speed adder. The delay and hardware overhead of this additional adder stage is a major disadvantage of higher radix recoding. Therefore, we will only focus on the radix-4 recoding scheme.

Partial Production Selection Table			
Multiplier Bits			Selection
0	0	0	0
0	0	1	+ Multiplicand
0	1	0	+ Multiplicand
0	1	1	+ 2 X Multiplicand
1	0	0	+ 2 X Multiplicand
1	0	1	- Multiplicand
1	1	0	- Multiplicand
1	1	1	0

Table 3.1 Modified Booth Algorithm

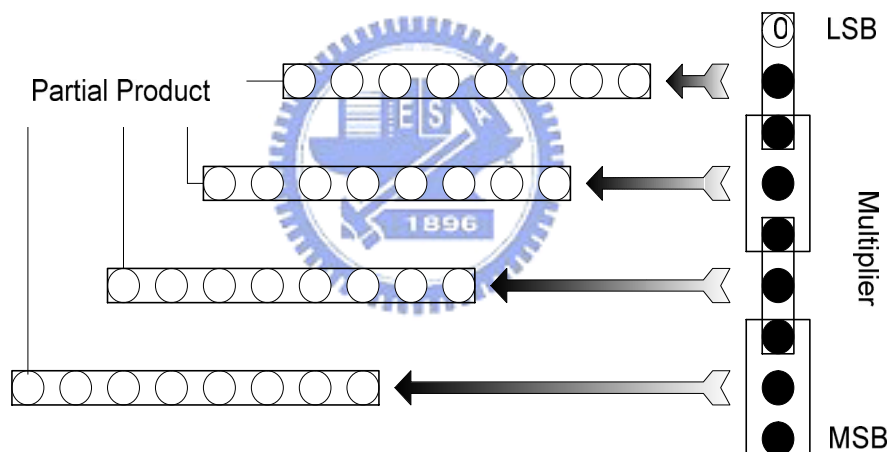


Figure 3.3 Modified Booth Algorithm

at a time, the double array add ten partial product at a time as well as the higher-order array add eighteen partial product at a time. In the double array array, when all the partial product are accumulated the two partial sum are combined using [4:2] compressor. The idea behinds the higher order array is to partition the array into more sub-arrays and again combined them using [4:2] compressor.

The basic elements used in reducing partial product matrix are counters or compressors. The (3,2) counter is essentially a full adder, also referred to as a carry save adder. It's simply a binary full adder that takes three bits of the same weighted as input and produces a sum bit and a carry bit. Specifically, counters adds up the 1's in a K-bit column outputting a log K wide count. While Compressors adds up the 1's in K-bit column plus j carry in outputting j carry out and two wide count. As Figure 3.5 shows a [4:2] compressor implemented with (3,2) counters, all of the inputs (4 external plus one internal) have the same weight and the internal output is carried to the next higher weight position.

In tree topology, a very fast structures for summing partial product, the counters and compressors are connected mostly in parallel along the vertical direction, the difference is in the interconnections between counters or compressors. In such a parallel connection, the placement of counter or compressor create a three dimensional structure. However, integrated circuits are planner; these trees must be

flattened to fit the two dimensional layout plane. As Figure 3.6 shows, the output of these counters which be flattened is therefore have to pass on top of the intermediate counters that lie between the input and output counters. Consequently, Flatten counter or compressor results the more complicated and irregular wire connection. It's difficult to design and layout [3.9].

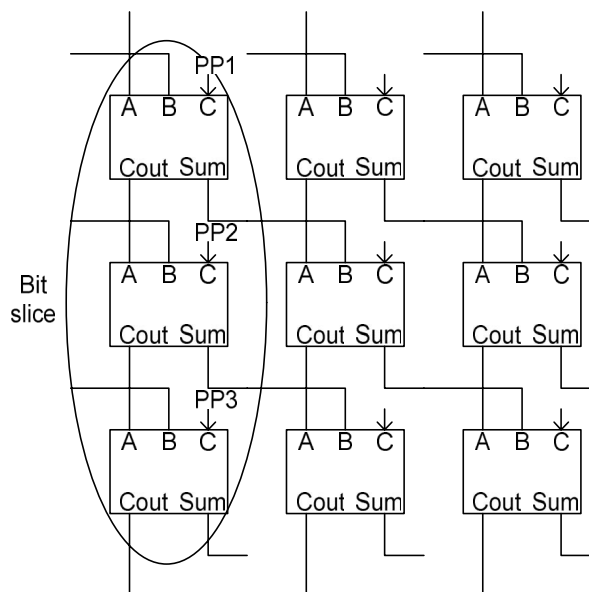


Figure 3.4 (a) single array topology



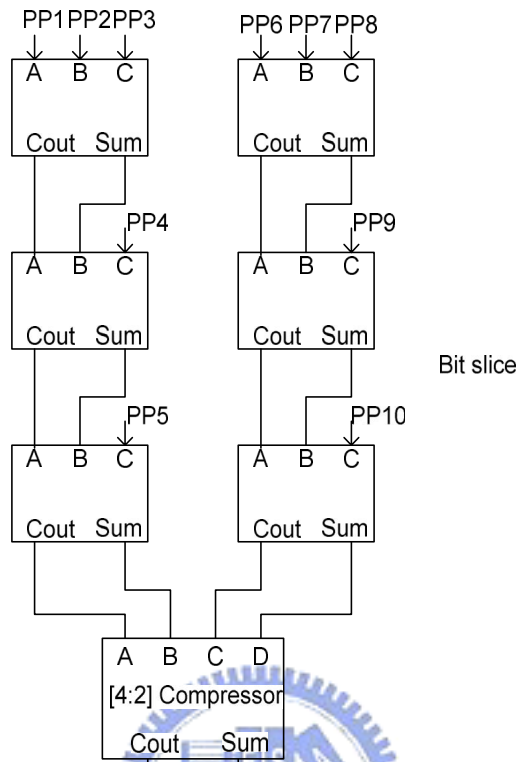


Figure 3.4 (b) double array topology

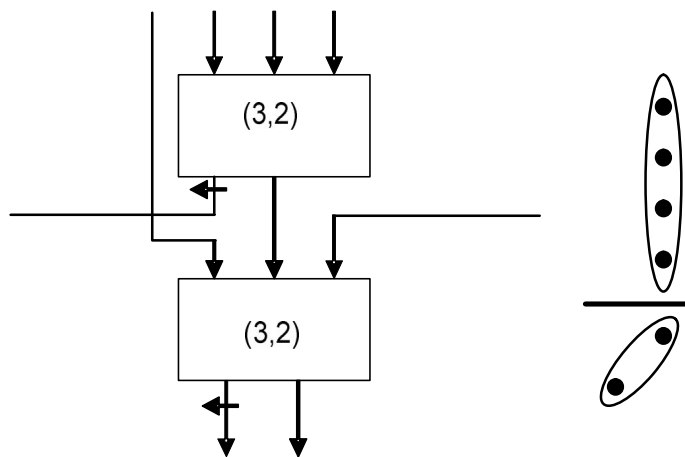


Figure 3.5 [4:2] Compressor

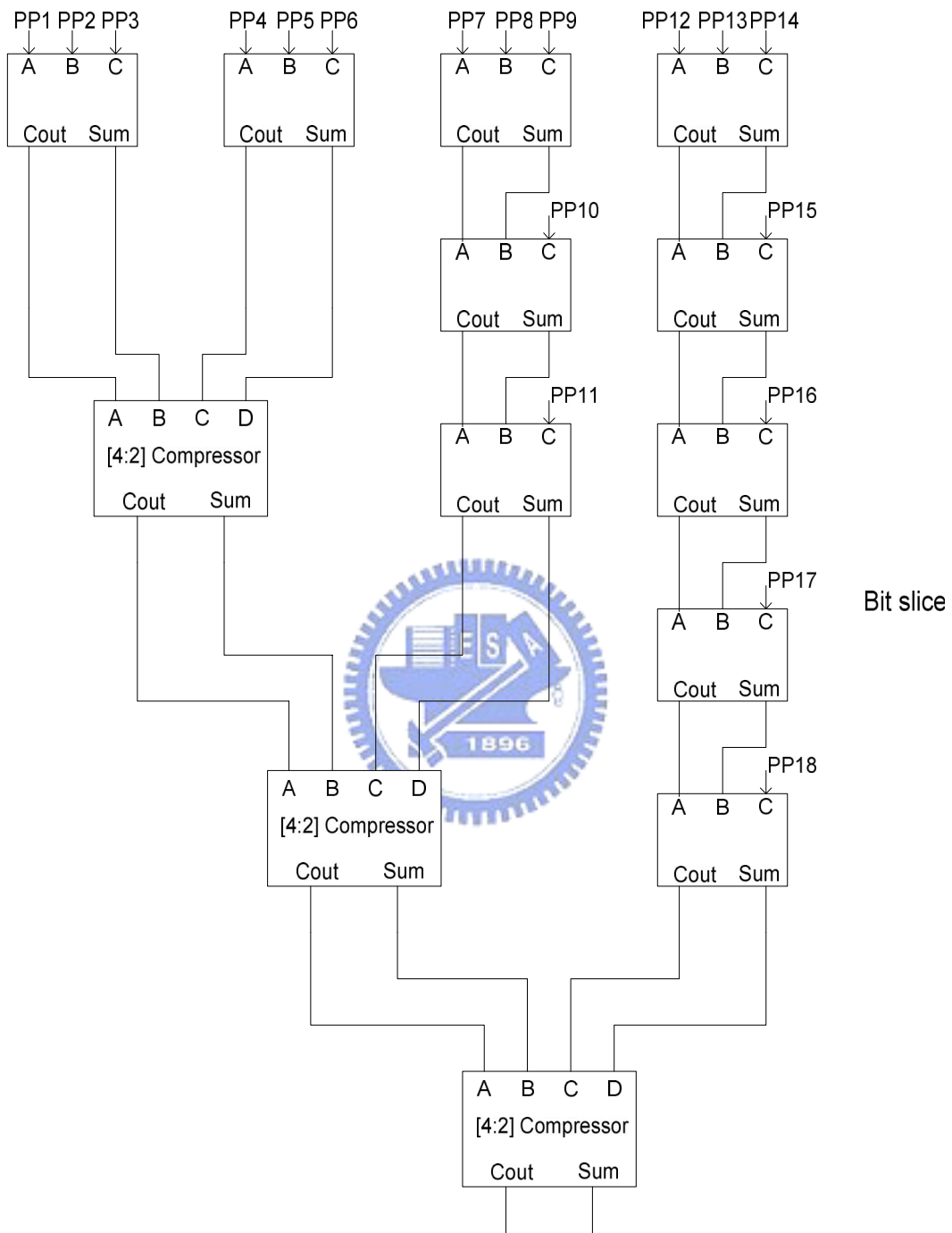


Figure 3.4 (c) higher order topology

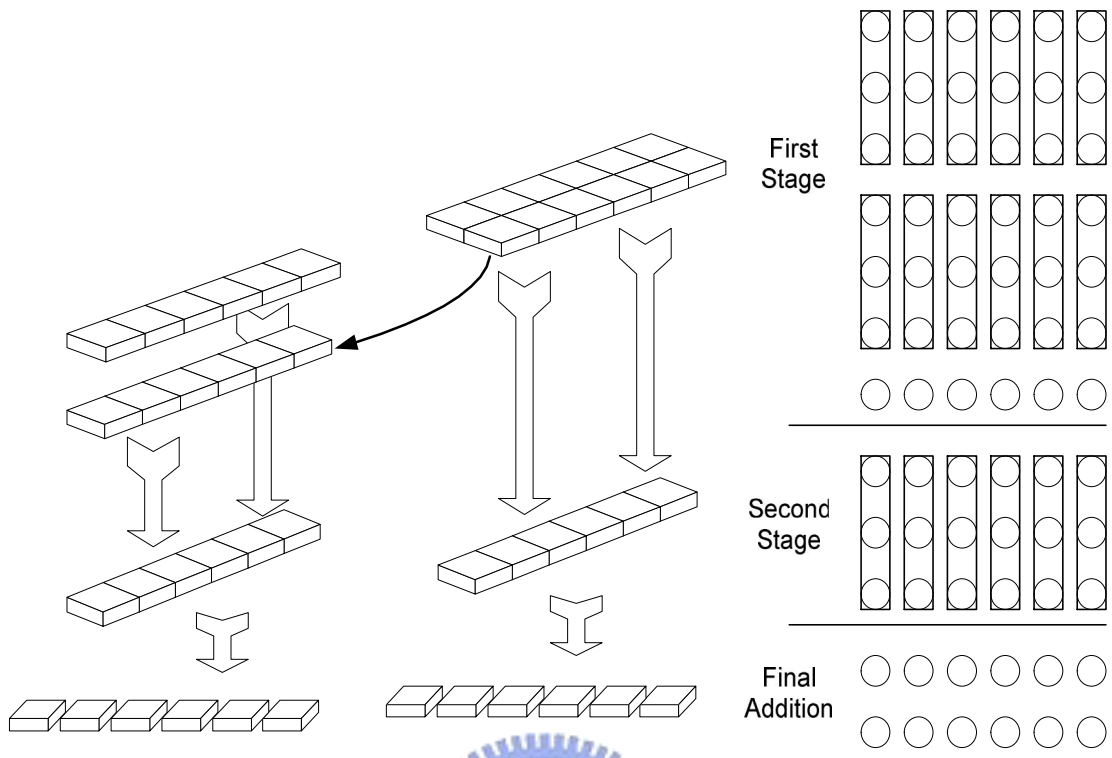


Figure 3.6 Three dimensional structural flatten into two dimensional plane



Wallace Tree 4FAs+6HAs+5-bit CPA							
	1	2	3	4	3	2	1
		(A)	A	A	A	(A)	
	2	2	2	3	2	1	1
	(A)	(A)	(A)	A	(A)		
1	2	2	2	2	1	1	1
	5-bit CPA						
1	1	1	1	1	1	1	1

Dadda Tree 2FAs+4HAs+6-bit CPA							
	1	2	3	4	3	2	1
			(A)	(A)			
	1	2	3	3	3	2	1
		(A)	A	A	(A)		
2	2	2	2	2	2	1	1
	6-bit CPA						
1	1	1	1	1	1	1	1

Figure 3.7 Wallace's strategy v.s. Dadda's strategy

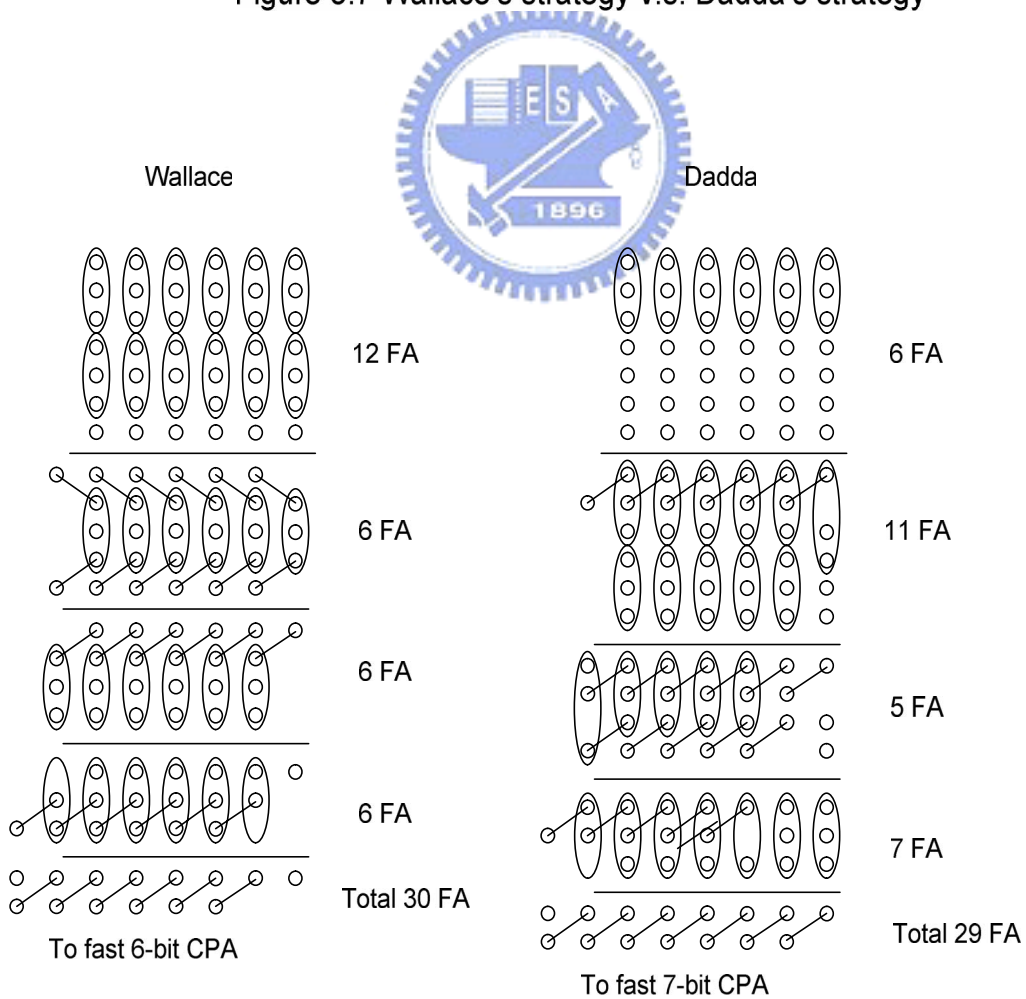


Figure 3.8 Wallace's and Dadda's strategy in 6X7 multiplication

### 3.1.4 Final Addition

When the number of partial products is reduced to two rows, which are sum and carry, a final adder is required to generate the multiplication result. The number of bits of the final adder is the sum of the number of bits of the multiplier and multiplicand. Thus, the adder width is usually doubled and the delay of this stage is more critical.

Typically, an adder is built under the assumption that all of the input signals arrive at the same time. In some cases, this is not true and a delay exists between input signals. The two rows, sum and carry, of partial products is such a case, that a non-uniform delay profile exists between different bit position. Therefore Some proposed strategies [3.11] [3.12] [3.13] will be introduced into our implementation to optimize the overall delay.

The high performance parallel adder is essential to a fast MAC. For several decades, carry lookahead adders have been the popular choice for high performance requirement. In this thesis, therefore, we would like to focus attention on these fast carry propagation adders. There are several variations of lookahead adders, such as the prefix adder [3.14] and the conditional-sum adder [3.15]. The difference between the prefix adder and the conditional-sum adder is that conditional-sum adder produce the sum bits directly with a minimum logarithmic gate depth whereas the prefix adder produce only the carry signals with equal depth and less logic [3.16]. Conditional-sum adders are thus potentially faster than prefix one, however it suffers from fan-out limitations since the number of multiplexers that need to be driven increases exponentially. Therefore, the prefix adder is preferable to our applications.

## 3.2 Booth Recoding Schemes

There has been extensive work on radix-4 Booth recoding scheme with the objectives of high speed, small area and low power. In [3.17], a comprehensive discussion and gate level comparisons are reported. In this thesis, our objective is to achieve high performance demand in micro-architecture level. Thus a parallel recoding scheme is only considered here. In parallel scheme, [3.18] categorized it into three classes: three-signal schemes, four signal schemes and five signal schemes. Table 3.2 lists some selected high speed recoding scheme from. In this sub-section, we implemented these schemes using Verilog-HDL in a 16X16+40 MAC which was implemented as a sub-module to compare fairly. These Verilog-HDL designs are optimized and mapped into TSMC 0.13 $\mu$ m standard cell library using Synopsys Design Compiler. The area, delay and power consumption will be compared based on the report of the design compiler.

Table 3.2 Selected high speed recoding scheme

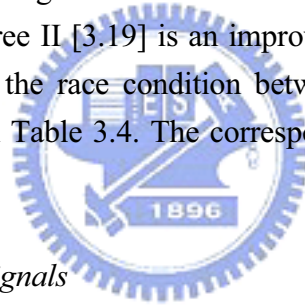
Name	Feature
Sign-select [3.3]	3 signal neg / pos / two
PR3 [3.17]	3 signal neg / two / one
Race free II [3.19] [3.2]	4 signal neg / one / two / z
Standard [3.18]	5 signal P1 / P2 / M1 / M2 / Z

### 3.2.1 Standard Scheme – Five signals

The most common Implementation of Booth encoder and PP generation, this recoding scheme is to generate five separate controls signals, P1, P2, Z, M1, and M2, corresponding to recoding digits +1, +2, 0, -1, and -2. The recoding table is shown in Table 3.3 and the logic diagram after logical operation are shown in Figure 3.9.

### 3.2.2 Race Free II Scheme – Four Signals

The second high-speed recoding schemes is a four signal schemes, which was named according to [3.2]. This Race-Free II [3.19] is an improved version from [3.2], to prevent the power consumption due to the race condition between the X and Y operands. The recoding scheme is presented in Table 3.4. The corresponding recoding and PP generator are shown in Figure 3.10.



### 3.2.3 PR3 Scheme – Three Signals

The third choice of recoding scheme is a three signal schemes [3.17]. This encoding scheme is an improved version from [3.18]. The recoding scheme is presented in Table 3.5. The corresponding recoding and PP generator are shown in Figure 3.11.

### 3.2.4 Sign-select Scheme – Three Signals

The last scheme was presented in [3.3], it use fewer transistors making it attractive in small area. However, it suffers the latency problem as the output PP<sub>j</sub> are obtained from two cascaded multiplexers. The recoding scheme is presented in Table 3.6. The corresponding recoding and PP generator are shown in Figure 3.12.

$Y_{i+1}$	$Y_i$	$Y_{i-1}$	$P_1$	$P_2$	$Z$	$M_1$	$M_2$
0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0
1	1	0	0	0	0	1	0
1	1	1	0	0	1	0	0

Table 3.3 Standard Encoding scheme

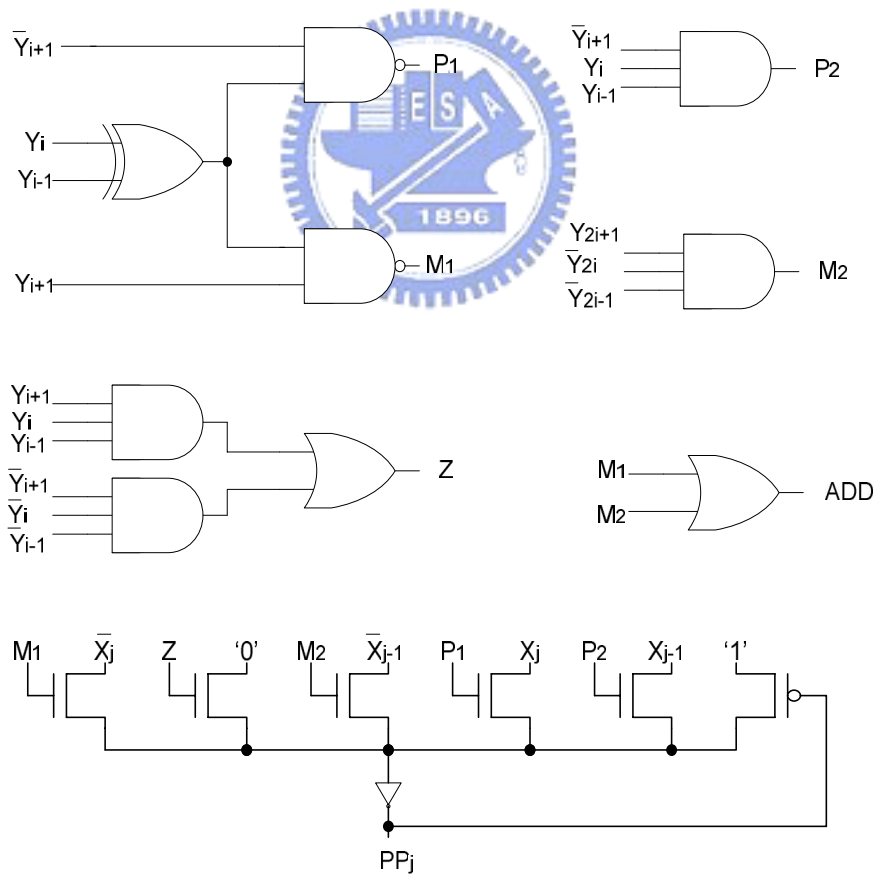


Figure 3.9 Standard encoder and decoder

Table 3.4 Race free scheme

$Y_{i+1}$	$Y_i$	$Y_{i-1}$	$X_{1b}$	$X_{2b}$	NEG	Z
0	0	0	1	0	0	1
0	0	1	0	1	0	1
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	0	0	1	0	1	0
1	0	1	0	1	1	0
1	1	0	0	1	1	1
1	1	1	1	0	1	1

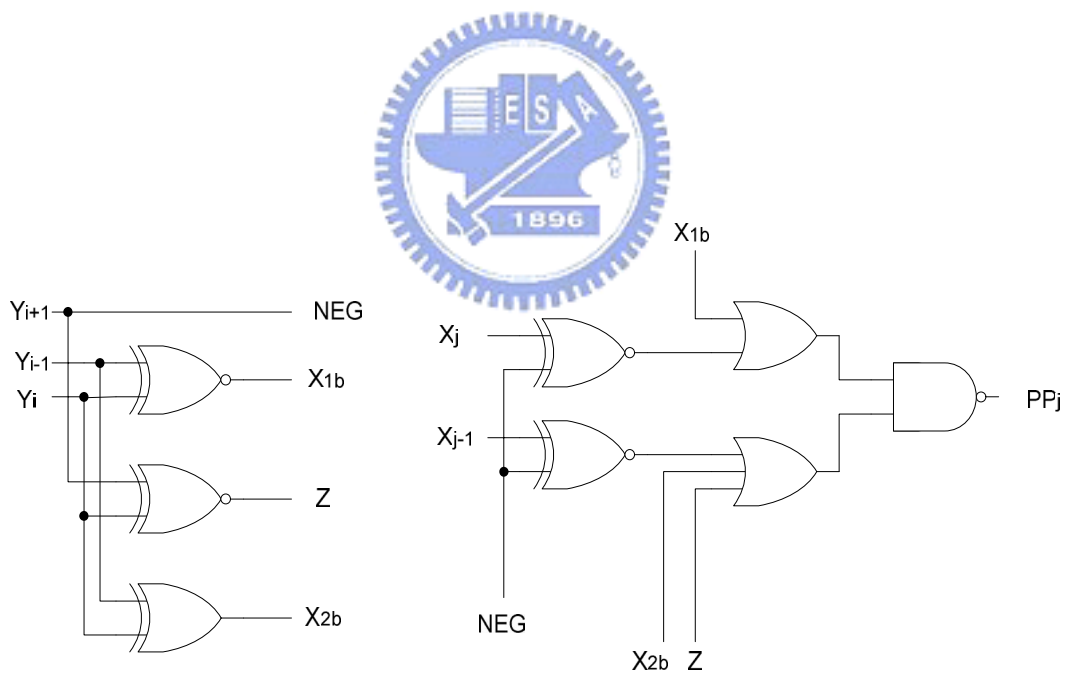


Figure. 3.10 Race free encoder and decoder



Table 3.5 PR3 encoding scheme

$Y_{i+1}$	$Y_i$	$Y_{i-1}$	one	two	neg
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	0	0

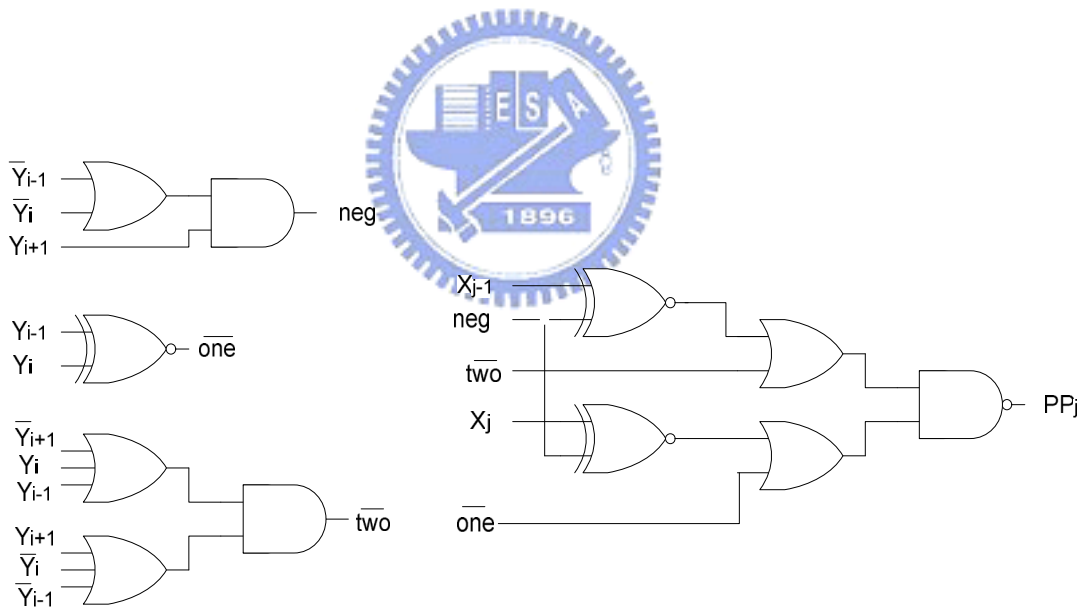


Figure 3.11 PR3 encoder and decoder

Table 3.6 Sign select scheme

$Y_{i+1}$	$Y_i$	$Y_{i-1}$	neg	two	one
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	1	0

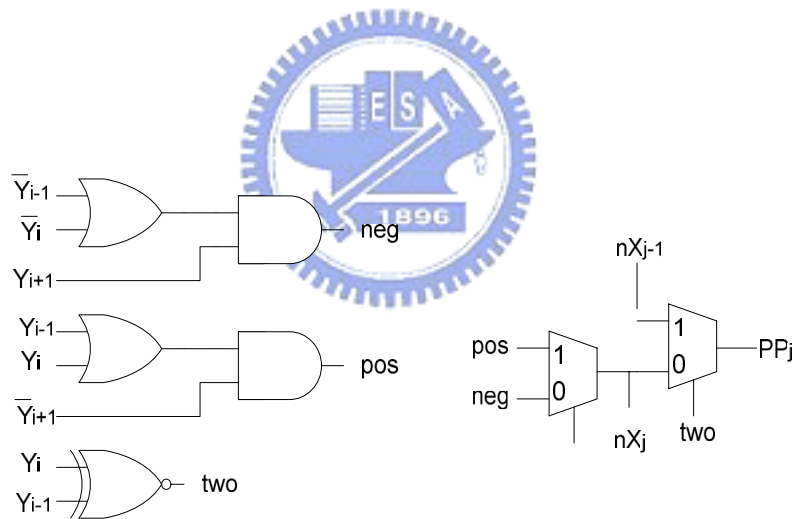


Figure 3.12 Sign- select encoder and decoder

The power / area / delay comparison results of the Booth recoding macro for 16 x 16 multipliers are reported in this final section. We have implemented and simulated four existing parallel recoding schemes, all these schemes are described with Verilog HDL in gate level. Each encoding scheme will be served as independent sub-module, as depict in Figure 3.13, to optimized and mapped into TSMC 0.13 $\mu$ m generic cell library using Synopsys Design Compiler.

The synthesized results without given any design constrain are listed in Table 3.7, while the synthesized results with minimum (tight) input/output delay constrain are listed in Table 3.8.

The results which without any design constrained reflects that more nature and original circuit structure, in terms of such circuit structure one can know that whether it is more suitable for circuit – level implementation. From this point of view, the standard encoding scheme are suitable for circuit level implementation because of the small delay in it's original circuit structure. The results with timing constrains reflects which recoding scheme is superior to others. As Table 3.9 shows, the Race-free recoding scheme is the fastest one but it consumes largest power. While the speed and power consumption of standard recoding scheme presents a moderate level of power consumption and delay .

Overall, from cell based implementation, standard recoding scheme is a good choice for 16X16 multiplier in terms of power and delay. Note that this conclusion is valid when a similar standard cell library is used.

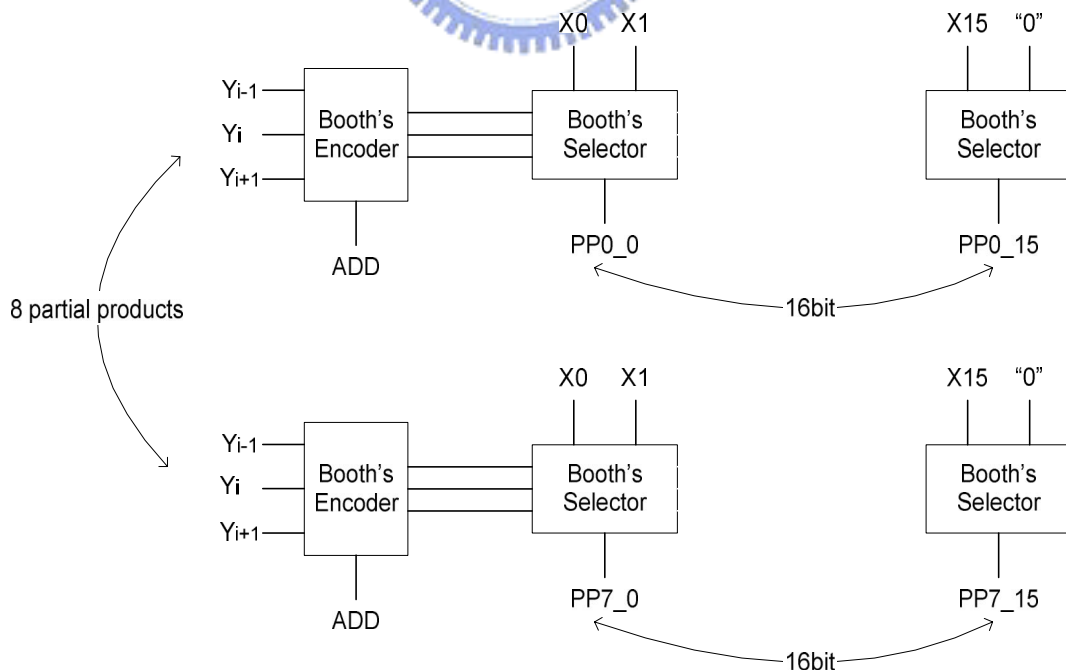


Figure 3.13 Booth Recoding Macro

Table 3.7 Comparison of Booth Recoder without any synthesized constrains

Schemes	Area ( $\mu\text{m}^2$ )	Power (mW)	Delay (ps)
Standard	4725	2.70	630
Race-free	3612	4.06	930
PR3	3313	3.55	720
Sign-select	4087	4.12	1260

Table 3.8 Comparison of Booth Recoder with 100ps timing constrains

Schemes	Area ( $\mu\text{m}^2$ )	Power (mW)	Delay (ps)
Standard	10050	8.92	260
Race-free	11464	16.9	230
PR3	9929	12.15	280
Sign-select	10145	10.24	320

### 3.3 Partial Product Matrix Topologies

The partial product matrix topologies, have been discussed in section 3.1.3, were classified as either array topology or tree topology. The tree topology have the smallest logic delay proportional to  $\log(n)$ . However, they have irregular layout with complicated interconnects. On the other hand, array topology have large delay but offer regular layout and simpler interconnect.

In this section we will implement several 2's complement  $16 \times 16 + 32$ ,  $16 \times 16 + 40$  multiplier-accumulator with the regular array topology as well as irregular tree topology [2.2] [2.3]. The detailed architecture of each topology will be discussed briefly. To further speed up, Three-Dimensional reduction Method (TDM) [3.11] was employed in tree topology. The TDM algorithm finds optimal interconnection of partial product reduction tree by carefully connecting the delay paths of compressors. As a result of the effectiveness of column compressor [2.3], the tree topology of partial product matrix constructed by using TDM algorithm outperforms the conventional designs.

#### 3.3.1 Carry Save Adder (CSA) Array Topology

Figure 3.14 shows the architecture of  $16 \times 16 + 32$  carry save adder array MAC operation. Eight rows, the 1<sup>st</sup> row ~ the 8<sup>th</sup> row, representing the summands PP0~PP7, are generated by the Booth recoders. These summands, which connected in carry save form are added using 1 bit full adder as shown by the empty circles. In order to prevent overflow, sign

extension have to be provided as shown by the black circles. In traditional topology, the sign extension have to be provided in full width of datapath, the 2<sup>nd</sup> row has to be extended by two bits, the 3<sup>rd</sup> row by 4 bits and the 4<sup>th</sup> row by 6 bits and so on. Evidently, this is extremely wasteful and does not result in regular topology. In this technique, only one sign extension bit needs to be provided in each row will lead to very regular topology. The dashed arrows on the rightmost column represents add one at the LSB for the two's complement operation of Booth recoder, and the dashed arrows on the leftmost represents sign extension process. The diamonds at the rightmost row and the bottom row represent the final stage fast adder that are required to produce the final result of MAC operation.

Generally, the long accumulator is a major bottleneck of MAC operation. Since add operation is commutative, we can introduce the accumulated value in to partial product reduction array results in a design which multiply and accumulate operations are merged. In Figure 3.14 shows, the accumulated value are introduced to partial product reduction array in two place. Z0~Z15 are introduced at the 1<sup>st</sup> row , while Z16~Z31 are introduced the 9<sup>th</sup> row. Notice that Z16 can not be introduced into 1<sup>st</sup> row, since the MSB of partial product which generated by Booth recoder is a sign extension.

The carry save adder array of 16X16+32 MAC will be served as independent module without fast final adder, to optimized and mapped into TSMC 0.13 $\mu$ m generic cell library using Synopsys Design Compiler. Table 3.9 shows the synthesized results with no timing constrain as well as with 0.5ns timing constrain.

Table 3.9 Synthesized result of CSA array with and without timing constrains

Timing constrain	Area ( $\mu\text{m}^2$ )	Power (mW)	Delay (ns)
No	5500	7	3.2
1ns	18396	29	1.7

### 3.3.2 Column Compression Tree (CCT) Topology

Previously, 1-bit full adder is the primitive cell for whole CSA array, while in the column compression tree topology the primitive cell is the compressors. The 1-bit full adder can be a [3:2] compressor or (3,2) counter to build tree topology, results in Wallace's or Dadda's tree multiplier. However, the multipliers using [3:2] compressor or (3,2) counter needs more stages and inefficiently. In order to further improve the speed of partial product reduction step, higher order compressor or counter needs to be provided [3.20].

In this thesis, [2:2] compressor (half adder), [3:2] compressor or counter(full adder), [4:2] compressor and [5:2] compressor have been the basic elements of CCT topology of 16X16+32 MAC. Figure 3.15 depicts logical decomposition of [4:2] and [5:2] compressor [29]. The [4:2] compressor has 5 input bits including a carry-in from the neighboring cell

and the [5:2] compressor has 7 input bits including two carry-in. They have 3XOR and 4XOR gate delay, respectively. In full custom design, XORs and multiplexers (MUX) can be designed efficiently via aggressive transistor sizing or other circuit families (will be discussed in chapter4).

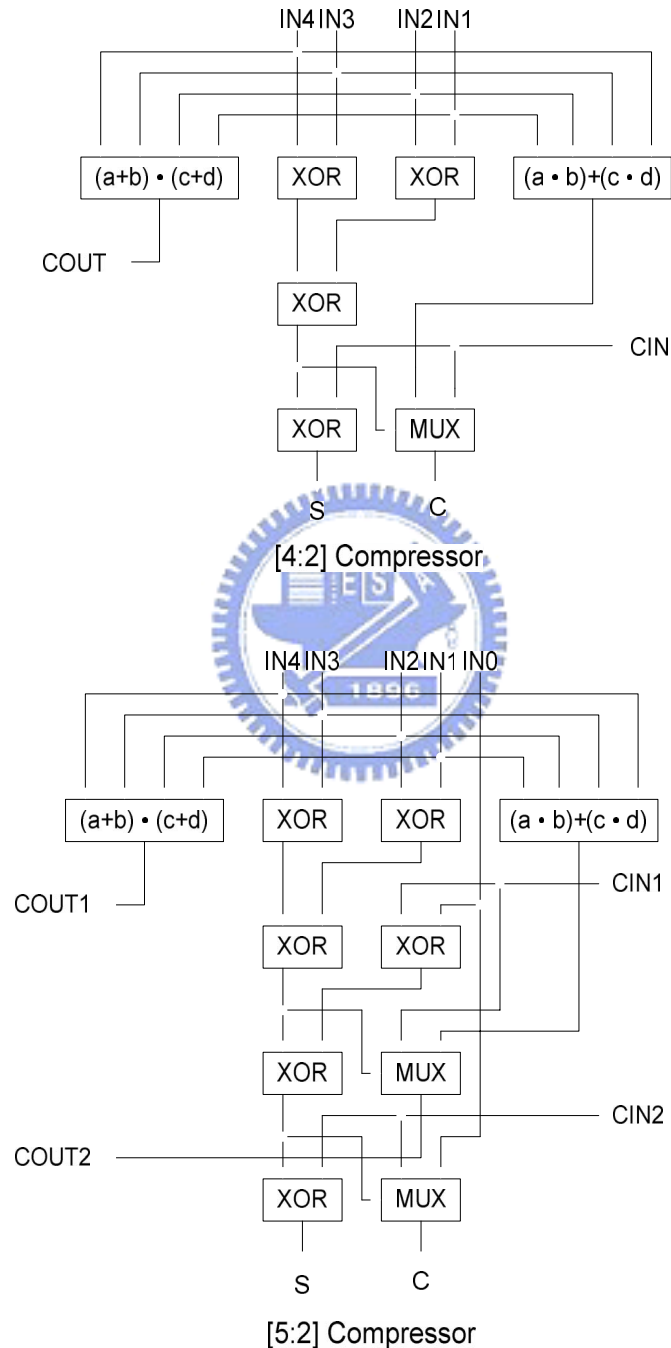


Figure 3.15 Logical decomposition of [4:2] and [5:2]compressor

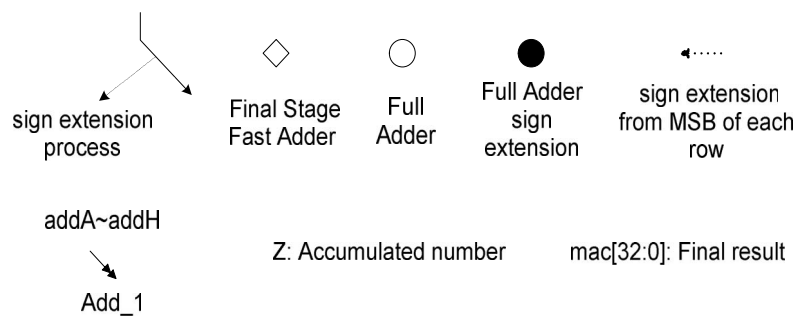
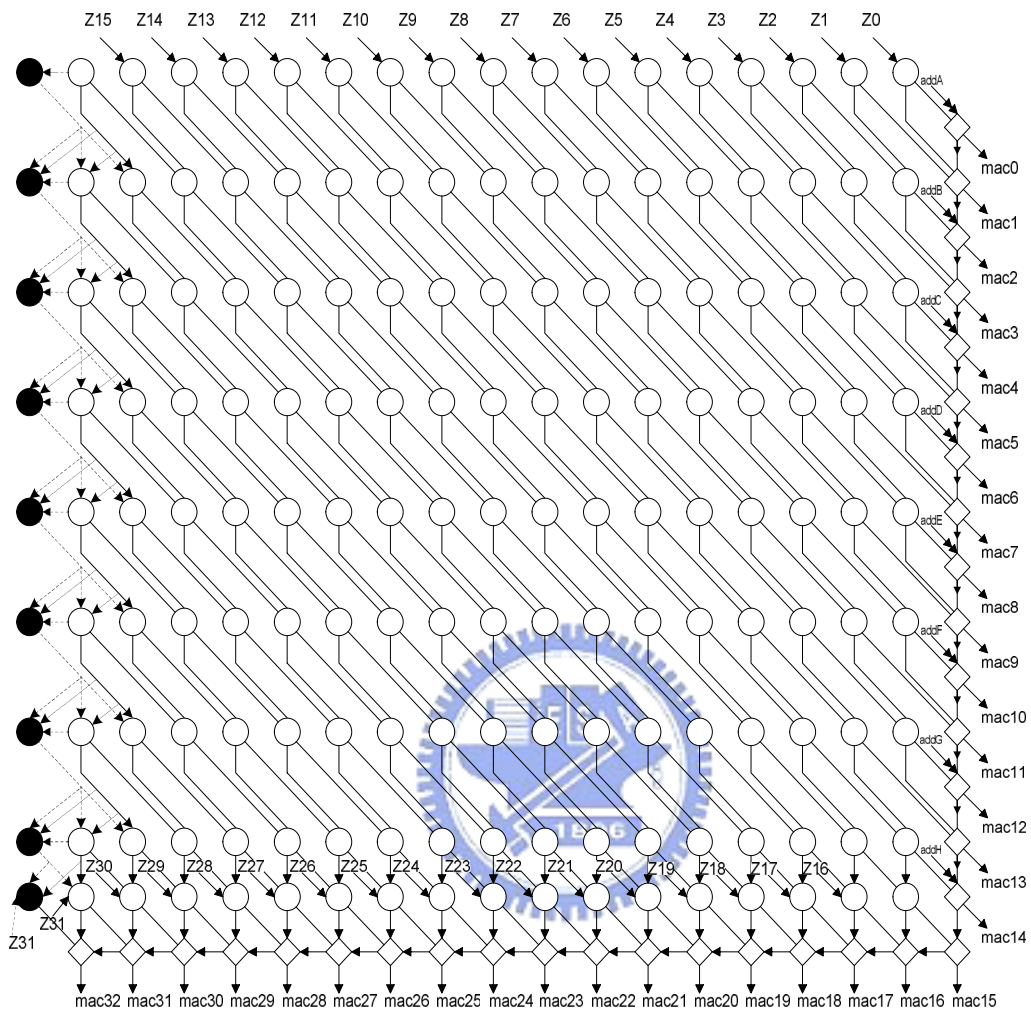


Figure 3.14 16 X 16 + 32 MAC using carry save adder array

Figure 3.16 shows the structure of merged 16X16+32 MAC operation. In the figure, the sign bit of partial products are shown as empty circle with label “S”, the grey circle “Add\_1” represents add one at the LSB for the two’s complement operation of Booth recoder and the black circle represents the accumulated number Z. A bar above a circle means the logical inverse of that circle. The mathematical proof of the sign extension derivation is given by [3.21].

The architecture of Figure 3.16 has a little irregular structure because of the additional partial product term at the LSB of each partial product. To compensate this irregular topology for doing CCT reduction smoothly, the LSB of each partial product and “Add\_1” are combined and further simplified using logic operation. The resulting equations for the new LSB and new Add\_1 can be written as (2) (3), respectively. Figure 3.17 shows the modified partial product array. The additional partial term is now moved to next higher order bit position, and LSB-part array become more regular.

$$\text{NEW\_PP}_{\text{LSB}} = \text{Original\_PP}_{\text{LSB}} \cdot (B[0] \oplus B[1]) \quad (2)$$

$$\begin{aligned} \text{NEW\_Add\_1} = & (\text{Original\_PP}_{\text{LSB}} \cdot B[2] \cdot B[0]') + \\ & (\text{Original\_PP}_{\text{LSB}} \cdot B[2] \cdot B[1]') + \\ & (B[2] \cdot B[1] \cdot B[0]') \end{aligned} \quad (3)$$

where B is the multiplier of MAC operation.

The maximum height of bit slice is 10, consists of 8 partial products, 1 Add\_1, and 1 accumulated number. Figure 3.18 shows the [10:2] reduction bit slice obtained by using two [5:2] compressors and one [4:2] compressor. In the first stage, two [5:2] compressors will act concurrently then in second stage one [4:2] compressor adds sums from first stage. Therefore, the critical path only takes 7 XOR gate delay, which results a high speed partial product reduction step. The overall reduction process of a 16X16+40 MAC operation is depicted in Figure 3.19.



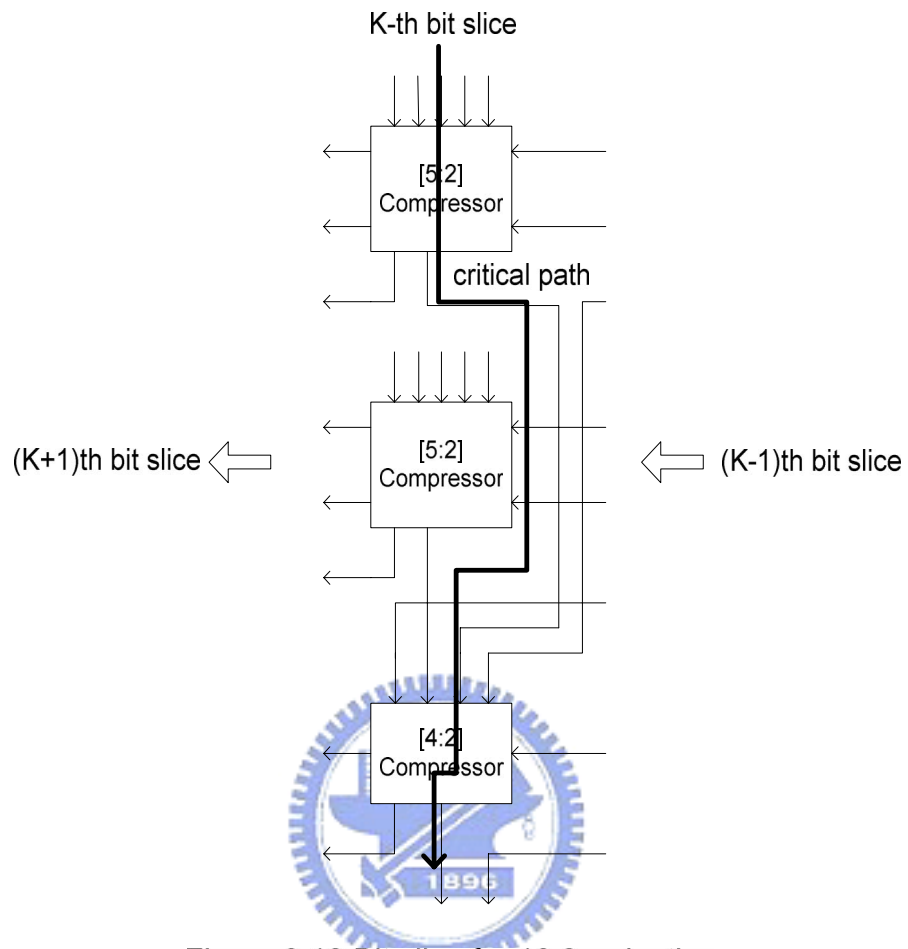


Figure 3.18 Bit slice for 10:2 reduction

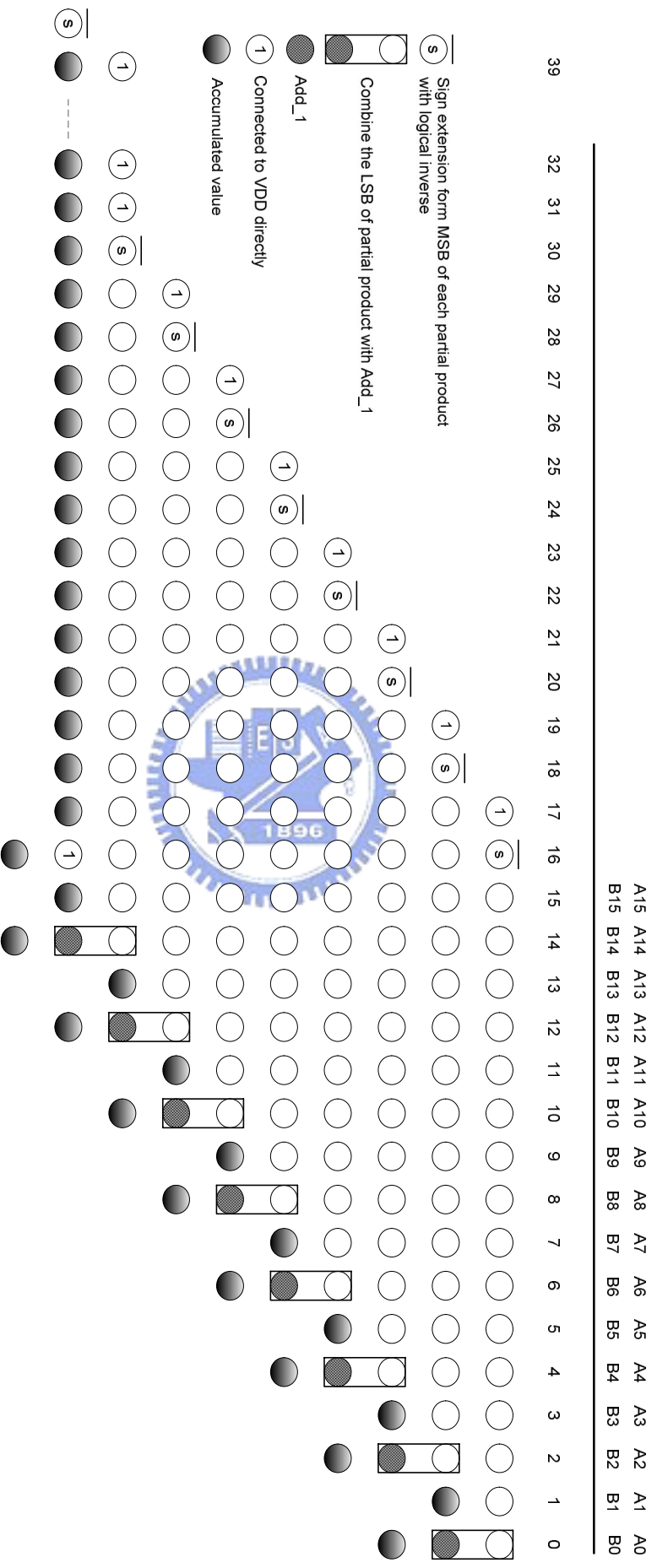


Figure 3.16 Dot diagram of a 16X16 2's complement MAC with arbitrary accumulator

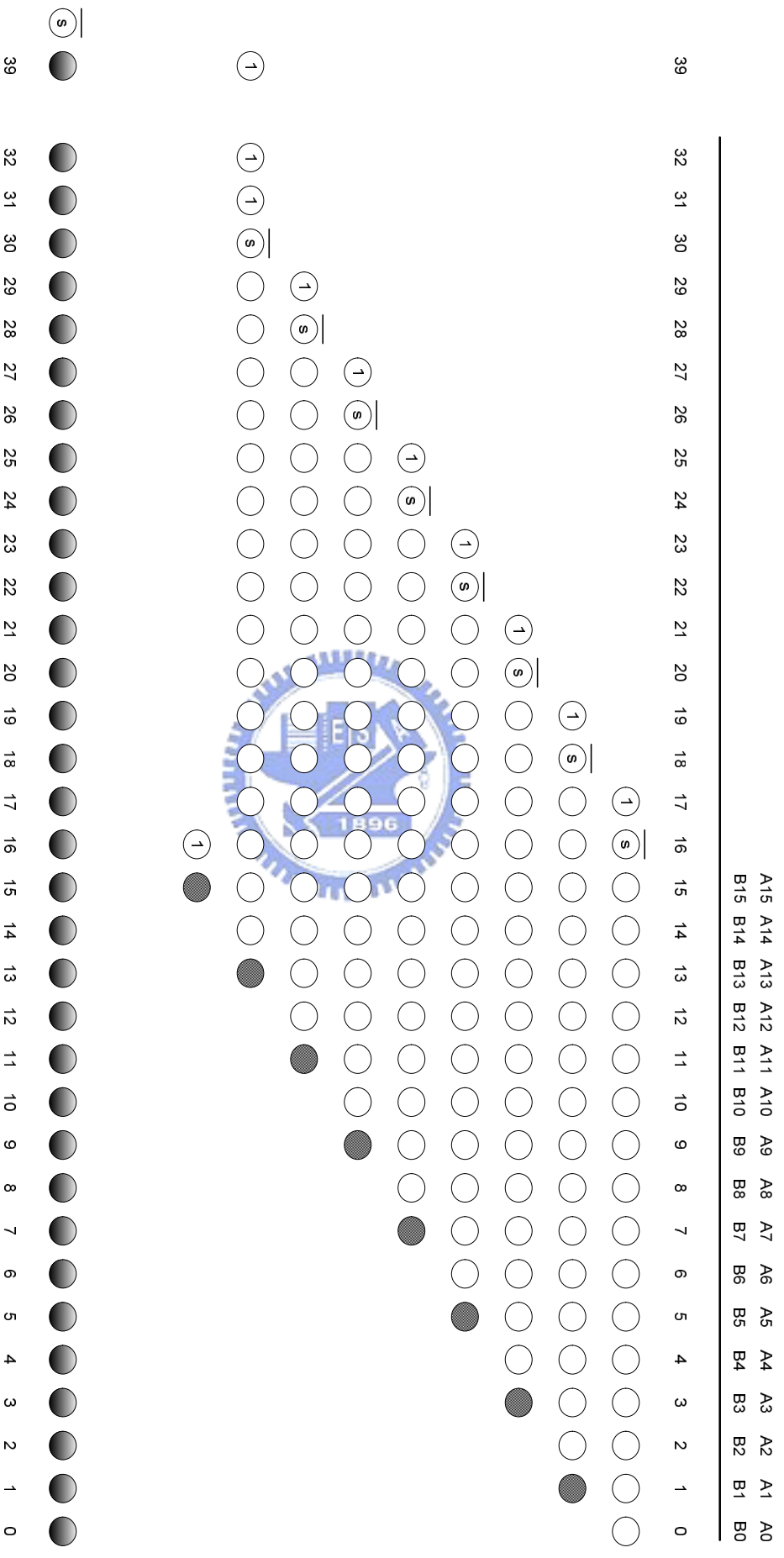


Figure 3.17 A modified 16X16 2's complement MAC with arbitrary accumulator

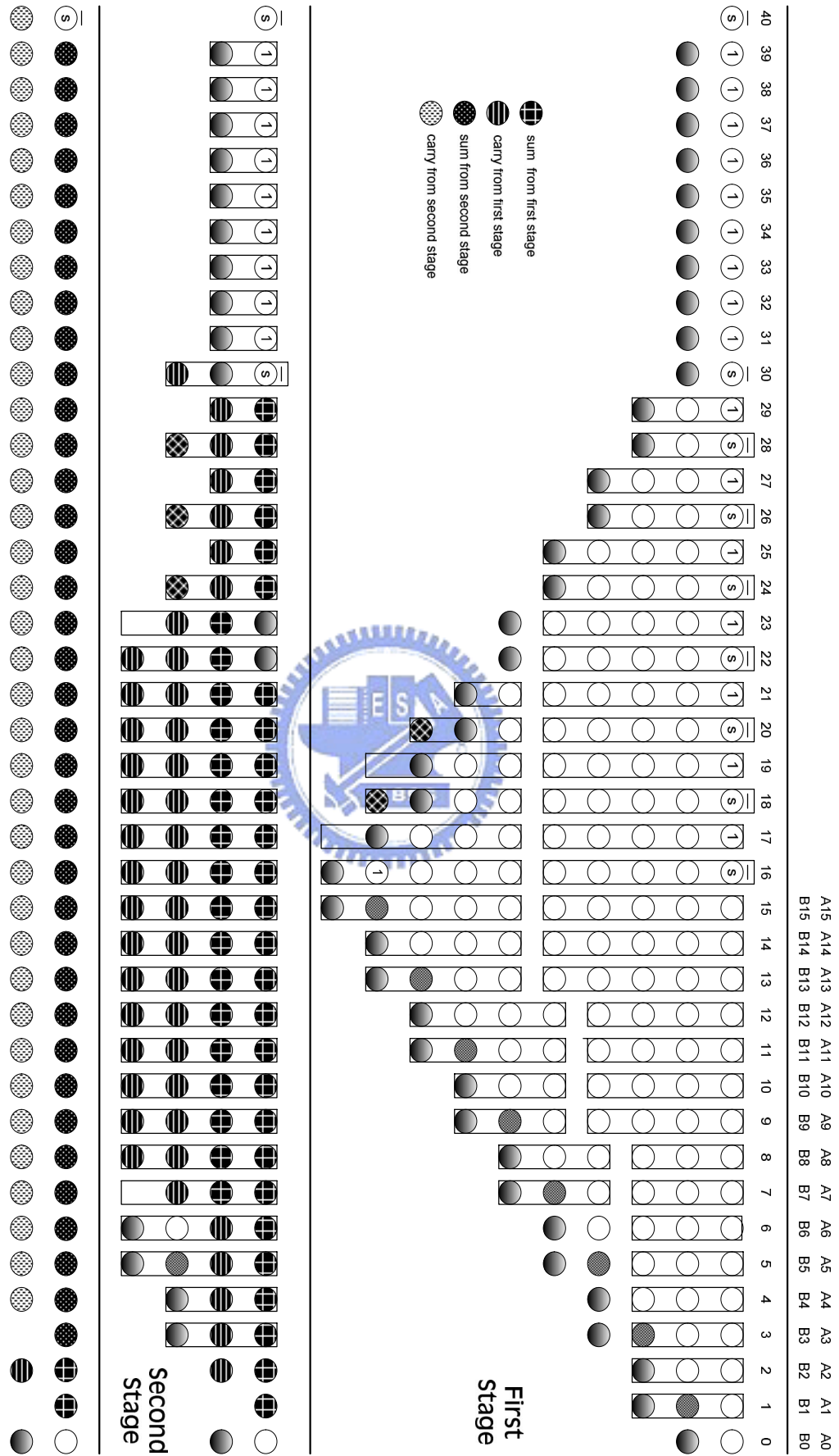


Figure 3.19 The reduction process of 16X16+40 MAC using column compression tree topology

### 3.4 High-Performance Parallel Adder

The overall MAC speed can be further improved via the high-performance parallel adder to add the sum and carry, which produced by partial production step. For the economize the use of high performance adder, which represents high power consumption and high complexity of hardware, without affect overall MAC speed, we should estimate the arrival profile of the two rows of partial product reduction accurately.

For achieving high speed MAC operation. The use of high performance parallel adder is unenviable. Therefore, there are some types of high performance parallel adder will be discussed, implemented, and compared in this thesis.

#### 3.4.1 Design Strategies

In order to obtain the arrival profile of each topology (CSA and CCT) accurately, we use a static timing analyzer (Synopsys Primetime) linked with TSMC 0.13 $\mu$ m generic cell library to analyze the signal arrival time of two rows of partial product matrix topologies (CSA and CCT), without final adder.

Figure 3.20 shows the arrival time profile of CSA array topology at each bit position. As Figure 2.20 shows, the arrival time of bit-position 0 ~ bit-position 15 increased gradually while the arrival time of bit-position 16 ~ bit-position 31 kept constant. Hence, we optimize the 32-bit final addition in two part. The first part, bit-position 0 ~ bit-position 15, we use a 16-bit linear-time carry lookahead adder [2.15] to fit the signal which arrival linearly. The second part, bit-position 16 ~ bit-position 31, a 16-bit high performance adder (such as prefix adder or conditional carry adder) will be employed.

Figure 3.21 shows the arrival time profile of CCT topology at each bit position. According to this profile, we optimize the 32-bit addition into short 8-bit addition and 24-bit addition. For the short 8-bit adder, the ripple carry adder (RCA) or two cascaded 4-bit carry lookahead adder is a suitable choice. On the other hand, a high performance adder would be a desired choice for remainder addition.

#### 3.4.2 Several High Performance Adders

Conditional carry adders and prefix adders, both have logarithmic gate depth, are high performance parallel adders. Hence, there are two types of parallel prefix adder and one conditional carry adder will be implemented in this thesis for high speed MAC operation. In prefix adder, one is 32-bit radix-2 Kogge-Stone adder [3.24], the other is 32bit Han-Carlson adder [3.26]. The interconnection structure of K-S adder is regular, which ease to implement. Moreover, as Figure 3.22 shows the fan-out of the dot operator is uniformly distributed, especially on the critical path. However, the dense wire connection between each dot operator results in large area overhead and power consumption.

Therefore, we trade-off some delay for area and power by choosing sparse tree adder, the Han-Carlson adder [3.26]. For Han-Carlson, a hybrid of Ladner-Fischer [3.25] and Kogge-Stone, which computes the carry every two bits, as illustrated in Figure 3.23. The high speed conditional carry adder [3.27] which is also implemented in gate level is a competitive design, Figure 3.24-1 and Figure 3.24-2 shows its architecture.

Table 3-12 and Table 3-13 shows the comparisons of synthesized results. As Table 3.21 shows, if the performance is a major concern the K-S adder is a preferable one. On the other hand, however, if we considered the power or area, the H-C adder is an appropriate choice.

Table 3.11 Synthesized results of 32-bit adder without timing constrain

Without timing constrain	Area ( $\mu\text{m}^2$ )	Power (mW)	Delay (ns)
K-S adder	2963	2.1	1.21
H-C adder	2113	1.7	1.3
C-C adder	2946	2.7	1.51

Table 3.12 Synthesized results of 32-bit adder with 0.5ns timing constrain

0.5 ns timing constrain	Area ( $\mu\text{m}^2$ )	Power (mW)	Delay (ns)
K-S adder	9547	10.8	0.58
H-C adder	6570	7.54	0.7
C-C adder	6886	6.9	0.86

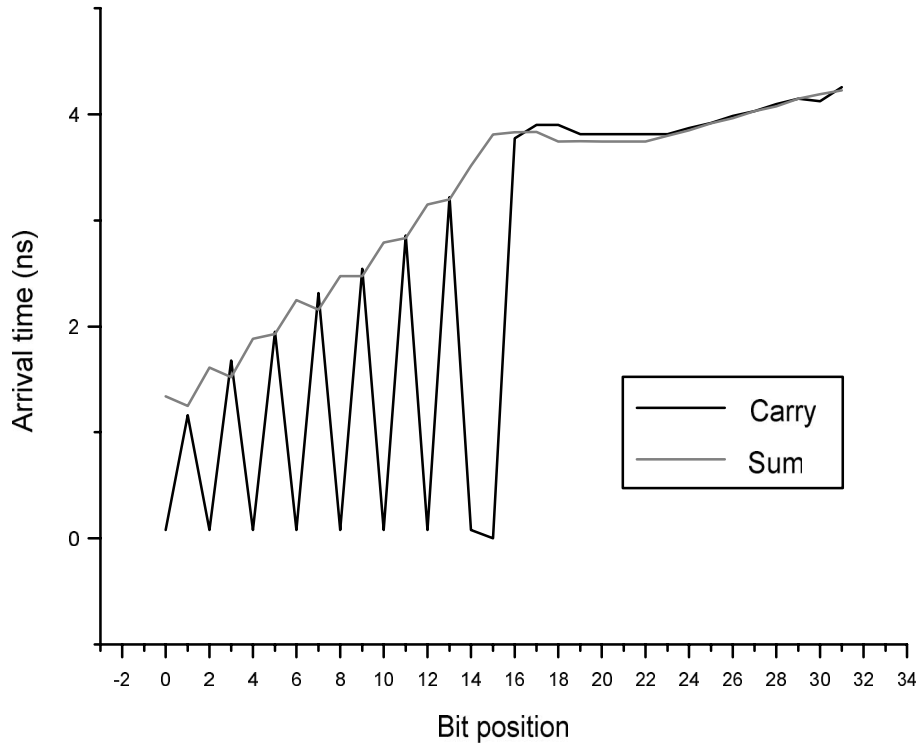


Figure 3.20 Signal arrival profile of CSA array topology

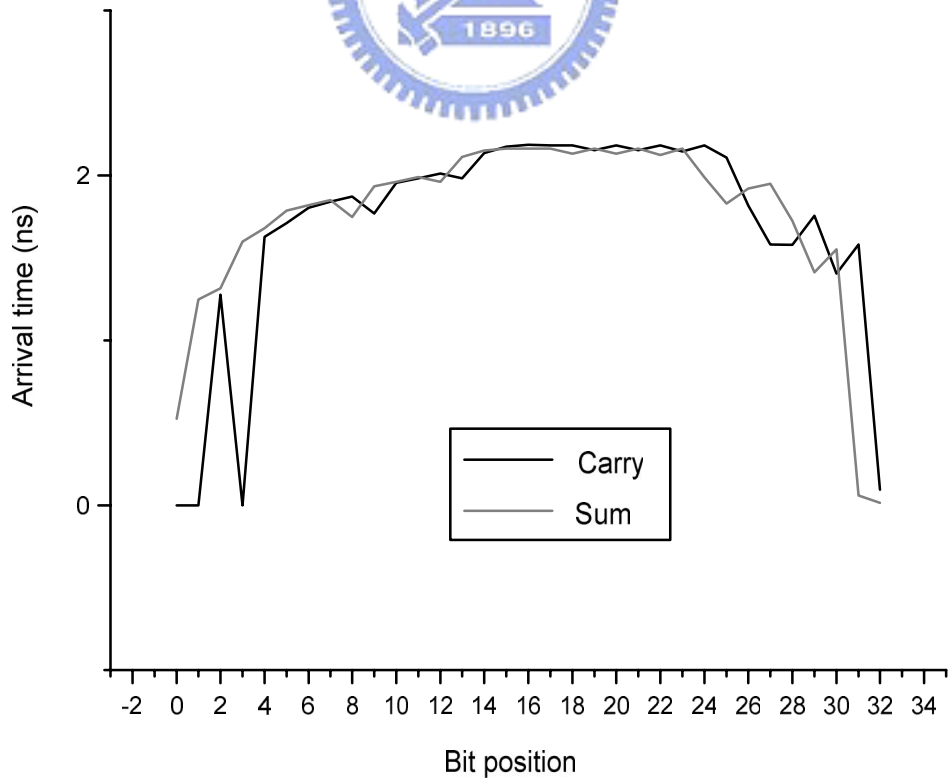


Figure 3.21 Signal arrival profile of CCT topology

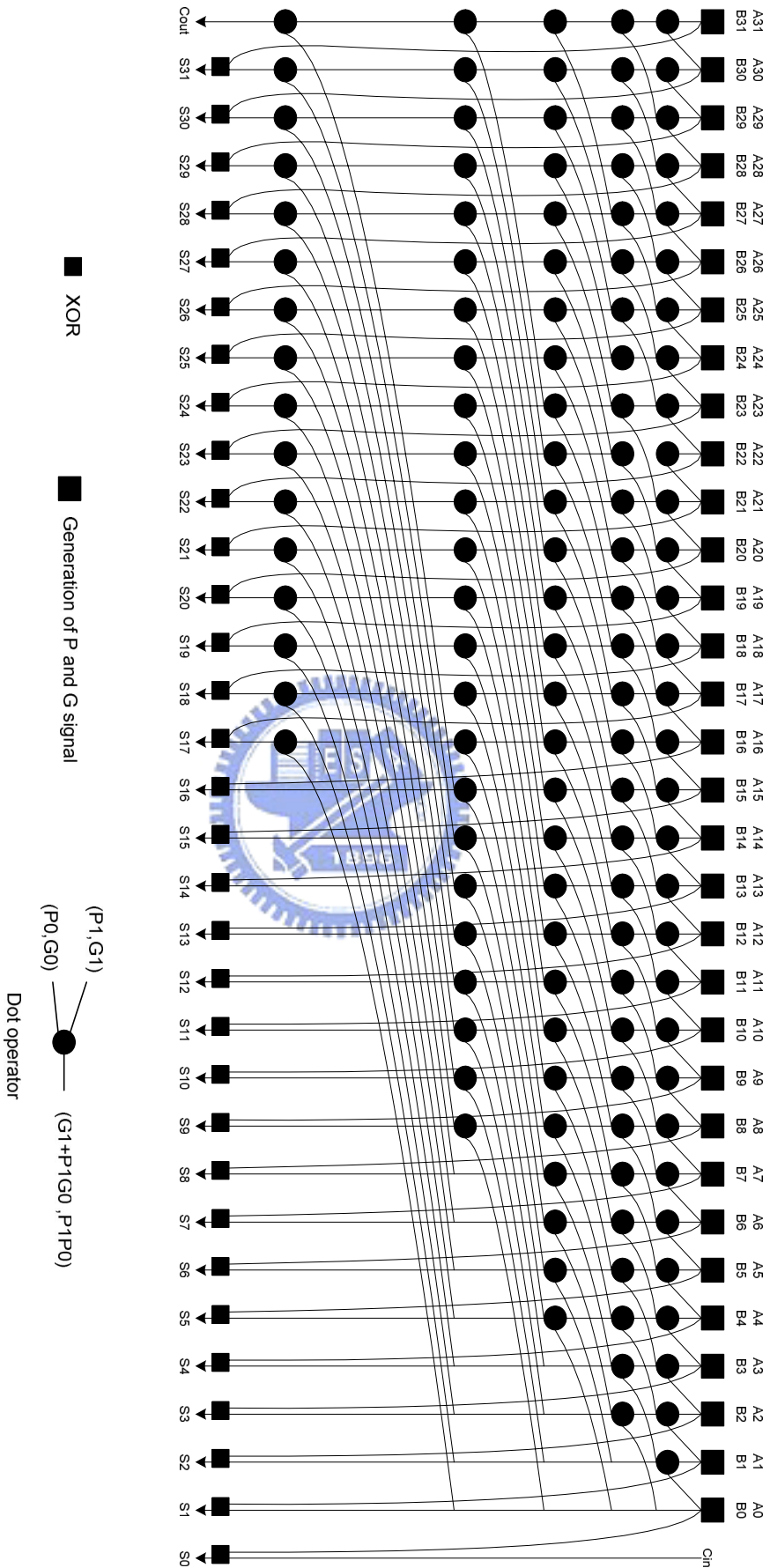


Figure 3.22 32-bit Kogge-Stone prefix adder



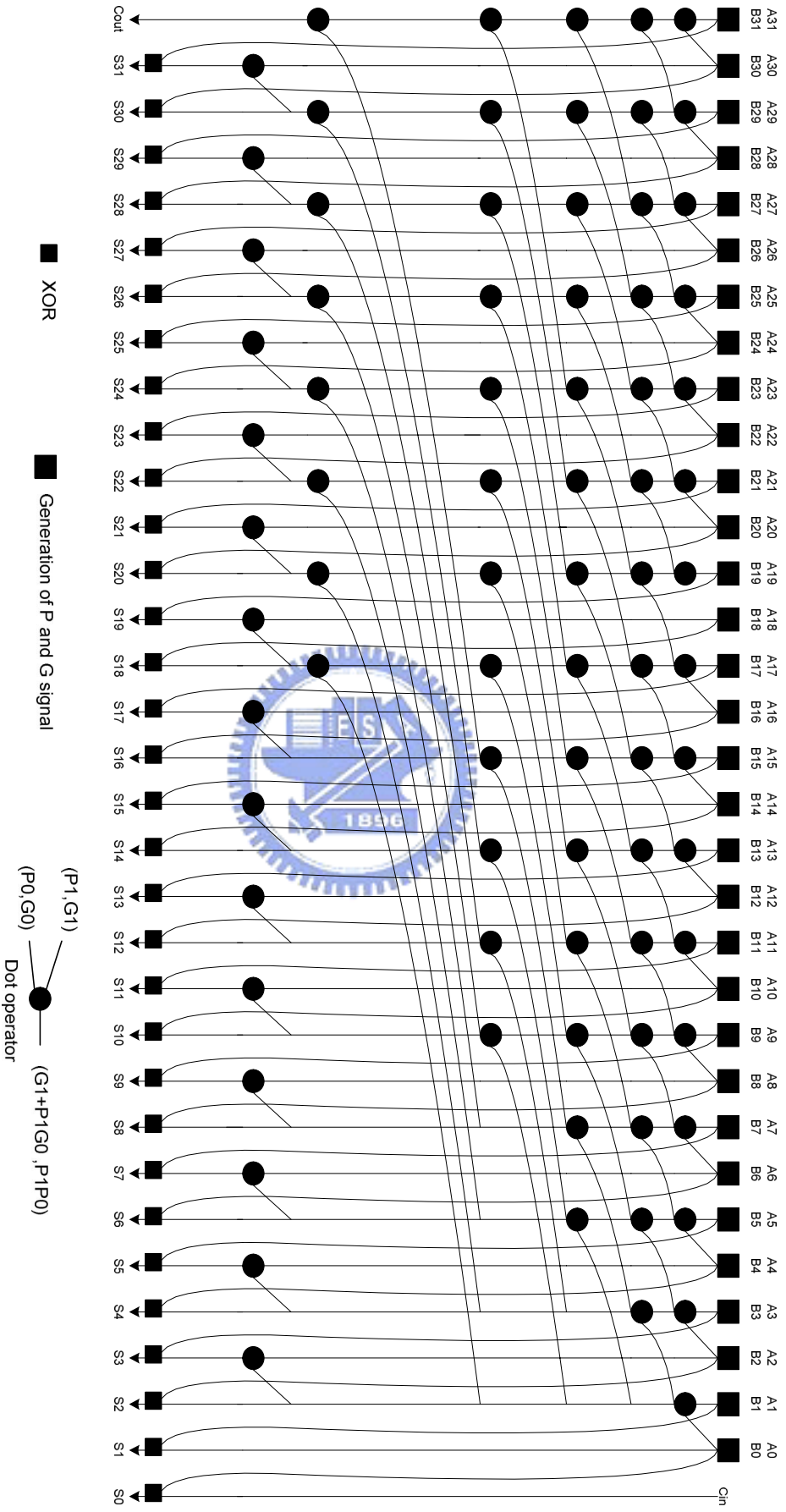


Figure 3.23 32-bit Han-Carlson prefix adder

Figure 3.24-1 32-bit Conditional Carry Adder

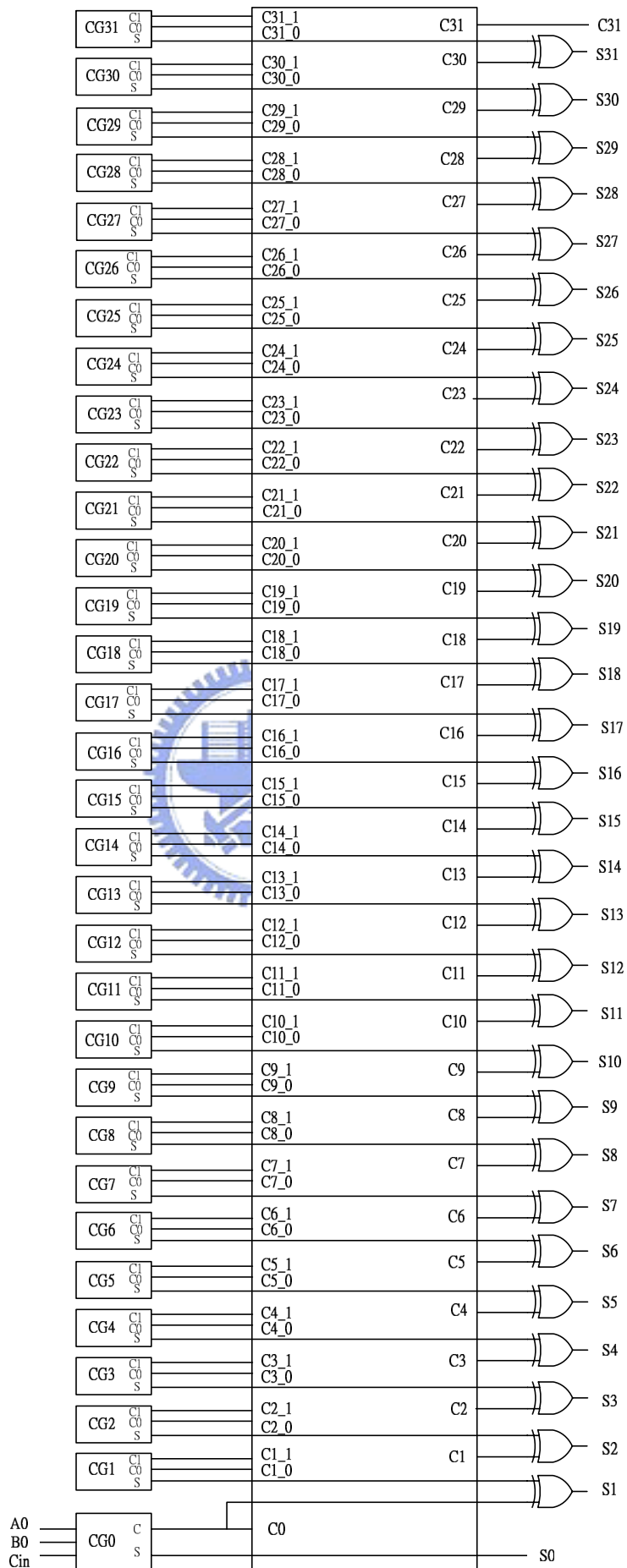
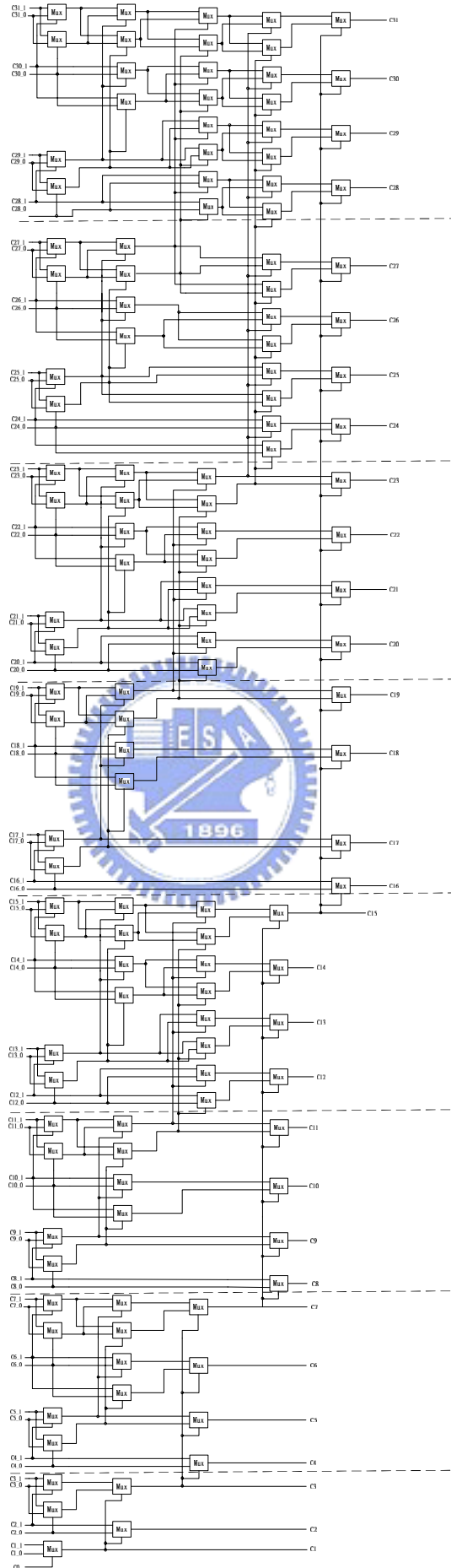


Figure 3.24-2 32-bit Conditional Carry Adder



### 3.5 Synthesized Results

We have implemented all the necessitated module of MAC previously. Accordingly, we have to combine them to form a high speed MAC macro. We will use the fastest Booth encoder, race free II, at the first stage of MAC operation. In the second stage, both the CSA array topology and CCT topology will be included. In the final stage, several different length and types of adders will be involved to trade-off delay for area and power consumption.

In order to evaluate previously implemented MAC micro-architecture, the Booth recoder, partial product topologies as well as final propagation adder, we have introduced a optimized 16X16+40 MAC from Synopsys Design Ware libraries to compare with our design. Figure 3.25 depicts the sketch of it's framework. In the first step, a 16x16 multiplier will generate multiplication result in redundant form 40-bits sum and 40-bits carry. In the second step, a 40-bit carry save adder will add these 40-bit sum, 40-bit carry as well as the accumulated number Z. Finally, a 40 bit optimized Brent-Kung [3.22] prefix adder will complete the MAC operation.

Table 3-14 and 3-15 shows several synthesized 16X16+32 MAC macros, the CCT topology always outperform CAS array topology whatever in the area, power consumption, and delay. In CCT topologies, if we use the H-C adder as the final carry propagation adder, resulting in 12% lower area and 12.7% reduction in power consumption compared with using K-S adder. However, about 7.8% speed will be sacrificed.

Furthermore, the longer accumulator's MAC macro (16X16+40) will be implemented and synthesized to compare with the MAC macro which was developed by Synopsys Design Ware Foundation R&D team. Table 3-16 and 3-17 shows the synthesized results. As Table 3-17 shows, after optimization with tight timing constrain, the power consumption of each macro is quite close. If we use two 4-bit carry lookahead adder at the first 8-bit addition, the performance will be the best. Although the performance is outperform Design Ware in 13%, the area will larger than Design Ware in 32%.

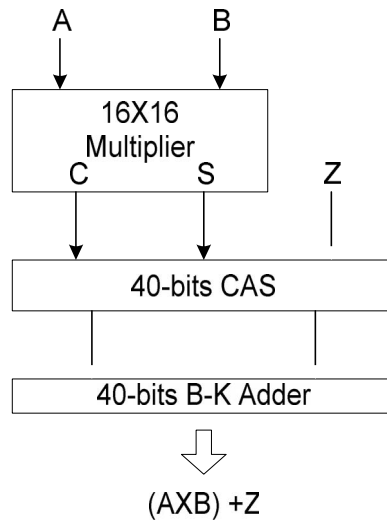


Figure 3.25 A 16X16+40 MAC from Synopsys Design ware libraries



Table 3.15 Synthesized result of 16X16+40 MAC macro without timing constrain

16X16+40	Area ( $\mu m^2$ )	Power (mW)	Delay (ns)
CCT 32 K-S adder + 8 RCA	12813	14.9	4
CCT 32 K-S adder + 4CLA*2	12922	14.9	3.41
CCT 32 K-S adder + 8 K-S	13061	15	3.5
Design Ware Libraries	109389	16.7	10.7

Table 3.16 Synthesized result of 16X16+40 MAC macro with 1ns timing constrain

16X16+40	Area ( $\mu m^2$ )	Power (mW)	Delay (ns)
CCT 32 K-S adder + 8 RCA	45794	70	1.71
CCT 32 K-S adder + 4CLA*2	44621	69.5	1.69
CCT 32 K-S adder + 8 K-S	43567	67	1.81
Design Ware Libraries	29596	68.9	1.92

### 3.6 Conclusions

This chapter has implemented various MAC micro-architecture for achieving high performance. The parallel adder were compared, as Figure 3.26 shows, the H-C adder is preferable in terms of power and area without sacrificing speed too much. As Figure 3.27 shows, the use of CCT topology which would be more difficult in wiring complexity, however, it provides the best performance in our simulation. For long accumulator MAC design, as shown in Figure 3.28,our CCT topology is faster than Design Ware, but the area is need to be improved aggressively.

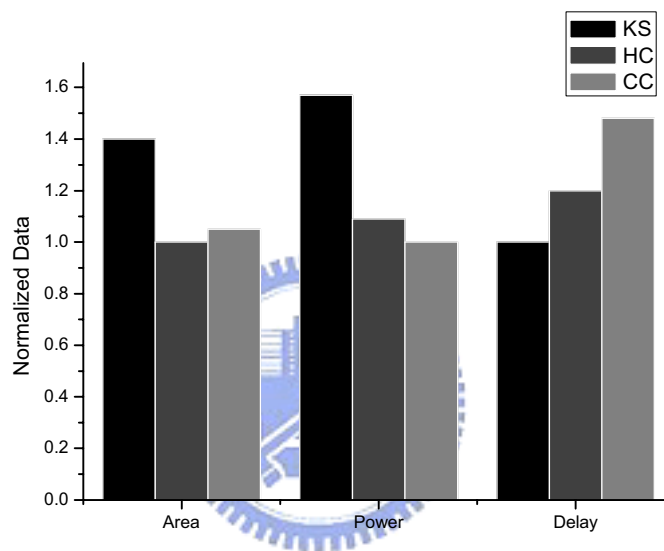


Figure 3.26 Normalized result from Table 3.12

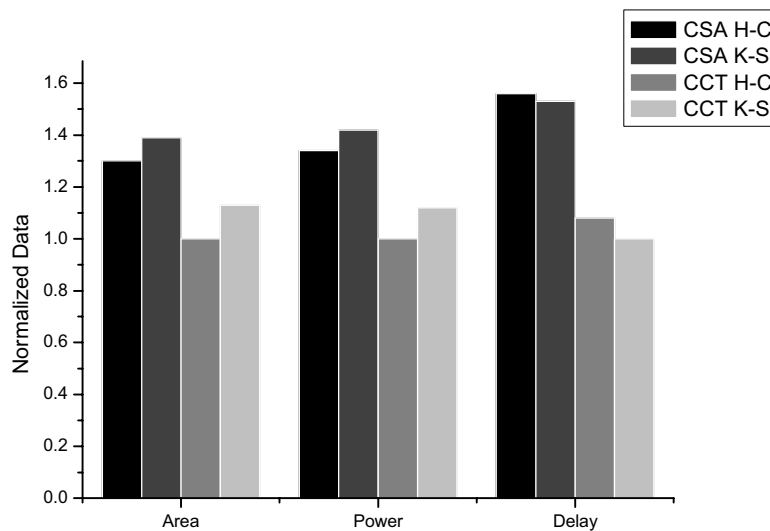


Figure 3.27 Normalized result from Table 3.14

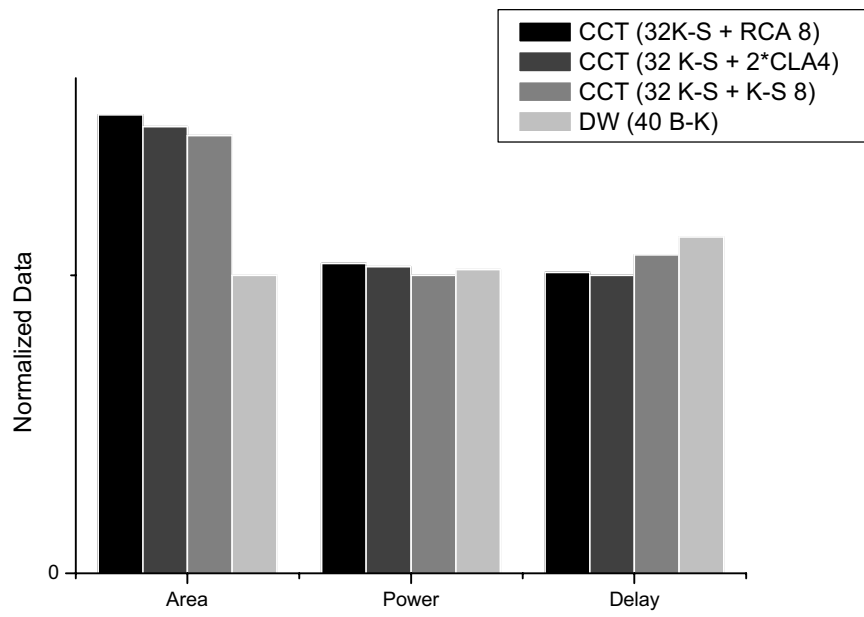


Figure 3.28 Normalized result from Table 3.16

