AN ABSTRACT OF THE THESIS OF

Che-Jen Chang for the degree of Master of Science in Electrical and Computer Engineering presented on June 5, 1997. Title: The Low-Power Design of Prefix Adder.

Abstract approved:

Redacted for Privacy

Shih-lien Lu

Minimizing the dynamic power consumption of a circuit is becoming a more and more important issue for digital circuit design in the age of portable electronics. Among all the arithmetic circuits, addition is the most fundamental operation. Therefore, designing low power adder is an important and necessary research area.

In this thesis, the dynamic switching power consumption of ripple carry adder, carry look ahead adder, carry skip adder, carry select adder, and prefix adder are discussed. The power factor, the sum of products of probability and fan-out of all internal nodes, is presented. This thesis also studies the power and time trade-off with efficiency index which is the product of power factor and worst case gate counts. The result shows that the carry look ahead adder has the lowest efficiency index in the design of a 64 bit adder. The carry skip adder is the best one in a design of a 16 and 32 bit adder. The ripple carry adder is the best choice for an 8 bit adder.

This study also presents a low power prefix adder design which will reduce the power consumption of the prefix adder without lost of performance.

The Low-Power Design of Prefix Adder

by

Che-jen Chang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 5, 1997
Commencement June 1998

Master of Science thesis of Che-jen Chang presented on June 5, 1997

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Chair of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Che-jen Chang, Author

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES(Continued)

LIST OF TABLES

The Low Power Design of Prefix Adder

1. Introduction

1.1 Motivation

With the advancement of new technology, we develop higher and higher expectations on the portability of electronic devices that perform computation and communication for us. These devices such as notebook computer, pager, cellular phone, and personal digital assistant(PDA) allow us to have the freedom to be mobile without cumbersome power source. However, they challenge engineers to design systems which consume less power. With low power designs, batteries used can be lighter and smaller. This added dimension in design tradeoffs besides the traditional requirement of cost and performance has recently attracted many researchers to work on the optimization of power consumption in digital circuits.

Currently, the dominating technology for implementing digital circuits is Complementary Metal Oxide Silicon(CMOS). Many papers on reducing power dissipation in digital electronic system have been published[1][2][3]. There are three major sources of power dissipation in digital CMOS circuits -- switching, short-circuit, and leakage. Traditionally, the switching component has been the main power dissipation source. Dynamic power consumption is summarized with the equation:

$$P_{dynamic} = \alpha\, C_L\, V_{dd}^2 f \qquad (1.1)$$

where α is the switching probability of the circuit, C_L is the load capacitance, V_{dd} is the supply voltage, and f is the frequency of this circuit.

Another important component of digital computers is adders. Adders are used in many different parts of the digital computer. They are not only used in the Arithmetic Logic Unit (ALU) but also in address calculation. Adders are also used in multipliers and other functional units. Therefore, it is important to study adder design as well as to reduce the power dissipation of adders. Many different addition algorithms exist and they range from the simple ripple carry adder to the complex carry look ahead adder[4]. We are particularly interested in the power optimization of the prefix adders[5].

The prefix adder is one implementation of parallel adders. It is the fastest adder in its design family[6] that is mostly is used in designs that demand high performance. The key points of prefix adder are: 1) it defines the "carry propagation" and the "carry generation" terms; 2) a carry generation network circuit calculates the carry of the current stage with either propagation from previous stages or generation at the current stage; 3) the sum bit is calculated in parallel within a very short time delay.

1.2 Overview of the Thesis

Ripple carry adder, carry look ahead adder, carry skip adder, and carry select adder are some of most popular adders. These adders and the background of power consumption of CMOS circuits are included in the thesis. This study particularly focuses on the analysis of dynamic power consumption of each adder design. An indicator, power factor, is introduced to measure the dynamic power dissipation. It is defined as the sum of products of probability and fan-out of all internal nodes. A comparison of the power factor of different adders is included as well.

The main objective is to modify the topology of the carry generation network of a prefix adder and to reduce the number of fan-out of each node. A large fan-out not only has longer propagation delays but also consumes more power. The probability and fan-out of each node are analyzed to design the low-power prefix adder.

This thesis contains 6 chapters. Chapter 1 is the introduction. Chapter 2 is the background of adder designs, CMOS circuit power dissipation, and methods to reduce the power consumption. Chapter 3 demonstrates the methodology used in this study, and power factor and efficiency index are introduced. Chapter 4 is the analysis of power factor and efficiency index of adders. It also includes power factor analysis of adders while they are performing addition or counting. Chapter 5 is the experimental result of the prefix adder with low power design. An 8 bit low-power prefix adder is also presented. Chapter 6 is the summary of the thesis and recommendation for further research.

2. Background

This chapter includes introduction of several adder algorithms, discussion of the power consumption in a CMOS circuit, and review of several methods which can be applied to minimize the power dissipation of different CMOS circuits.

2.1 Adders

Several kinds of adders and the characteristics of time and power will be discussed. We will also include the comparison of the time and space between adders.

First of all, for a binary adder, its logic functions can be summarized by the following equations:

$$\begin{aligned} S &= \overline{A}\cdot B + A\cdot\overline{B} = A\oplus B \\ C &= A\cdot B \end{aligned} \tag{2.1}$$

where *S* is the sum bit of input bits *A* and *B*, and *C* is the carry out bit of *A* and *B*. *S* can be calculated by an *XOR* logic gate, and *C* can be calculated by an *AND* logic gate.

2.1.1 Ripple Carry Adder

The basic function of a one bit full adder(FA) is expressed in equation (2.2), where A_i and B_i are the input data. C_{in} is the carry in from previous stage. As for outputs, S_i is the sum bit of A_i and B_i, and C_{out} is the carry out bit.

$$\begin{aligned} S_i &= \overline{A}_i\cdot\overline{B}_i\cdot C_{in} + \overline{A}_i\cdot B_i\cdot\overline{C}_{in} + A_i\cdot\overline{B}_i\cdot\overline{C}_{in} + A_i\cdot B_i\cdot C_{in} \\ C_{out} &= A_i\cdot B_i + A_i\cdot C_{in} + B_i\cdot C_{in} \end{aligned} \tag{2.2}$$

Fig. 2.1 displays a 4 bit ripple carry adder. The ripple carry adder sequentially generate the carries and ripple through the next full adder stages. The problem of ripple carry adder is the long time delay. If we define the time delay of each full adder as δ, total time delay of a n bit ripple carry adder will be nδ.

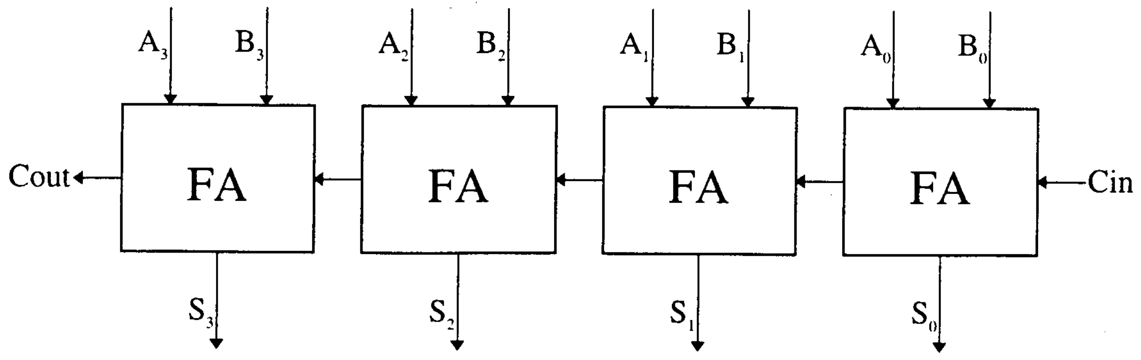


Fig. 2.1: Circuit of 4 bit full adder

The advantage of the ripple carry adder is its simplicity. Since its implementation takes less logic gates, the total power consumption is less than other adders.

2.1.2 Carry Look Ahead Adder

Many modifications have been made in the design of parallel adders to shorten the maximum time delay. One of the popular solutions is the carry look ahead adder[4]. The basic principle of a carry look ahead adder is that it calculates all of the values of carry bits before it begins to calculate the sum.

In order to implement the carry look ahead adder, we need to define the propagation bit(P) and generation bit(G) of each input signal in equations:

$$
\begin{aligned}
P_i &= A_i + B_i \\
G_i &= A_i \cdot B_i \\
C_{i+1} &= G_i + P_i \cdot C_i
\end{aligned}
\tag{2.3}
$$

The key point of carry look ahead adder is that the carry bit can be expressed in terms of the combination of P and G. Each carry bit, unlike in the ripple carry adder, can be calculated without rippling through the whole length of the adder. The logic function of the carry bits' in a 4 bit carry look ahead adder is presented in equations:

$$\begin{aligned}
C_1 &= G_0 + P_0 \cdot C_0 \\
C_2 &= G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0) = G_1 + P_1G_0 + P_1P_0C_0 \\
C_3 &= G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot (G_1 + P_1G_0 + P_1P_0C_0) \\
&= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0 \\
C_4 &= G_3 + P_3 \cdot C_2 = G_3 + P_3 \cdot (G_2 + P_2 \cdot G_1 + P_2 \cdot P_1G_0 + P_2P_1P_0C_0) \\
&= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0
\end{aligned} \tag{2.4}$$

An equivalent carry circuit of 3 bit carry look ahead is displayed in Fig. 2.2.

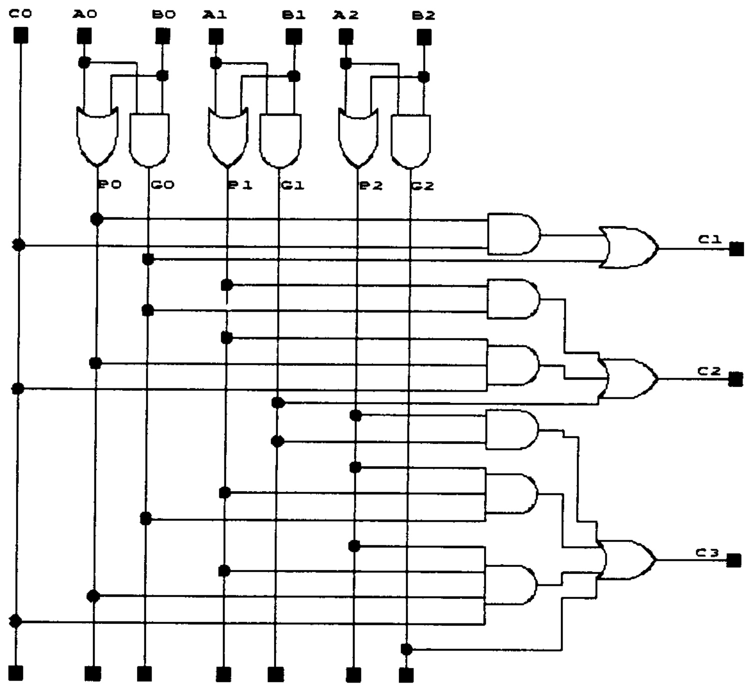


Fig. 2.2: The carry circuit of a 3 bit carry look ahead adder

There are some problems within the circuit. In an N-bit adder, the fan-out of the OR gates on the propagation bit is proportional to N. The fan-in of the OR gate is N+1. It is impracticable to build a full carry look ahead adder when N is large.

However, carry look ahead adders can still be practical when N is large if a simple and regular structure is used. The idea is to build up the P's and G's in steps. The equation of carry bits is known as:

$$C_1 = G_0 + P_0 C_0 \tag{2.5}$$

This equation means that there will be a carry out for the 1st position (C_1) if there is either a carry generated in the 0th position(G_0), or a carry in to the 0th position(C_0) and the carry propagates(P_0). Accordingly,

$$C_2 = G_{01} + P_{01} C_0 \tag{2.6}$$

G_{01} means there is a carry generated out of the block consisting of the first two bits. P_{01} means that a carry propagates through this block. P and G have the following logic functions:

$$\begin{aligned} G_{01} &= G_1 + P_1 G_0 \\ P_{01} &= P_1 P_0 \end{aligned} \tag{2.7}$$

Generally, for any j with i<j, j+1<k, we have the recursive relations:

$$\begin{aligned} C_{k+1} &= G_{ik} + P_{ik} C_i \\ G_{ik} &= G_{j+1,k} + P_{j+1,k} G_{ij} \\ P_{ik} &= P_{ij} P_{j+1,k} \end{aligned} \tag{2.8}$$

Equation (2.8) indicates that a carry is generated out of the block consisting of bits i through k inclusive if it is generated in the high-order part of the block (j+1,k) or if it is generated in the low-order (i,j) part of the block and then propagated through the high-part. These equations will also be hold for $i \le j < k$ if we set $G_{ij} = G_i$ and $P_{ij} = P_i$.

With these preliminaries, the design of a practical carry look ahead adder can be expressed. The adder consists of two parts. The first part computes various values of P and G from p_i and g_i, and the second part uses these P and G values to compute all the carries. The circuit of first part is presented in Fig. 2.3, and the second part is presented in Fig. 2.4.

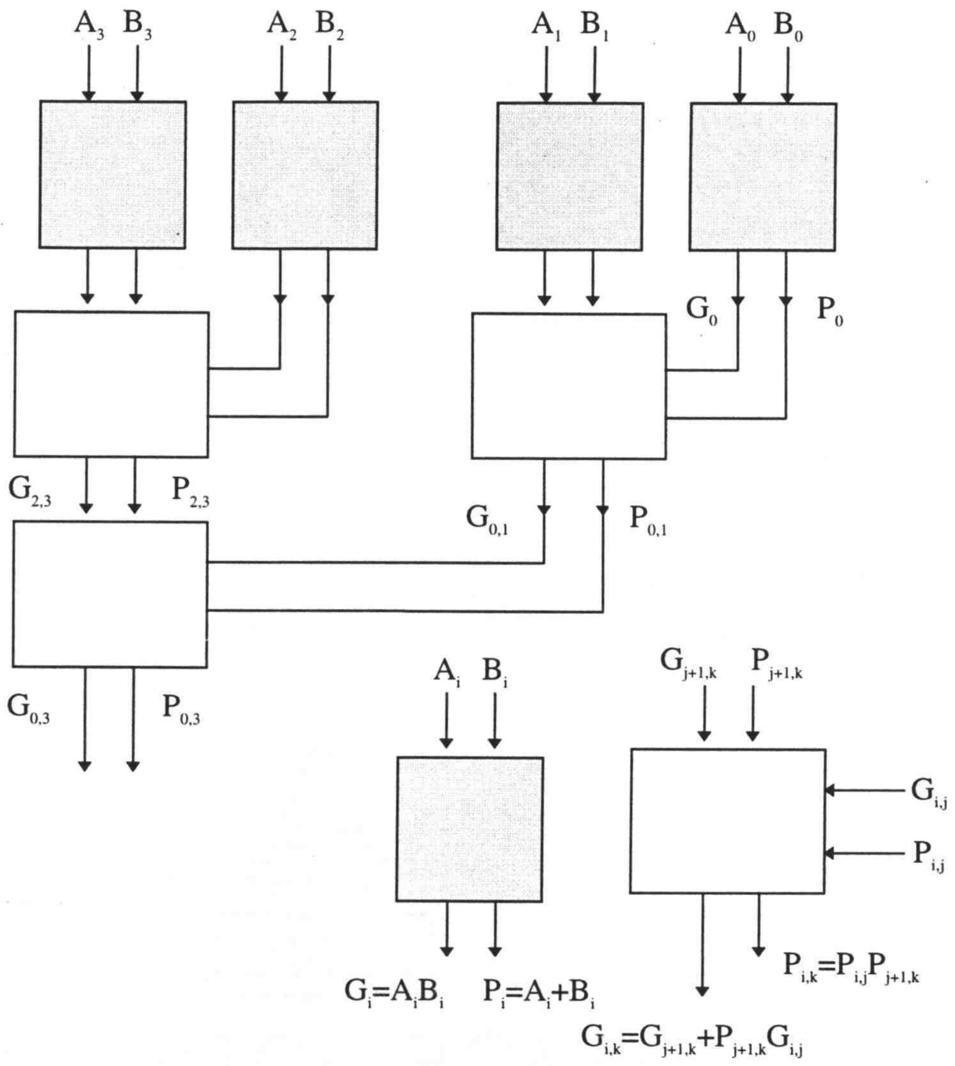


Fig. 2.3: The first part of carry look ahead adder

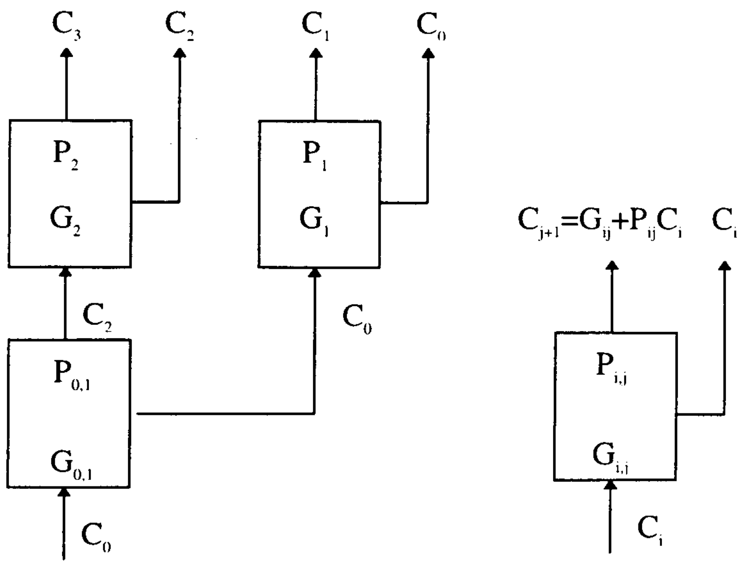


Fig. 2.4: The second part of carry look ahead adder

By feeding in C_0 at the bottom of this tree, all the carry bits come out at the top. Each cell must know a pair of (P,G) values in order to do the conversion, and the needed values are written inside the cells.

Comparing Fig. 2.3 and Fig. 2.4, there are one-to-one correspondences between cells, and the values of (P,G) needed by the carry generating cells are exactly the same values known by the corresponding (P,G) generating cells. The combined cells are presented in Fig. 2.5. The numbers needed to be added are flowing from the top and downward through the tree, combining with C_0 at the bottom, and flowing back up the tree to form the carries.

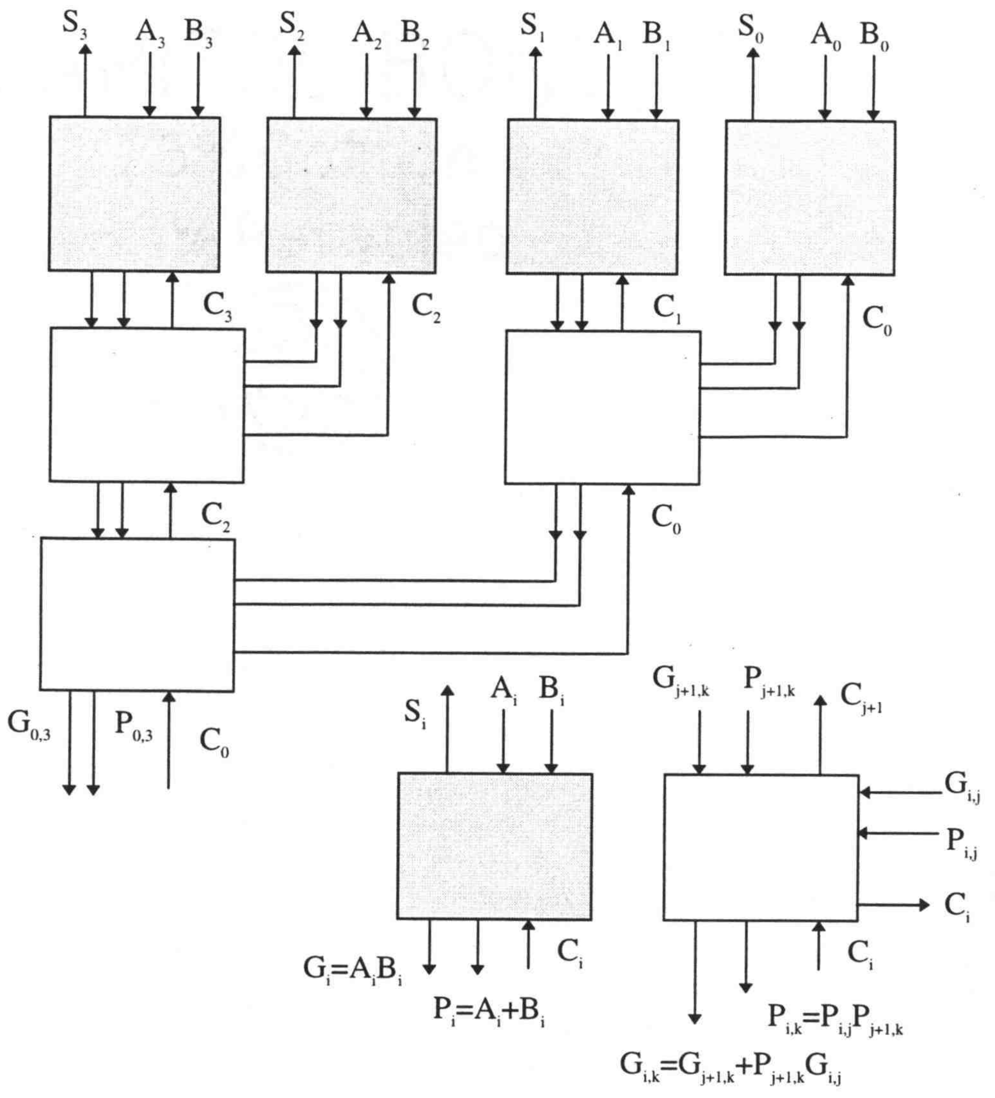
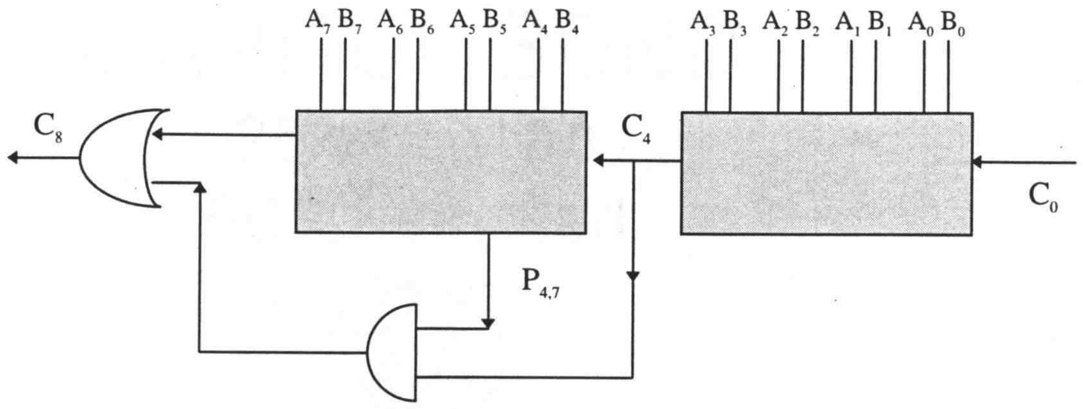


Fig. 2.5: The complete carry look ahead adder

For the carry look ahead adder, the maximum path length is the size of element delay and this delay remains almost constant no matter how many additional stages are provided in the adder. This is a significant increase in speed and the problem of time delay of the carry look ahead adder has been greatly improved. The total delays of N bit carry look ahead adder are $\log_2 N$.

2.1.3 Carry Skip Adder

A carry skip adder is mid-way between the ripple carry adder and carry look ahead adder. In the carry look ahead adder, the computation of P is much simpler than that of G. The carry skip adder computes only P to speed up. An 8 bit carry skip adder is illustrated in Fig. 2.6.



A_7 B_7 A_6 B_6 A_5 B_5 A_4 B_4
A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
C_8
C_4
C_0
$P_{4,7}$

Fig. 2.6: An example of an 8 bit carry skip adder

In Fig. 2.6, each gray block is a 4 bit ripple carry adder. Carries begin rippling simultaneously through each block. If any block generates a carry, the carry out of a block will be true even though the carry in of the block may not be corrected yet. If the carry in of each block is zero at the beginning of each add operation, no spurious carry out will be generated. Thus, the carry out from the least significant block is generated. It not only feeds into the next block, but is also fed through the AND gate with P signal from the next block. If the carry out and P signals are both true, the carry skips the second block and is ready to feed into the third block and so on.

The speed of the carry skip adder can be analyzed. Let us assume that it takes one time unit for a signal to pass through two logic levels. Then, it will take k time units for a carry to ripple across a block of size k and one time unit for a carry to skip a block. The longest signal path in the carry skip adder starts with a carry being generated at the 0_{th} position. It takes k time units to ripple through the first block, n/k-2 time units to skip blocks, and k more to ripple through the last block. To be specific: if we have a 20 bit adder broken into groups of 4 bit, it will take 11 time units to perform an add.

2.1.4 Carry Select Adder

Another modification of parallel adder, which attempts to shorten the maximum time delay, is the carry select adder. This circuit is faster than the carry look ahead adder, but it also has higher hardware cost.

A carry select adder works based on the following principles: two additions are performed in parallel--assuming one of the carry in is zero, and the other is one. When the carry in is finally known, the correct sum which has been pre-computed will be simply selected. An example is presented in Fig. 2.7.

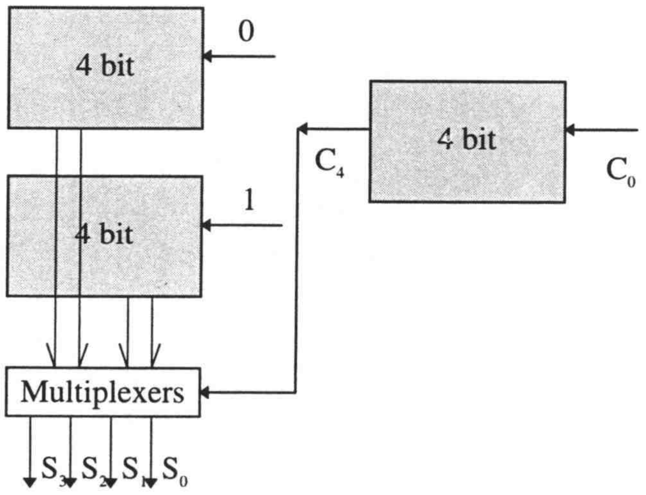


Fig. 2.7: A simple carry select adder

In Fig. 2.7, an 8 bit adder is divided into two halves, and the carry out from the lower half is used to select the output of upper half.

Another issue should be noticed here. The carry signal from the lower half must drive many multiplexers, which may cause great time delay. Instead of dividing the adder into halves, it could be divided into quarters for further speedup.

If it takes *k* time units for a block to add *k*-bit numbers and one time unit to compute the multiplexers inputs from two carry out signals, for optimal operation, each block should be one bit wider than the last one (Fig. 2.8). Therefore, in the carry skip adder, the best design involves various sized blocks.

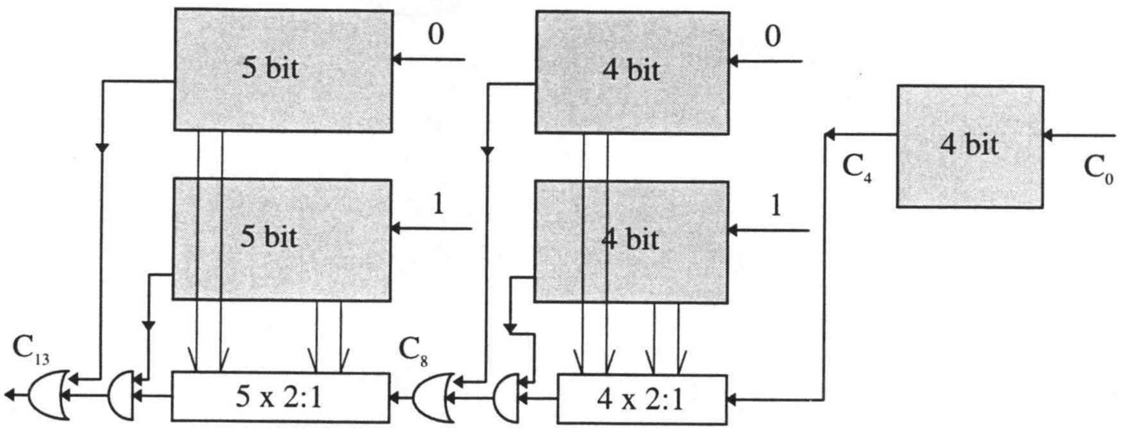


Fig. 2.8: Modified carry select adder

2.1.5 Prefix Adder

A prefix adder works like a carry look ahead adder. The idea of prefix problem is to compute all the products

$$X_1 \circ X_2 \circ \cdots \circ X_k \quad for\ i \leq k \leq n.$$

where $\circ$ is an associative operation. A recursive construction is used to obtain a product circuit for solving the prefix problem.

The mathematical model of the prefix addition is expressed in the following equations:

$$\begin{aligned} gIN_i &= a_i \cdot b_i \\ pIN_i &= a_i \oplus b_i \\ c_i &= G_i \quad \text{for } i = 1,2,\ldots,n \end{aligned} \tag{2.9}$$

where

$$(G_i, P_i) = \begin{cases} (gIN_i, pIN_i) & \text{if } i = 1 \\ (gIN_i, pIN_i) \circ (G_{i-1}, P_{i-1}) & \text{if } n \geq i > 1 \end{cases} \tag{2.10}$$

and $\circ$ is a concatenation operator which is defined as:

$$(g_l, p_l) \circ (g_r, p_r) = (g_l + p_l \cdot g_r, p_l \cdot p_r) \tag{2.11}$$

After the carry bit C_i is computed, the sum bit S_i is:

$$\begin{aligned} s_i &= pIN_i \oplus c_{i-1} \quad \text{for } i = 2,\ldots,n \\ s_1 &= p_1 \end{aligned} \tag{2.12}$$

Given the fact that $\circ$ is associative, *m* can be chosen such that $i \geq m > 1$ and $(G_{i,1}, P_{i,1})$ can be written as:

$$(G_{i,1}, P_{i,1}) = (G_{i,m}, P_{i,m}) \circ (G_{m-1,1}, P_{m-1,1}) \tag{2.13}$$

where

$$(G_{i,m}, P_{i,m}) = \begin{Bmatrix} (gIN_m, pIN_m) & \text{if } i = m \\ (gIN_i, pIN_i) \circ (G_{i-1,m}, P_{i-1,m}) & \text{if } i > m \end{Bmatrix} \tag{2.14}$$

$(G_{i,m}, P_{i,m})$ and $(G_{i-m+1,1}, P_{i-m+1,1})$ have similar function forms; they both are functions of *i-m+1* consecutive input bits and require *i-m* applications for the associative operator $\circ$. Therefore, both of them can be computed by the same circuit.

To implement these functions, three circuit blocks (Fig. 2.9) are required. The first one is a combination circuit, labeled as *Pre-condition Circuit*, which calculates the adder inputs a_i and b_i to generate the initial pIN_i and gIN_i for each bit position *i*. Secondly, the computed *p* and *g* are fed into the *Fast Carry Generator* which performs the operations defined in equations 2.9 to 2.14. It is this circuit that allows accelerated carry computations. The third block is a *Sum Circuit*, consisting of a row of XOR gates, to combine the carry propagate bits *(pIN_i)* from the first block with the carry bits *(c_i)* from the second block.

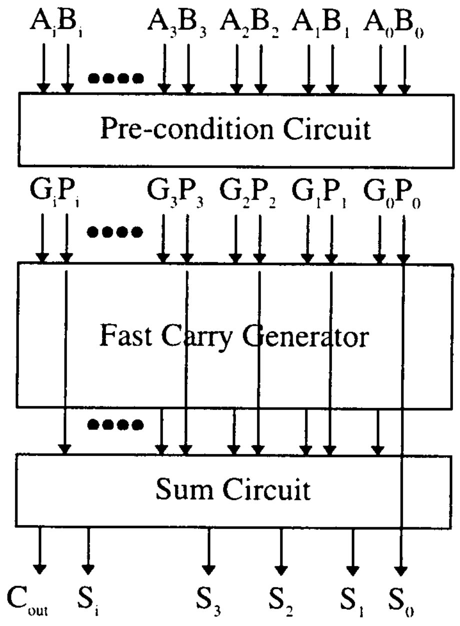
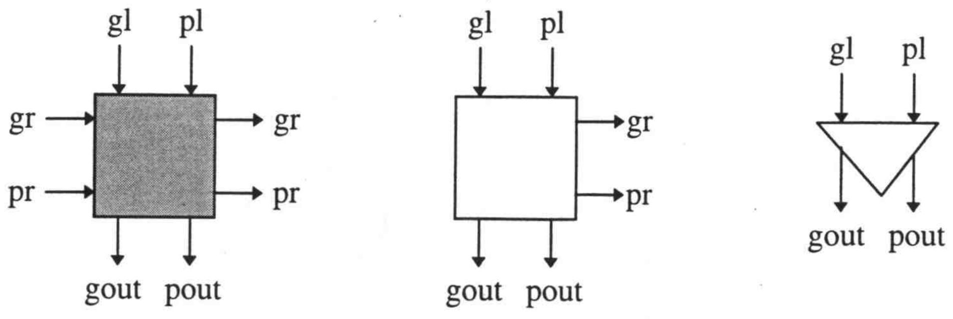


Fig. 2.9: Three functional blocks of a prefix adder

Fig. 2.10 shows three basic types of cells to implement the fast carry generator in the prefix adder: black cells, white cells, and driver cells. The black cell performs the associative concatenation. The white and driver cells act as "through" cells. An example of a 6 bit prefix adder is presented in Fig. 2.11.



pout=pl*pr
gout=gl+pl*gr

pout=pl
gout=gl

Fig. 2.10: Three basic cells of carry look ahead adder

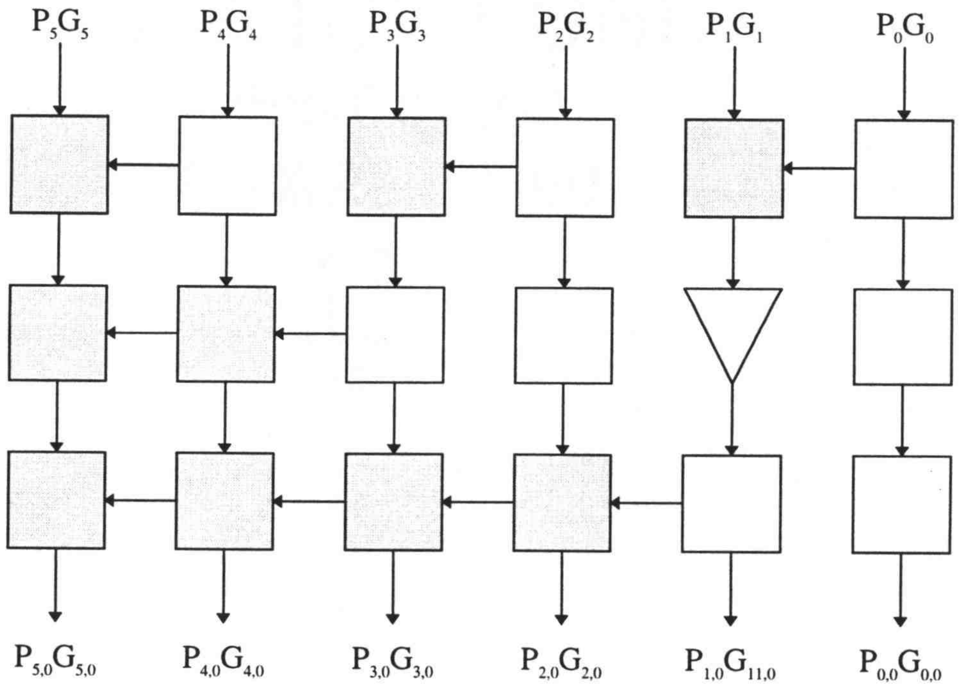


Fig. 2.11: An example of carry generation network of a 6 bit prefix adder

2.1.6 Comparison of Adders

The asymptotic time and space requirements for the different adders are summarized in Table 2.1. These different adders should not be looked at as disjoint choices, but as building blocks to be used in constructing an adder. The utility of these different building blocks is highly dependent on the technology used.

For example, the carry select adder works well when a signal can drive many multiplexers, and the carry skip adder is attractive in technologies where signals can be cleared at the beginning of each operation.

Table 2.1: Asymptotic time and space requirement for five adders

Adder	Time	Space
Ripple carry adder	O(N)	O(N)
Carry look ahead adder	O($logN$)	O($NlogN$)
Carry skip adder	O($\sqrt{N}$)	O(N)
Carry select adder	O($\sqrt{N}$)	O(N)
Prefix adder	O($logN$)	O($NlogN$)

2.2 Power Consumption of CMOS ICs

In digital CMOS circuit, there are four main sources of power dissipation which are summarized with following equations:

$$\begin{aligned} P_{avg} &= P_{switching} + P_{short-circuit} + P_{leakage} + P_{static} \\ &= \alpha_{0\to 1} \cdot C_L \cdot V \cdot V_{dd} \cdot f_{clk} + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} + I_{static} \cdot V_{dd} \end{aligned} \qquad (2.15)$$

$P_{switching}$ denotes the switching component of power, where C_L is the load capacitance, f_{clk} is the clock frequency, and $\alpha_{0\to 1}$ is the node transition activity factor (the average number of times that the node makes a power consuming transition in one clock period). The node transition activity factor is a function of the implemented logic function, the logic style, the circuit topologies, signal statistics, signal correlations, and the sequencing of operations.

$P_{short\text{-}circuit}$ is due to the direct-path short circuit current, I_{SC}, which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. Through proper choice of transistor sizes, the short-circuit power can be kept to less than 10% of total power consumption. Alternatively, operating the circuits at a supply voltage less than the sum of NMOS and PMOS threshold voltages will essentially eliminate any short-circuit currents.

$P_{leakage}$ is due to the leakage current, $I_{leakage}$, which can arise from reverse bias diode currents and sub-threshold effects, is primarily determined by fabrication technology considerations.

Finally, static currents, $I_{statics}$, arise from circuits that have a constant source of current between the power supplies (such as bias circuitry, pseudo-NMOS logic families, etc.). This static current will affect a lot if the circuit is idle most of the time (when the circuit is clocked at low frequencies), then the static power will tend to increase the total power consumption.

For properly designed circuits, the switching component of power will dominate and contribute to more than 90% of the total power consumption, which should be the primary target for power reduction.

2.3 Low Power Design

The fundamental cause of CMOS dynamic power dissipation is the organization of the energy transport in the circuit. When charging a node with node capacitance to voltage V_{dd}, a signal energy is stored in the node. When the node is discharged, the energy is drained away from the node to ground. Thus, all energy drawn from the supply is used only once before being discarded.

To decrease the dynamic power dissipation, some methods can be applied, such as minimizing the switching events, reducing the node capacitance, and decreasing the voltage swing. Combination of some or all of these methods can be used as well.

2.3.1 Minimizing Switching Capacitance

Since CMOS circuits do not dissipate dynamic power if they are not switching, primary approach of low power design is reducing the switching activity to the minimal level required for performing the computation. One of the simple methods is simply powering down the whole or portion of the circuit. However, sophisticated methods including gated clocks or optimizing circuit architectures can be used as well.

Following sections will describe the methods of the low power design in minimizing the switching capacitance at algorithm level, architecture level, logic level, circuit level, and physical level.

2.3.1.1 Algorithmic Optimization

The choice of algorithm is the most highly leveraged decision in meeting the power constraints. The ability for an algorithm to be paralleled will be critical and the basic complexity of the computation must be highly optimized.

First method of algorithmic optimization is to minimize the number of operations. For example, consider the problem of compressing a video data stream using the vector quantization algorithm. Three vector quantization algorithms are tested and the result for 16 pixels input vector are presented in Table 2.2[7].

Table 2.2: Computational complexity of vector quantization encoding algorithm

Algorithm	# of Memory access	# of Multiplications	# of Adds	# of Subs
Full Search	4096	4096	3840	4096
Tree Search	256	256	240	264
Differential Tree Search	136	128	128	0

Second method is minimizing temporal bit transition activity by choosing data representation. For example, Gray-coding is a popular coding algorithm used in low power design. The reason why it is so useful is because there is only one bit difference between consequence bits. Fig. 2.12 shows the reduction in switching activity for instruction address coding for a set of benchmark programs. BPI is the number of bit transitions per instruction executed[8].

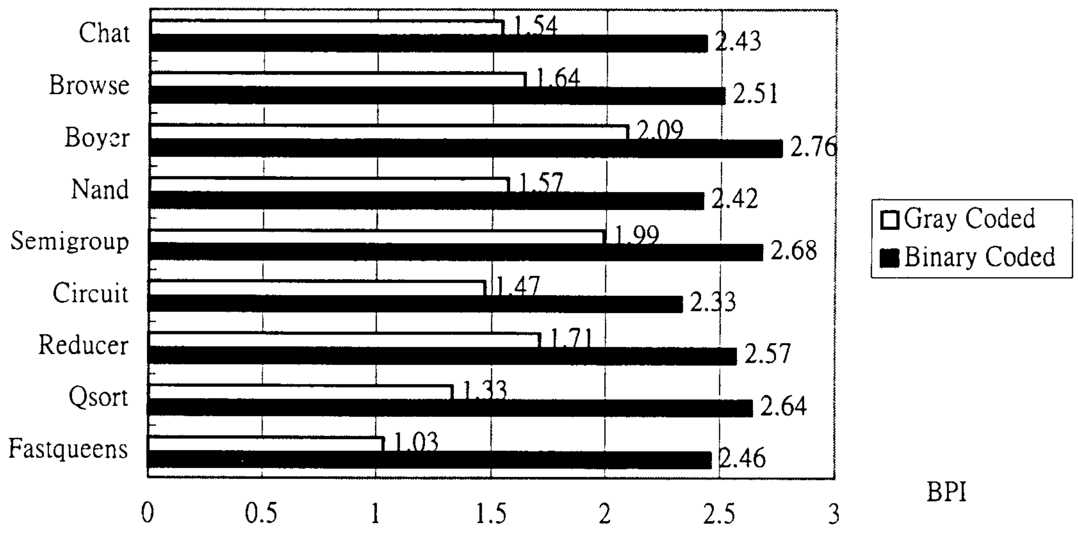


Fig. 2.12: Temporal transition activity comparison for instruction addresses

2.3.1.2 Architecture Optimization

Architecture optimization can also be used to significantly reduce the switching activity by optimizing the number representation, optimizing the ordering of operations, optimizing resource utilization, and minimizing glitching activity.

2.3.1.3 Logic Optimization

The choice of logic topology has a strong influence on the total transition activity, which will directly affect the switching activity and the power consumption. Callaway et. al.[9] emulated five kinds of adders with limited input sample to get the results of average number of gate transitions per addition in Table 2.3.

Table 2.3: Average number of gate transition per addition

Adder Type	16 bit	32 bit	64 bit
Ripple Carry	90	182	366
Carry Look Ahead	100	202	405
Carry Skip	108	220	437
Carry Select	161	344	711
Conditional Sum	218	543	1323

Their research showed that the carry look ahead adder was the best based on the product of transitions number and delay. These simulation results were poor because they were only based on 50,000 randomly distributed input patterns.

2.3.1.4 Circuit Optimization

There are a number of options available in choosing the basic circuit approach and topology for implementing various logic and arithmetic functions. Choices between static vs. dynamic implementations, passgate vs. conventional CMOS logic styles, and synchronous vs. asynchronous timing are some of the options for system designer.

First, though dynamic logic must have pre-charge operation and charge-sharing, it can reduce switching activity due to hazards, eliminate short-circuit dissipation, and reduce parasitic node capacitance. Dynamic logic style appears to be the better low power performance.

Second, passgate logic requires fewer transistors to implement logic functions, such as XOR. Besides, passgate logic can lower the threshold voltage and let it operate at the lowest possible voltage level, which is very important to low power design.

Third, self-timed implementations can minimize switching activity by power-down of unused modules. This is a better choice for low power design.

2.3.1.5 Physical Design

At the level of physical design, the place and route can be optimized. For example, signals with high switching activity can be assigned to short wires; signals with low switching activity can be allowed to have long wires.

2.3.2 Voltage Reduction

The dominant component of power consumption for properly designed CMOS circuits is proportional to the square of the supply voltage. Operating the circuit at the lowest voltage is the key to minimize the energy consumed per operation. However, the individual circuit element runs slower at lower supply voltages and this must be compensated through appropriate architectural design.

For example, if possible, we can reduce the supply voltage from 5V to 1.5V. This power reduction scale will be $1.5^2/5^2= 0.09$, which means 91% of power reduction. The trade-off is the increase of circuit delay. While reducing the voltage, there must be some slack in the critical path of the circuit so that the increased gate delays do not diminish the desired throughput. If not enough slack exist, changes must be made at the algorithm and architectural level to accommodate the slower gates. Some techniques, including parallelism and pipe-lining, that have been used to reduce the delay of critical paths can still maintain constant throughput when we reduce the supply voltage.

2.3.3 Minimizing Other Power Components

While the other components of power dissipation are generally minimal, there are design constraints that must be followed to prevent these components from becoming significant. Primary concern is the short-circuit power consumption – if signal rise/fall times are allowed to vary too much, this power can become a significant, or even the dominant component of the total power.

The reverse-bias diode leakage current power is a function of process and transistor count. In an example of one million transistor chip, the average leakage current is approximately 25μA, which is insignificant given that amount of transistors. Thus, leakage power is negligible in most CMOS ICs. Even that, it can only be optimized by minimizing the total diffusion area.

3. Methodology

In this chapter, the methodology that used in studying switching power consumption of different adders is presented. Two indicators, power factor and efficiency index, will be introduced. An example about these two indicators to illustrate the trade-off between the time and power in different adder designs is presented in the end of this chapter..

3.1 Probability Analysis

Two different probability analyses will be presented. The first one is addition, and the second one is counting.

3.1.1 Addition

The basic function of adder is to perform the addition. Two input signals come into the adder, then the sum bit and the carry out bit are generated. For a random input data, the probability of input equal to 1 is 0.5. Therefore, we define the probability of input as 0.5. The probability of each internal node is counted by the simulation program.

3.1.2 Counting

Counting is more frequently operated in general computing than addition. Usually, it is used by the program counter to control the program flow. Counting includes increment and decrement. In this study, we will simulate the power characteristic when the adder is performing the increment or decrement by 1, 2, 4, and 8.

3.2 Power Factor

In order to discuss the power in statistic way, the probability of each output changing from 0 to V_{dd} is considered. Power is drawn from the power supply and stored into the load capacitance. With the background discussed in the last chapter, the dynamic power consumption can be expressed as:

$$P = \alpha_{0\to 1} C_L V_{dd}^2 f \tag{3.1}$$

Since all internal nodes of a gate may have transitions, the transition activity of every node must be calculated in a circuit. The total power of a circuit is the sum of power of all internal nodes. For example, for a circuit with *n* internal nodes, the total power should be:

$$P_{total} = \sum_{i=1}^{n} \alpha_{0\to 1\, i} \cdot C_i \cdot V_{dd}^2 \cdot f = V_{dd}^2 \cdot f \cdot \sum_{i=1}^{n} \alpha_{0\to 1\, i} \cdot C_i \tag{3.2}$$

For a circuit, V_{dd} and f can be fixed values. So, the total power will be proportional to the sum of the products of $\alpha_{0\to 1}$ (switching probability) and C_L(load capacitance).

$$\sum_{i=1}^{n} P = Const \cdot \sum_{i=1}^{n} \alpha_i C_{Li} \tag{3.3}$$

where $Const = V_{dd}^2 f$. Each load capacitance of gate is assumed to be the same, and the interconnected capacitance is ignored. Therefore, the total load capacitance is proportional to the fan-out of each gate. Then, the sum of products of probability and fan-out of all internal nodes can be defined as *power factor*.

$$Power\ Factor = \sum_{i=1}^{n} \alpha_i \cdot Fanout_i \tag{3.4}$$

3.3 Efficiency Index

Time delay is another important issue in the circuit design. Mostly, time delay occurs when the load capacitance(C_L) is charging and discharging. The value of load capacitance will determine the delay of rising time and falling time. To achieve the goal of low power design, every gate should be the minimum size in order to save power. If the size of the N-transistor and P-transistor are the same, the rising time will become twice as the falling time. For a combination circuit, the maximum time delay is derived from the sum of gates charged from 0 to V_{dd}.

An efficiency index can be introduced here. We assumed that every gate has the same time delay. Based on this assumption, the maximum time delay will occur when all gates are in the rising stage. Using the power factor described in equation (3.4), the efficiency index can be expressed with the following equation:

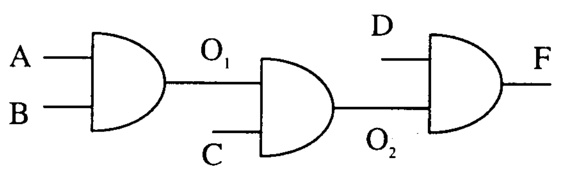
$$\mathit{Efficiency\ Index} = \mathit{Power\ factor} \cdot \mathit{Worst\ Case\ Gate\ Counts} \qquad (3.5)$$

With the Efficiency Index, we will be able to evaluate the tradeoffs between two systems with same function but different logic structure.

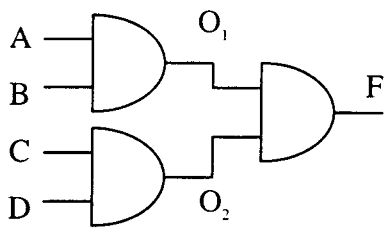
3.4 An Example of Power Factor and Efficiency Index Analysis

Based on the equations described in previous paragraphs, a simple example of power factor and efficiency index is presented. While illustrating the simple logic function in equation (3.6), there are alternative implementations: chain structure (Fig. 3.1(a)) and tree structure (Fig. 3.1(b)).

$$F = A \cdot B \cdot C \cdot D \qquad (3.6)$$



(a) chain



(b) tree

Fig. 3.1: Chain structure(a) and tree structure(b)

Fig. 3.1 includes two circuit topologies. Both circuits perform the same function but their power consumption and time delay is different. Those differences will be described in the following paragraphs.

First, the probability analysis of the internal nodes O_1 and O_2 is presented in Table 3.1. The rising probability (charging) of node O_2 in chain structure is smaller than that in tree structure.

Table 3.1: Probability analysis of chain and tree structure

	O_1	O_2	F
P_1(chain)	1/4	1/8	1/16
$P_0=1-P_1$(chain)	3/4	7/8	15/16
$P_{0\to1}$(chain)	3/16	**7/64**	15/256
P_1(tree)	1/4	1/4	1/16
$P_0=1-P_1$(tree)	3/4	3/4	15/16
$P_{0\to1}$(tree)	3/16	**3/16**	15/256

Both fan-outs of node O_1 and O_2, are 1. So, the power factors are:

$$\begin{aligned} \textit{Power Factor}(\textit{chain}) &= 3/16 \cdot 1 + 7/64 \cdot 1 = 19/64 \\ \textit{Power Factor}(\textit{tree}) &= 3/16 \cdot 1 + 3/16 \cdot 1 = 3/8 \end{aligned} \tag{3.7}$$

Based on the assumption that every gate has the same time delay, the efficiency indexes should be:

$$\begin{aligned} \textit{Efficiency Index}(\textit{chain}) &= 19/64 \cdot 3 = 57/64 \\ \textit{Efficiency Index}(\textit{tree}) &= 3/8 \cdot 2 = 3/4 \end{aligned} \tag{3.8}$$

Equation (3.8) shows that the efficiency index of tree structure is smaller than that of the chain structure although the chain structure has less power factor. It seems that the tree structure is a better choice than chain structure for designing the function in equation (3.6).

3.5 Simulation

Based on the concepts discussed previously, some C++ programs were written to simulate the gate level switching behavior of different adder circuits. These programs count the switching activities of every internal node based on every possible input. For example, for a 4 bit adder, two inputs should have 2^4 x 2^4 input combinations. First, we set the initial value of each internal node while the initial values come in. Then, another set of input signal is fed into the simulation program, and the rising and falling activities of every internal cell is counted. The algorithm is presented in Fig. 3.2.

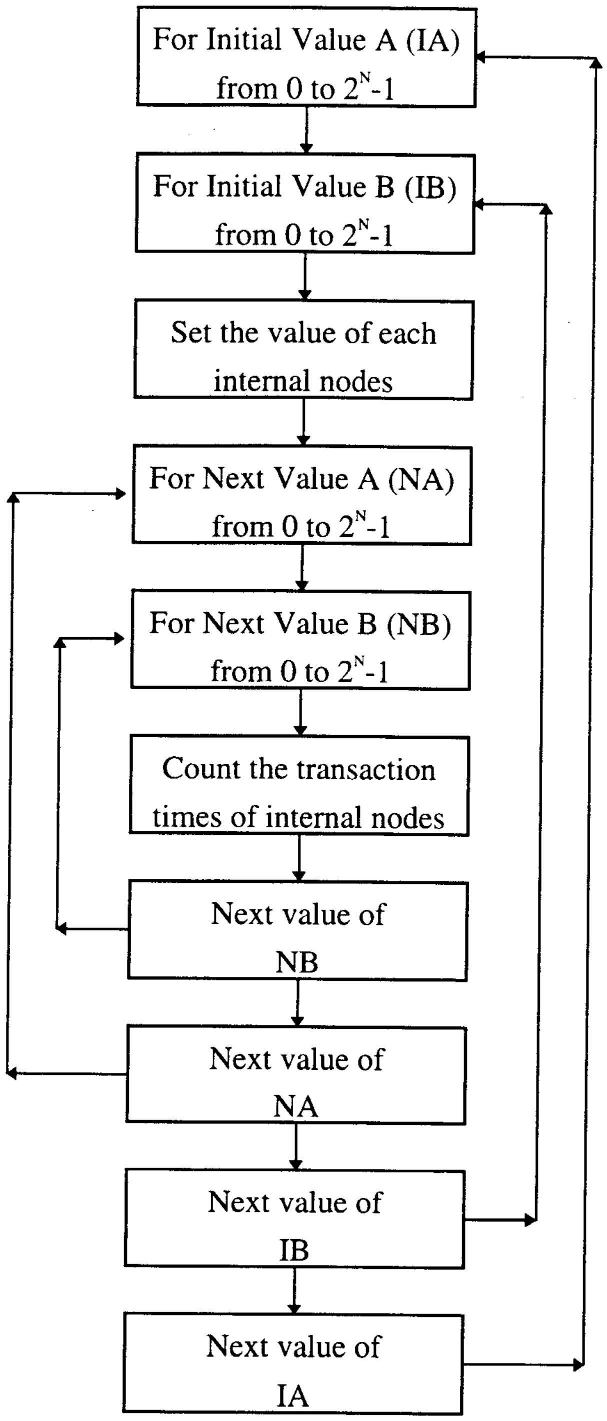

For Initial Value A (IA) from 0 to 2^N-1
For Initial Value B (IB) from 0 to 2^N-1
Set the value of each internal nodes
For Next Value A (NA) from 0 to 2^N-1
For Next Value B (NB) from 0 to 2^N-1
Count the transaction times of internal nodes
Next value of NB
Next value of NA
Next value of IB
Next value of IA

Fig 3.2: The algorithm of counting the probability of each cell

For the fan-out of each gate in the circuit, the sub-program defines it as the number of nodes connected with the output values. For example, the fan-out of node O_1 in Fig. 3.1(a) is 1.

The power factor value is calculated by summing up the products of the probabilities and fan-outs. The power factor value is proportional to the total power consumption of a circuit. The smaller the value is, the lower the power consumption is. Thus, the power factor can be used as an index for comparing the power consumption between different adder designs.

When the program is performing counting instead of addition, we change the NA value from a loop to the fixed number we want to count. For example, when decrease by 4, we will put (11111100) in the NA value and run the program.

4. Dynamic Power Analysis of Adders

In this chapter, dynamic power characteristic of adders will be analyzed based on the definition of chapter 3. The power factor and efficiency index of each adder will be presented and the comparison of different adders is included as well.

4.1 Ripple Carry Adder

Ripple carry adder is a combination of N-bit full adders. Each full adder has a carry out signal ripple through next stages in order to calculate the sum bits. When the adder is performing the general addition, it shows that the probability of sum bit being 1 is 0.5 and the probability of carry out bit being 1 is 0.5, too.

In a counting operation, the probability of each carry out bit being 1 is different while the sum bit remains the same. For example, when we use an 8 bit ripple carry adder to perform "increment by 1" operation, the probabilities of each carry out bit from bit 0 to 7 are 0.75, 0.375, 0.18755, 0.09375, 0.04688, 0.02344, 0.01172, and 0.00586 respectively. In fraction form, these numbers are 3/4, 3/8, 3/16, 3/32, 3/64, 3/128, 3/256, and 3/512, respectively.

The fan-out of sum bit in the full adder is 1 and the fan-out of carry out bit is also the same. By summing up the products of probability and fan-out of each bit, the power factor analysis of ripple carry adder is presented in Table 4.1.

Table 4.1: Power factor analysis of a ripple carry adder

Method	8 bit	16 bit	32 bit	64 bit
General Addition	4	8	16	32
Increment by 1	2.74415	4.74998	8.75	17.1667
Increment by 2	2.90693	4.91663	8.91667	17.3333
Increment by 4	2.98252	4.99993	9	17.4167
Increment by 8	3.00883	5.04154	9.04167	17.4583
Decrement by 1	2.41471	4.41666	8.41667	16.8333
Decrement by 2	2.74415	4.74998	8.75	17.1667
Decrement by 4	2.90306	4.91661	8.91667	17.3333
Decrement by 8	2.97099	4.99989	9	17.4167

For example, the power factor of an 8 bit ripple carry adder is 4 when the adder does the general addition. The power factor is proportional to the number of bits because of its regular layout. The counting operations consume less power than general addition because the increment and decrement operations have less carry chain and less switching probabilities. Comparing the decrement with the increment operation, the increment operation in the ripple carry adder has more switching probability than decrement has.

4.2 Carry Look Ahead Adder

Fig. 4.1 shows the structure of an 8 bit carry look ahead adder. In this figure, the gray blocks calculate the propagation and generation bit; the white blocks calculate the result of equation (2.6). The probabilities of propagation and generation bits being 1 are 3/4 and 1/4, respectively.

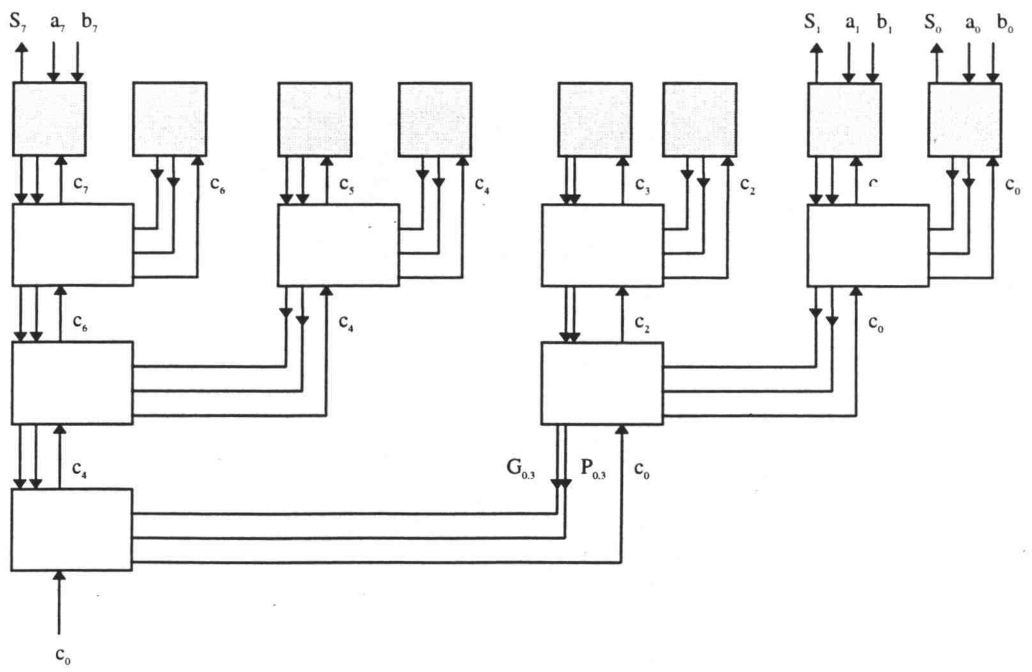


Fig. 4.1: An 8 bit carry look ahead adder

Table 4.2: Power factor analysis of a carry look ahead adder

Method	8 bit	16 bit	32 bit	64 bit
General Addition	14.266325	29.373644	59.417049	119.43512
Increment by 1	8.765146	14.810695	26.54528	49.764025
Increment by 2	8.94802	15.018055	26.752753	49.971498
Increment by 4	9.124691	15.243207	26.97813	50.196875
Increment by 8	9.198288	15.4118	27.147173	50.365919
Decrement by 1	5.8951111	10.464186	19.34846	36.846049
Decrement by 2	7.9506073	13.04445	22.428837	40.426426
Decrement by 4	7.7789154	12.7968	22.056412	39.929001
Decrement by 8	8.5079193	13.465657	22.569469	40.285808

Table 4.2 shows the power factor analysis of a carry look ahead adder. The carry look ahead adder has more power consumption than ripple carry adder has since it uses more circuits to generate the carry out without ripple through all the gates. It also shows that the power factor of counting operation is around half of the power factor of addition operation. When we increase the number of bits, the power factor of the counting operation is not proportional to the number of bits. The table shows that the decrement operation has smaller power factor than increment operation has.

4.3 Carry Skip Adder

An 8 bit carry skip adder is shown in Fig. 2.8. In our simulation program, we use a 4 bit ripple carry adder in each gray block. The switching probability of carry skip adder is more than that of ripple carry adder because of the carry skip circuit. Table 4.3 shows the power factor of carry skip adder.

Table 4.3: Power factor analysis of a carry skip adder

Method	8 bit	16 bit	32 bit	64 bit
General Addition	4.7558393	9.767518	19.790875	39.581751
Increment by 1	2.8407631	4.8473661	8.847392	16.847392
Increment by 2	3.0581093	5.0691012	9.0691445	17.069144
Increment by 4	3.2192116	5.2389502	9.239028	17.239028
Increment by 8	3.3220558	5.3591629	9.3593099	17.35931
Decrement by 1	2.5642438	4.8007008	9.2694589	18.089771
Decrement by 2	2.9524612	5.1930744	9.6618489	18.482161
Decrement by 4	3.2053134	5.4542078	9.923013	18.743327
Decrement by 8	3.3666675	5.6319996	10.100872	18.921185

The carry skip adder is in the mid-way between the ripple carry adder and carry look ahead adder. It has less complicated design than the carry look ahead adder has. The power factor of carry skip adder is higher than the power factor of the ripple carry adder because of the skip circuit. In the counting operations, the increment operation has less power factor than the decrement has.

4.4 Carry Select Adder

The circuit of an 8 bit carry select adder is shown in Fig. 2.6. The ripple carry adder is used in each gray block. We have two options for implementing the carry select adder architectures. The first one is to use fixed width blocks to implement the carry select adder. For example, if each block is a 4 bit ripple carry adder, a 16 bit adder is the combination circuit of 4 blocks of 4 bit ripple carry adders.

The second architecture is shown in Fig. 2.7. We use blocks with various width; each successive block is one bit wider than the last one. In our simulation program, we choose the combination of 4-4-5-6 to implement a 19 bit adder. The combination for a 34 bit adder is 4-4-5-6-7-8. For a 64 bit adder, the combination is 4-4-5-6-7-8-9-10-11. The speed of the various width carry select adder is faster than the fixed width adder because the carry signal ripple through less stages. Table 4.4 and Table 4.5 show the power factors of fixed and various width carry select adders.

Table 4.4: Power factor analysis of a fixed width carry select adder

Method	8 bit	16 bit	32 bit	64 bit
General Addition	7.583984	24.34523	58.09737	125.667
Increment by 1	5.519531	12.85278	27.2748	56.11855
Increment by 2	5.769531	13.17456	27.59668	56.44043
Increment by 4	5.957031	13.48218	27.90449	56.74824
Increment by 8	5.957031	13.62866	28.05137	56.89512
Decrement by 1	5.082031	24.27966	62.70137	139.5451
Decrement by 2	5.519531	24.68469	63.10605	139.9498
Decrement by 4	5.832031	24.90881	63.32949	140.1732
Decrement by 8	5.957031	24.76331	63.18262	140.0264

Table 4.5: Power factor analysis of a various width carry select adder

Method	8 bit	19 bit	34 bit	64 bit
General Addition	7.583984	30.00806	60.27868	120.2067
Increment by 1	5.519531	15.27783	28.05663	52.87793
Increment by 2	5.769531	15.60303	28.38186	53.20316
Increment by 4	5.957031	15.91748	28.69639	53.51768
Increment by 8	5.957031	16.07764	28.85668	53.67798
Decrement by 1	5.082031	29.69781	61.47648	122.2978
Decrement by 2	5.519531	30.0993	61.87782	122.6991
Decrement by 4	5.832031	30.31635	62.09454	122.9158
Decrement by 8	5.957031	30.15668	61.93425	122.7555

These tables show that the carry select adder really consumes more power than other adders because of its double addition circuits and multiplexers. Power factors show that the counting operation consumes less power than the addition operation does, and increment operations have better performance than decrement operations have. Decrement operation consumed twice the power as increment operation. The various width carry select adder has more improvement on power than the fixed width carry select adder has since it takes less stages to get the result.

4.5 Power Factor Analysis

After analyzing adders, we will discuss the differences between the power factors of adders. Fig 4.2 shows the power factor comparison of five adders when they are performing general addition. The ripple carry adder has the best performance in saving power. The second choice is the carry skip adder. The carry look ahead adder takes a lot more power than others when it is only 8 or 16 bit.

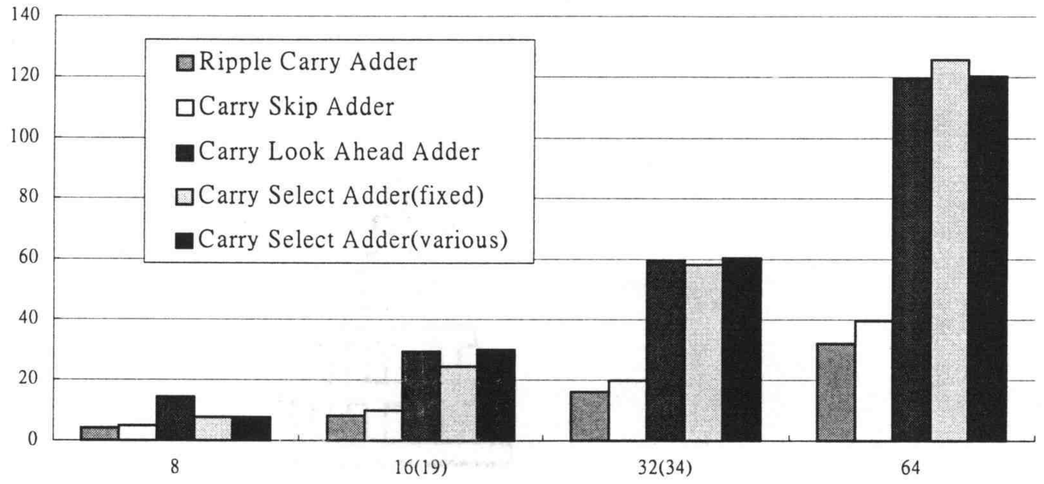


Fig. 4.2: The power factor comparison of adders

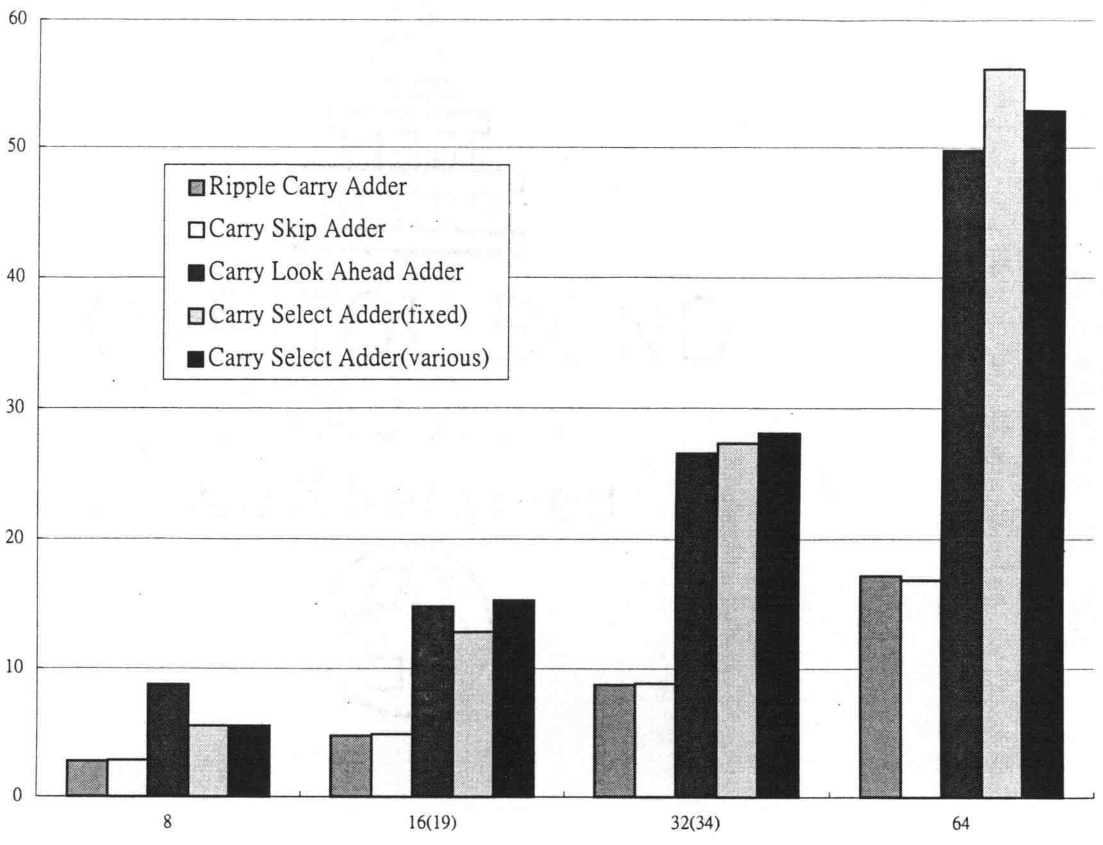


Fig. 4.3: The power factor analysis of adders when increment by 1

The result of "increment by 1" operation is presented in Fig. 4.3. For 8, 16, and 32 bit adders, ripple carry adder is the best choice. For a 64 bit adder, carry skip adder is better than others. Carry look ahead adder is the worst one in 8 bit adders. As for the 64 bit adder, fixed size carry select adder has the biggest power factor.

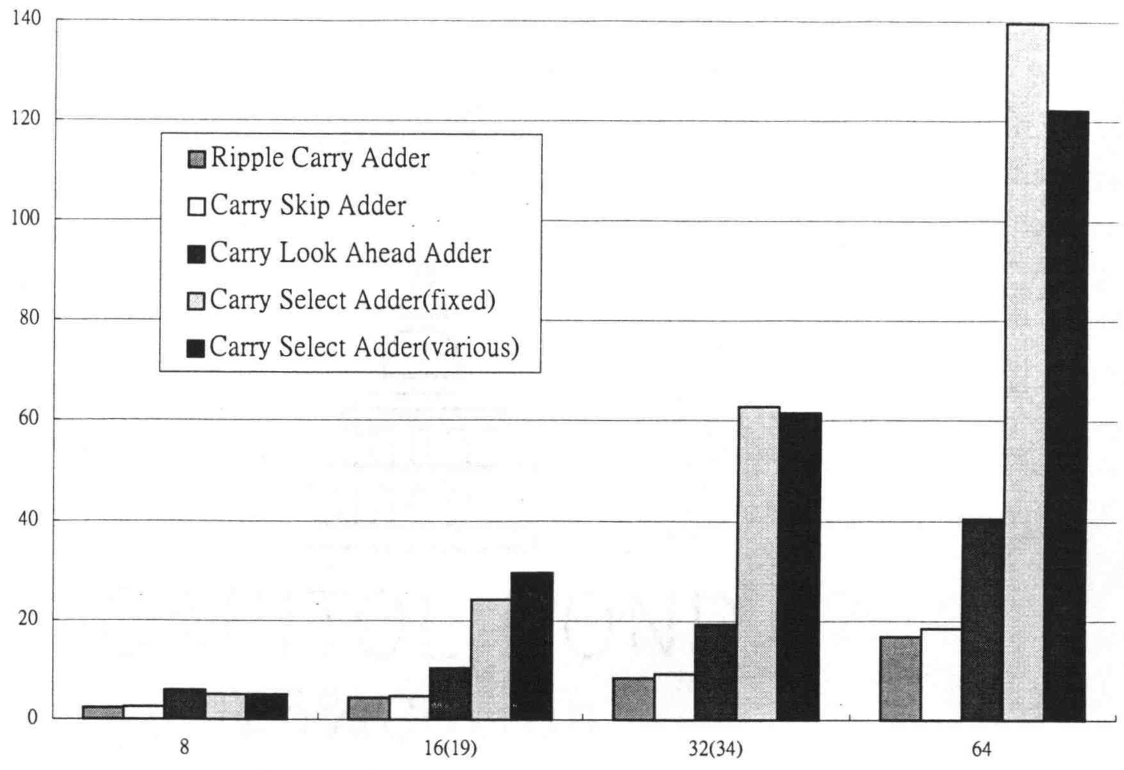


Fig. 4.4: The power factor analysis of adders when decrement by 1

Fig. 4.4 shows the comparisons of adders when performing “decrement by 1”. The ripple carry adder still is the best choice because of its low power factor. Carry skip adder is the second choice. For both fixed and various size of carry select adders, the power factor analyses show that they have the worst performance.

4.6 Efficiency Index Analysis

The comparison of the efficiency indexes of different adders is presented in the following paragraphs. The efficiency indexes of 8, 16, 32, and 64 bit adders are listed in Table 4.6, 4.7, 4.8, and 4.9.

Table 4.6: The efficiency index of 8 bit adders

Adder	Power factor	Longest Delay	Efficiency Index
Ripple Carry	4	8	32
Carry Look Ahead	14.266325	3	42.798975
Carry Skip	4.7558393	8	38.046714
Carry Select (fixed)	7.583984	5	37.91992

According to Table 4.6, the ripple carry adder has the smallest efficiency index. The carry look ahead adder does not show its improvement because it has a relatively big power factor. Carry skip adder and carry select adder have almost the same performance.

Table 4.7: The efficiency index of 16 bit adders

Adder	Power factor	Longest Delay	Efficiency Index
Ripple Carry	8	16	128
Carry Look Ahead	29.373644	4	117.494576
Carry Skip	9.767518	10	97.67518
Carry Select (fixed)	24.34523	7	170.41661

Table 4.7 shows that the carry skip adder is better than other 16 bit adders. Though the carry look ahead adder is the fastest adder, it consumes much more power than others. That make it the second choice of 16 bit adder design.

Table 4.8: The efficiency index of 32 bit adders

Adder	Power factor	Longest Delay	Efficiency Index
Ripple Carry	16	32	512
Carry Look Ahead	59.417049	5	297.085245
Carry Skip	19.790875	14	277.07225
Carry Select (fixed)	58.09737	11	639.07107

According to Table 4.8, the carry skip adder is still the best choice for a 32 bit adder. The table shows that the carry select adder is not a good adder design because it not only takes more power but also longer time delay.

Table 4.9: The efficiency index of 64 bit adders

Adder	Power factor	Longest Delay	Efficiency Index
Ripple Carry	32	64	2048
Carry Look Ahead	119.43512	6	716.61072
Carry Skip	39.581751	22	870.798522
Carry Select (fixed)	125.667	19	2387.673

In Table 4.9, it is shown that the carry look ahead adder is the best choice for a 64 bit adder. Its short time delay makes it the best choice. The ripple carry adder consumes the least power but takes a relatively long time that makes it not a good adder design.

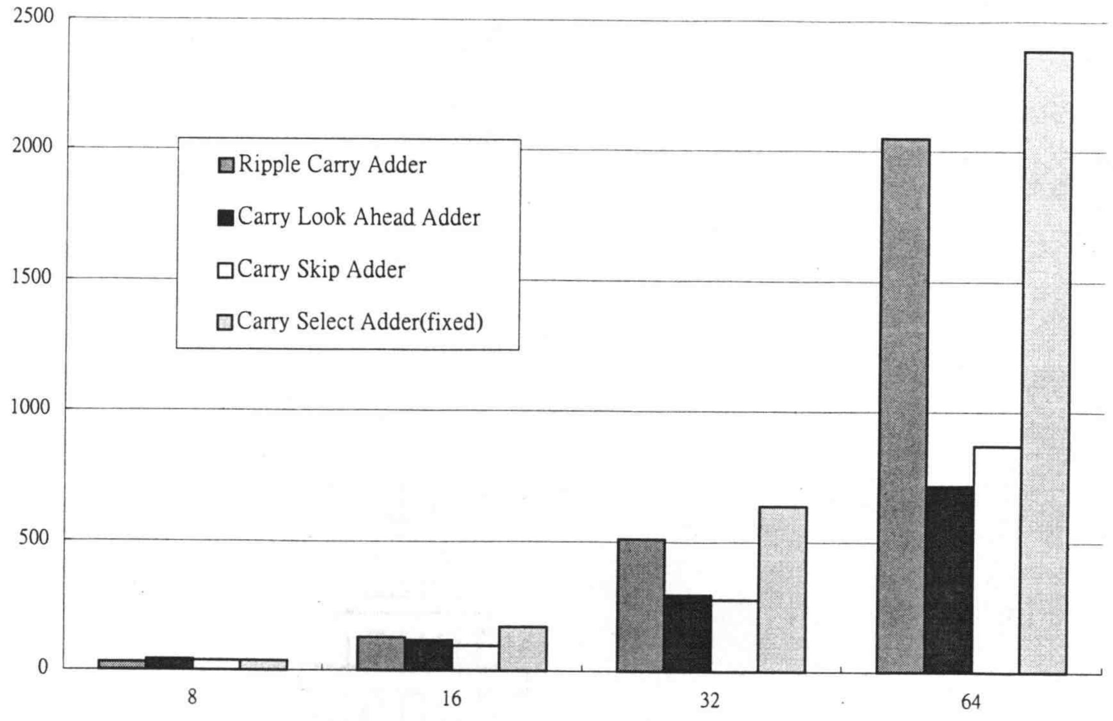


Fig. 4.5: The comparison of efficiency index between 4 adders

Fig. 4.5 is the comparison of the efficiency indexes of four adders. Based on this analysis, for an 8 bit adder, the ripple carry adder is the best choice. Carry skip adder is the best choice for 16 and 32 bit adder. Carry look ahead adder is the best adder for a 64 bit adder design.

5. Implementation of Low-Power Prefix Adder

This chapter included the results of simulation programs. First, the carry generation network of prefix adder designed by Wei, Thompson, and Chen[6] was analyzed. Then, programs were written to calculate every possible combination of carry generation network within the prefix adder. With those results, one may evaluate the power factor of every combination and find out the smallest number of power factor to achieve the low power design of the prefix adder.

For an 8 bit prefix adder, minimum depth of 3 is needed in the carry generation network. As the depth of the carry generation network increase, the number of possible combinations also increase. Those combinations will have different time delay and fan-out for each internal cell. Because of the change in topology, different power factor of the carry generation network can be evaluated. Fig. 5.1 shows the 8 bit prefix adder is designed by Wei, Thompson, and Chen.

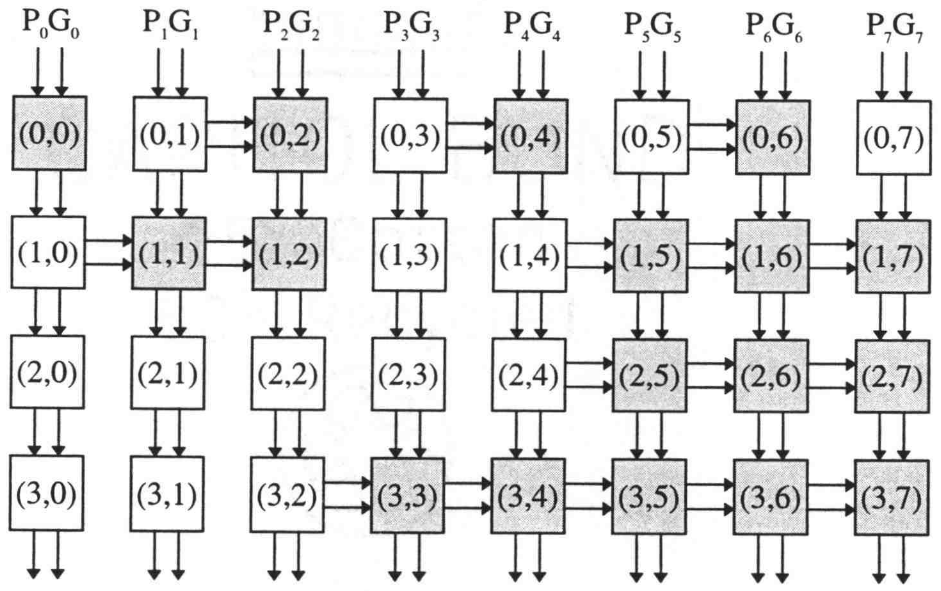


Fig. 5.1: Carry generation network of Wei's 8 bit Prefix adder

5.1 Analysis of Wei's Prefix Adder

The prefix adder is the fastest adder in their design family. The problem of the prefix adder is its complicated carry generation network which consumes a lot of power. In order to save the power without losing its speed, we need to analyze the switching probability and fan-out of each internal node. Then, we can compare the power factor between different combinations. The result shoes the one with lowest power factor is the low-power prefix adder designed in the thesis.

5.1.1 Probability Analysis

The probability of each cell is calculated by simulation program. The algorithm of the program is illustrated in Fig. 3.2. Table 5.1 is the probability analysis of P and G for each corresponding cell of Fig. 5.1. For example, *(1/4, 3/8)* is the probability of *P=1* and *G=1* of the cell (0,0).

Table 5.1: The probability analysis of Wei's 8 bit prefix adder

(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)
(1/4, 3/8)	(1/8, 7/16)	(1/16, 15/32)	(1/2,1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/8,7/16)
(1/4, 3/8)	(1/8, 7/16)	(1/16, 15/32)	(1/2,1/4)	(1/4,3/8)	(1/8,7/16)	(1/16,15/32)	(1/32, 31/64)
(1/4, 3/8)	(1/8, 7/16)	(1/16, 15/32)	(1/32, 31/64)	(1/64, 63/128)	(1/32, 31/64)	(1/64, 63/128)	(1/128, 127/256)

5.1.2 Fan-out Analysis

The fan-out of each cell inside the carry generation network plays a very important role. It has a great impact in the power consumption as well as the time delay. The bigger the fan-out is, the larger the total load capacitance is. Since the black cell and white cell have different function, it is possible to rearrange the topology to reduce the fan-out. From the analysis of topology, every possible combination will be test to evaluate its fan-out.

When output cell is black, the fan-out of propagation bit is 2 and the fan-out of generation bit is 1. When the output cell is white, the propagation and generation fan-out depend on how many connected black cells on the right of this white cell. In the example of Fig. 5.2, the propagation and generation bit of cell (1,4) are used by the cells (2,4), (2,5), (2,6), and (2,7). The fan-out of propagation and generation bit for cell (1,4) is 4.

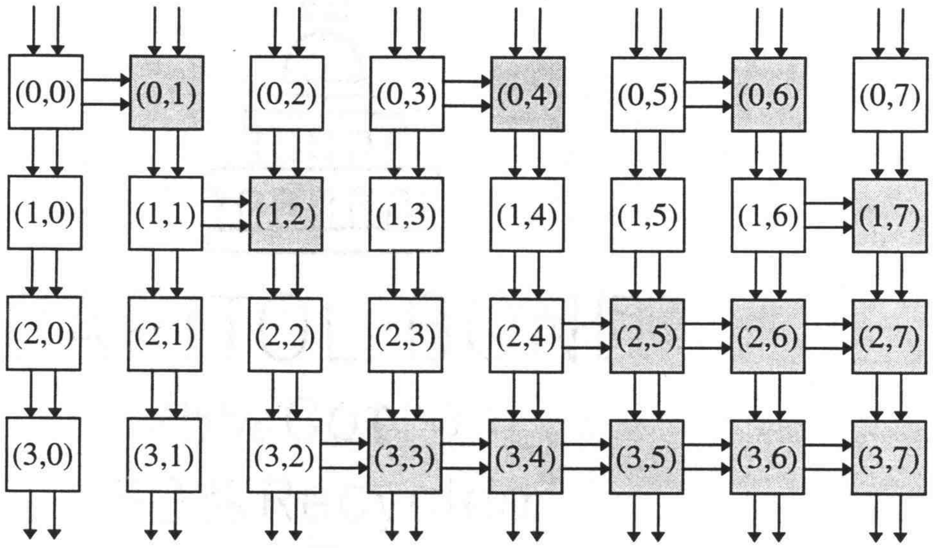


Fig. 5.2: An example of 8 bit carry generation network

The fan-out analysis of Wei's 8 bit prefix adder is shown in Table 5.2. The cell (2,2) have a very big fan-out which is the critical path of this prefix adder. This critical path is the source of the longer time delay and power consumption. Also, the cell (1,4) have the fan-out (4,4) for both the propagation and generation bit. It is very important to reduce the fan-out or distribute the fan-out to other cells.

Table 5.2: The fan-out analysis of Wei's 8 bit prefix adder

(3, 3)	(2, 1)	(2, 1)	(1, 1)	(1, 1)	(1, 1)	(2, 2)	(2, 1)
(1, 1)	(1, 1)	(1, 1)	(1,1)	(4, 4)	(2, 1)	(2, 1)	(2,1)
(1, 1)	(1, 1)	(6, 6)	(2,1)	(2, 1)	(2, 1)	(2, 1)	(2, 1)
(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)

5.1.3 Power Factor Analysis

Table 5.3 shows the power factor analysis of Wei's 8 bit prefix adder (Fig. 5.1); the prefix adder consumes much more power than any other adders. It is necessary to reduce the power consumption by proper arrangement of the black and white cells in the carry generation network.

Table 5.3: The power factor analysis of Wei's 8 bit prefix adder

Method		Power factor
General Addition		17.85698
Increment by	1	11.1674
	2	11.67924
	4	11.58037
	8	11.43925
Decrement by	1	14.44597
	2	14.84282
	4	15.01396
	8	13.36601

5.2 Low Power Prefix Adder Design

According to our analysis, a simulation program is designed to find out every possible arrangements of the carry generation network. It also checks the validity of the result. With these valid combinations, power factor analysis is needed to determine which one consume less power.

Fig. 5.3 shows the best model of the carry generation network for an 8 bit prefix adder. Based on our power factor analysis, this design has the lowest amount of power factor, the design consumes less power than others.

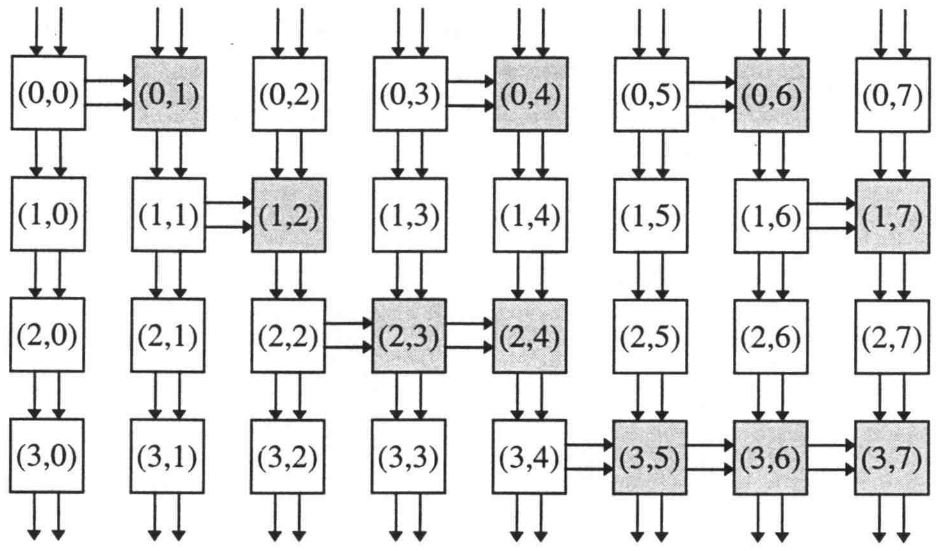


Fig. 5.3: The 8-bit low-power prefix adder design

Table 5.4 is the probability analysis of this low-power design, Table 5.5 is its fan-out analysis, and Table 5.6 is its power factor analysis. One can remark that has some improvement compare to Wei's prefix adder design presented in Table 5.3.

Table 5.4: The probability analysis of low-power 8 bit prefix adder

(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/2, 1/4)
(1/2, 1/4)	(1/4, 3/8)	(1/8, 7/16)	(1/2,1/4)	(1/4, 3/8)	(1/2, 1/4)	(1/4, 3/8)	(1/8,7/16)
(1/2, 1/4)	(1/4, 3/8)	(1/8, 7/16)	(1/16,15/32)	(1/32, 31/64)	(1/2, 1/4)	(1/4, 3/8)	(1/8, 7/16)
(1/2, 1/4)	(1/4, 3/8)	(1/8, 7/16)	(1/2, 1/4)	(1/4, 3/8)	(1/8, 7/16)	(1/16, 15/32)	(1/32, 31/64)

Table 5.5: The fan-out analysis of low-power 8 bit prefix adder

(1, 1)	(2, 2)	(2, 1)	(1, 1)	(1, 1)	(1, 1)	(2, 2)	(2, 1)
(1, 1)	(1, 1)	(3, 3)	(2,1)	(2, 1)	(1, 1)	(1, 1)	(1,1)
(1, 1)	(1, 1)	(1, 1)	(1,1)	(4, 4)	(2, 1)	(2, 1)	(2, 1)
(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)

Table 5.6: The power factor analysis of low-power 8 bit prefix adder

Methods		Power factor
General Addition		16.65601
Increment by	1	10.54199
	2	10.40918
	4	10.00293
	8	10.38867
Decrement by	1	14.27148
	2	14.13867
	4	13.48242
	8	12.35742

6. Conclusion and Future Work

6.1 Conclusions

This thesis is the first study that uses software to exhaustedly exercise all input combinations to find out the probability of every internal node. Using this method, this thesis introduced two indicators which are the power factor and the efficiency index to evaluate the power consumption and trade-off of digital circuit. The study includes the power factor and efficiency index analyses of ripple carry adder, carry look ahead adder, carry skip adder, carry select adder, and prefix adder.

From the power factor analysis, the ripple carry adder has the best performance in saving power. The second choice is the carry skip adder. When performing the "increment by 1" operation, the ripple carry adder is also the best choice for 8, 16, and 32 bit adders. The carry skip adder is the best choice for a 64 bit adder design. When performing the "decrement by 1" operation, the ripple carry adder is still the best solution.

According to the efficiency index analysis, ripple carry adder has the lowest efficiency index for an 8 bit adder design. It is the best choice because its simple and regular logic structure. Carry skip adder has the lowest efficiency index for 16 and 32 bit adder designs. The reason why it is the best is because it uses a simple "skip" circuit to shorten a lot of delay. The carry look ahead adder is the best choice for a 64 bit adder design. Though its sophisticated circuit consumes a lot of power, its architecture makes it the fastest design among the 4 adders.

The prefix adder is the fastest adder in its design family. After studying the power characteristics of the prefix adder, we can design the prefix adder by properly arranging the black and white cells in the carry generation network and making it consume less power without losing its speed.

The method used to reduce the power consumption of the prefix adder is by reducing the switching probability on some critical paths. The power consumption is proportional to the product of the probability and the fan-out. The smaller the power factor is, the smaller the power it consumes. With the design which is illustrated at the end of Chapter 5, we can enjoy the high speed without consuming a lot of power.

6.2 Future Works

Future work can be done by expanding the circuit to a more complex level. It may be expanded to a 16 bit prefix adder or a hybrid adder design. Moreover, we can apply this power factor and efficiency index to more complicated circuit designs. For example, different multiplexers designs can use the concept of power factor and efficiency index to evaluate their power characteristic and trade-off between time and power.

This study only emphasizes the statistical model of the power consumption analysis. Real circuit layout and SPICE verification can be done in further studies. The model of power analysis is only shown of the gate level. It would be possible to simulate the circuit down to the transistor level to get more accurate results.

Bibliography

[1] A. P. Chandrakasan, and R. W. Brodersen, "Low Power Digital CMOS Design", Kluwer Academic Publishers, 1995.

[2] T. Burd, "Low-Power CMOS Library Design Methodology", Master Thesis of UC Berkeley, 1994.

[3] N. Asher, "Area, Delay and Power characterisitcs of CMOS Adders", Master Thesis of Penn Stat U., 1996.

[4] J. L. Hennessy, and D. A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[5] R. E. Ladner, and M. J. Fischer, "Parallel Prefix Computation", JACM, Vol. 27, No. 4, pp. 831-838, 1980.

[6] B. W. Y. Wei, C. Thompson, and Y. Chen, "Time-Optimal Design of a CMOS Adder", Research Paper, UC Berkeley, 1984.

[7] A. Gersho, and R. Gray, "Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1992.

[8] C. Su, C. Tsui, and A. Despain, "Low-power Architecture Design and Compilation Techniques for High-Performance Processors", Compcon, pp. 489-498, 1994.

[9] T. Callaway, and E. Swatzlander, Jr., "Optimizing Arithmetic Elements for Signal Processing", VLSI Signal Processing V, pp. 91-100, IEEE Special Publication, 1992.

[10] A. De Gloria, and M. Olivieri, "Statistical Carry Lookahead Adders", IEEE Transactions on Computers, Vol. 45, No. 2, pp. 340-347, 1996.

[11] D. J. Kinniment, J. D. Garside, and B. Gao, "A Comparison of Power Consumption in Some CMOS Adder Circuits", Research Paper, Oxford University.

[12] M. A. Cirit, "Estimating Dynamic Power Consumption of CMOS Circuits", ICCAD, pp. 534-537, 1987.

[13] S. L. Lu, "Average Carry Chain Length", Research Paper, Oregon State University.