1-1-1987

VLSI design of high-speed adders for digital signal processing applications.

Seyfollah Seyfollahi Bazarjani
University of Windsor

VLSI DESIGN OF HIGH-SPEED ADDERS FOR DIGITAL SIGNAL PROCESSING APPLICATIONS

by

Seyfollah Seyfollahi Bazarjani

A Thesis
Submitted to the
Faculty of Graduate Studies and Research
through the Department of
Electrical Engineering in Partial Fulfillment
of the requirements for the Degree
of Master of Applied Science at
the University of Windsor

Windsor, Ontario, Canada

1987

ABSTRACT :

In this work the VLSI design and implementation of high-speed arithmetic circuits for digital signal processing applications, using the Residue Number System, is investigated. Different techniques for high-speed Binary and RNS arithmetic implementation are discussed. It will be shown that for high-speed, high-precision arithmetic computations, pipelined RNS adders offer advantages over their binary counterparts. These advantages manifest themselves in throughput rate, latency time, hardware complexity, and testability.

Various CMOS logic families are presented. Charge redistribution problems in DOMINO and DCVSL are addressed and some techniques to alleviate these problems are studied. Delay time analysis of Sample-Set Differential Logic, which has not yet appeared in the literature, is described and an analytical expression for the delay time of an SSDL gate is derived. This expression is applied to the delay time optimization of SSDL gates. The application of SSDL in pipelined architectures is presented along with a simplification in the circuit design.

ACKNOWLEDGEMENTS

My Thanks and sincere appreciation is directed especially to Dr. G.A. Jullien for his guidance, support and creative suggestions. I am also thankful to Dr. W.C. Miller for his support. I would also like to thank Dr. S. Bandyopadhyay, and Dr. M. Sid-Ahmed for their Assistance.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Digital signal processing has applications in a variety of areas such as, speech processing, biomedical engineering, geophysics research, telecommunication, and image processing. In the majority of these applications high-speed and high-precision computation is essential in order to have real-time and accurate processing. The term "real time processing" is defined as "the processing of data at the same rate as the input data rate". Consider, for example, an image processing application such as image enhancement. Smoothing operations are used to diminish the noises, due to sharp transitions in the gray levels, of the image. One technique for image smoothing is neighborhood averaging. The smoothed image is obtained by replacing the gray level of each pixel by the average of the gray level values of the surrounding pixels. Consider an image of size MxN pixels and T.V. scan rate of K frames/s; if L arithmetic operations are required for each pixel, the total number of arithmetic operations that have to be performed in one second is K.L.M.N . To process an image in real time, a processor has to be able to process the image at 30 frame/s for flicker-free viewing. For a 512x512-pixel image, 30 frame/s, and a 3x3 window, a multiplier/accumulator would have to operate at 71 Mhz to process the image in real time [1].

Some other applications can require throughputs from 100 Mhz to 1 Ghz. Achieving such a high-speed operation requires careful study and improvements in three major areas: arithmetic, architectural, and implementation domains [2]. Parallelism is naturally achieved over the dynamic range by using Residue Number System (RNS) arithmetic. The use of RNS arithmetic results in inherently parallel hardware designs because of independence over the dynamic range of arithmetic operations within each modulus. More on RNS is discussed in chapter 3.

In the architectural domain, pipelining techniques are utilized to increase the computation rate of a digital system. The effectiveness of this method depends on the structure of the algorithm. This structure is very effective when the algorithm is applied repeatedly to a stream of input data. In this case a significant increase in speed is obtained with only moderate increase in hardware.

High-speed low-power CMOS VLSI technology is used for the implementation of digital signal processors. The three main implementation approaches are as follows:

I - Microprocessor approach

II - Semicustom Integrated Circuit approach

III - Full Custom Integrated Circuit approach

Implementing a high-throughput rate (e.g. 200 Mhz) DSP function with microprocessors and building blocks is neither possible nor cost effective. Even dedicated processors such as Finite Impulse Response (FIR) filter chips fall short in bandwidth.

Semicustom approaches such as gate arrays, standard cells, and ASIC's (Application-Specific) also cannot provide the speed needed for most of the real time signal processing applications.

Full custom DSP approaches, which use 1-micron or submicron technology is often the only solution for meeting the speed performance requirement. Custom IC development is usually characterized by being expensive and having long development cycles. However, because of the regularity of DSP architectures the development time can be reduced. Even in some cases, large functional blocks such as FIR filter and Fast Fourier Transform (FFT) can be added to the library as macros. This reduces the development cycles, making the custom approach competitive with semicustom approaches [1]. In the circuit domain investigations must be carried out to determine the most suitable logic family to be used. A complete study of CMOS logic design techniques is given in chapter 4.

1.2 COMPUTER AIDED DESIGN TOOLS UTILIZED

A DAISY chipmaster work station was used for all mask layouts. The masks were design rule checked and verified using the Phoenix Data System integrated software package. The circuits were simulated extensively using the SPICE 2G analog simulation program to predict circuit performance and delay times.

1.3 OBJECTIVES

The main objective of this thesis is " CMOS VLSI design and implementation of high-speed arithmetic for digital signal processing applications". To achieve this goal the following steps has been taken:

1 - Investigation of various algorithms for high-speed arithmetic in Binary.

2 - Study of Residue Number System arithmetic and investigation of different algorithms for RNS addition.

3 - CMOS VLSI design and implementation of pipelined binary and RNS adders.

4 - Study of different CMOS logic families including, DOMINO, NORA, DCVSL, LDOMINO, CLOCKED CMOS, and SSDL.

5 - Design and optimization of a static full adder for standard cell application.

6 - Analysis and optimization of SSDL gate for pipelined architectures.

1.4 THESIS ORGANIZATION

In this chapter, the need for high-speed arithmetic in various areas of digital signal processing applications was described. The three main areas namely, arithmetic, architectural, and implementation domains, which all require careful study for speed improvement, have been reviewed.

Chapter 2 describes the random logic design of binary arithmetic circuits. Different techniques for designing fast two-operand adders are presented and logic design of parallel multipliers are discussed. Pipelined architectures

are studied and the structure of high throughput rate adders and multipliers are also explained.

Residue Number System arithmetic is reviewed in chapter 3. Different methods of implementing RNS adders are discussed and the design of a pipelined RNS adder is presented. Finally, it is shown that for wide-dynamic range of operations (cross over point of 10 bits), the RNS adders have advantages over binary adders in four areas; namely hardware complexity, speed, testability, and latency time.

Chapter 4 contains a comprehensive study of various CMOS logic families. Charge redistribution problem of dynamic CMOS logic circuits is addressed and different techniques to alleviate this problem are presented. A simple K-map procedure for the design of differential cascode voltage switch tree circuit is discussed. Delay time analysis of sample-set differential logic (SSDL) which has not appeared in the literature so far, is described and an analytical expression for the delay time is obtained. This expression is applied to the delay optimization of SSDL gates. The application of SSDL gates to pipelined structures is also presented along with a simplification of the SSDL structure. Finally, testing issues of CMOS VLSI digital circuits are studied.

Chapter 5 presents a number of designs for the implementation of CMOS full adders. Design optimization of a transmission gate full adder for use in a standard cell library is discussed.

Chapter 6 contains the summary and main conclusions of this work.

CHAPTER 2

LOGICAL DESIGN OF BINARY ARITHMETIC CIRCUITS

2.1 INTRODUCTION

This chapter will discuss the design methodology of combinational logic circuits for use in binary arithmetic units. One of the most important components in any digital arithmetic architecture is the binary adder. Adders are essential not only for addition, but also for subtraction, multiplication, and division. The operational speed of digital arithmetic processors depends on the speed performance of the adders which are used in the system. For example, high-speed digital signal processors require parallel pipelined multipliers. The throughput rate of these multipliers depends on the delay of the 1-bit full adder cell used in the circuit [3]. So the delay introduced by the adder usually limits the maximum clock frequency at which a system may operate.

In section 2.2 an overview on logic design of the Half Adder (HA), Full Adder (FA), and Ripple-Carry (borrow) Adder (Subtractor) is presented. High-speed adders will be discussed in section 2.3, and section 2.4 deals with the structure of parallel multipliers. Finally, pipelined binary arithmetic architectures will be studied.

2.2 LOGIC DESIGN OF BASIC BINARY ADDER (SUBTRACTOR)

In this section the design of combinational logic circuits for the binary half adder, full adder, and ripple-carry

(borrow) adder (subtractor) will be examined. Let us consider the addition of two binary digits A and B .

```
Addend      A
Augend    + B
          ----
           CS
```

C : Carry

S : Sum

The half adder adds two binary digits A and B to produce a sum output S and a carry output C. The truth table and block diagram for the Half-adder is shown in Fig. 2.1. From the truth table of the half-adder the following boolean equations can be obtained:

Carry = A.B

Sum = $A \oplus B$

Fig. 2.2 shows the logic circuit which realizes these equations. The addition of two n-bit numbers requires an adder with three inputs (A;addend bit, B;augend bit, and Cin; carry information from the previous stage) to produce a sum output S and a carry output C. This type of adder is called a Full Adder and the truth table and block diagram of the full adder can be seen in Fig. 2.3. The two outputs are related to the three inputs by the following boolean equations:

$$\begin{aligned} \text{Sum} &= ABC + A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C \\ &= A\,(\,\overline{B \oplus C}\,) + \overline{A}\,(\,B \oplus C\,) \\ &= A \oplus B \oplus C \end{aligned}$$

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

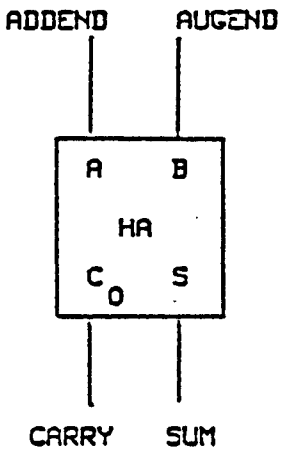


Fig. 2.1 Truth Table and Block Diagram of Half Adder

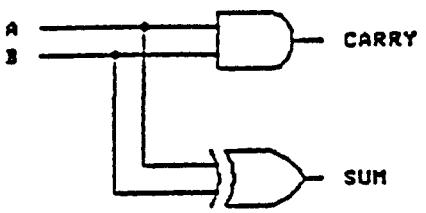


Fig. 2.2 Half Adder (HA) Logic Circuit

$$\text{Carry} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$= AB + C (A \oplus B)$$

$$= (ABC + \bar{A}BC) + (ABC + A\bar{B}C) + (ABC + AB\bar{C})$$

$$= AB + AC + BC$$

$$\text{Carry} = AB + C (A + B)$$

The schematic diagram of the full adder is shown in Fig. 2.4. This basic adding cell can be modified to become a 4-input 4-output Controlled Adder/Subtractor cell (CAS) as shown in Fig. 2.5 [4]. The additional input P is used to control the ADD (P = 0) or SUBTRACT (P = 1) operations. In the case of subtraction, the Ci input is called the borrow-in and the Ci+1 output, the borrow-out. The input-output relationship of a CAS cell is specified by the following boolean equations.

$$Si = Ai + (Bi \oplus P) + Ci$$

$$Ci+1 = (Ai + Ci)(Bi \oplus P) + Ai\ Ci$$

When P = 0 these equations are identical with the equations of the full adder. When P = 1 we have

$$Si = Ai \oplus \overline{Bi} \oplus Ci$$

$$Ci+1 = Ai\ \overline{Bi} + \overline{Bi}\ Ci + Ai\ Ci$$

By cascading n Full adders, an n-bit ripple-carry adder is formed. This cell can also be used for subtraction. Subtraction of two's complement numbers is performed by first obtaining the two's complement of the subtrahend and then adding it to the minuend. The schematic design of a binary two's complement adder/subtractor is illustrated in Fig. 2.6. The initial carry input to the rightmost full adder is connected to the function mode line M, which equals

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

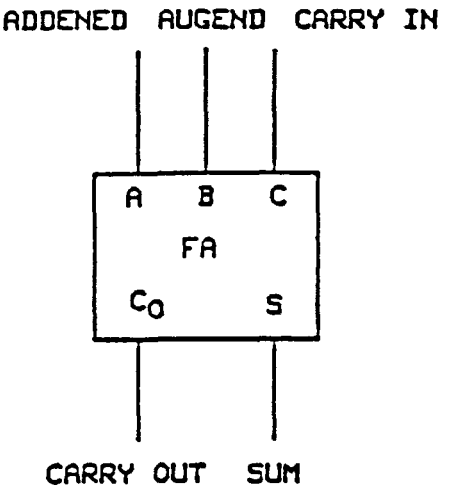


Fig. 2.3 Truth Table and Block Diagram of Full Adder

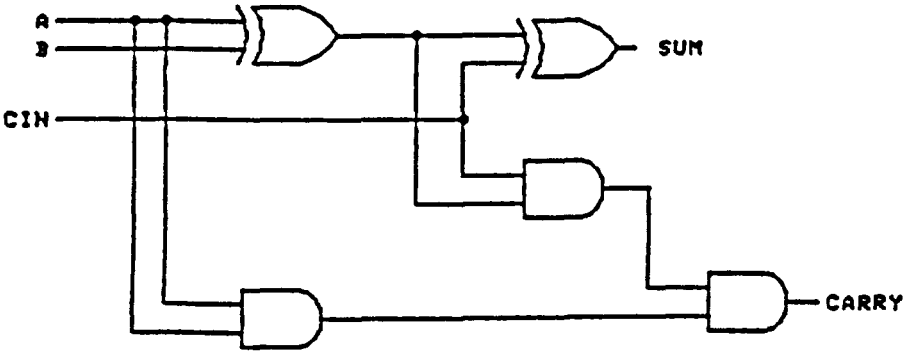


Fig. 2.4 Full Adder (FA) Logic Diagram

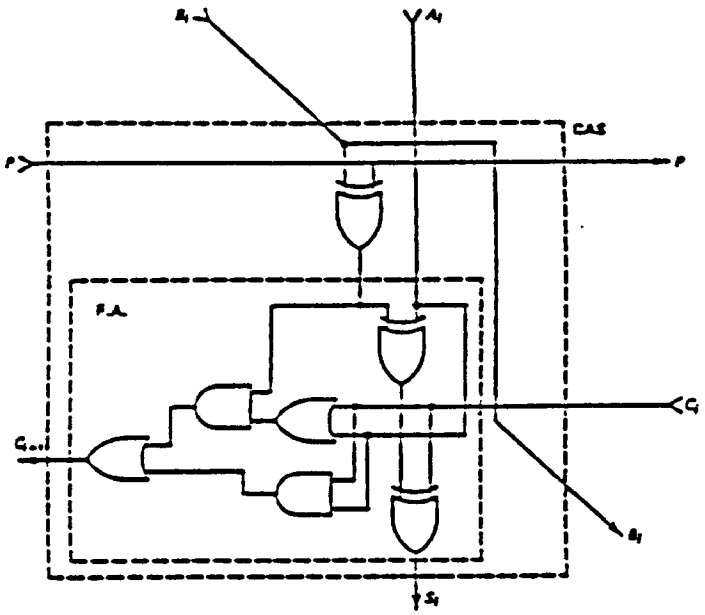


Fig. 2.5 Controlled Add/Subtract (CAS) Cell

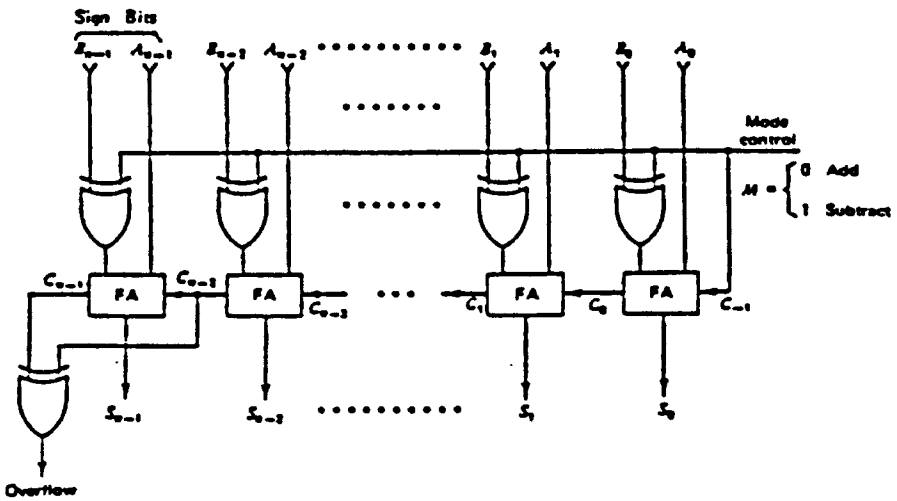


Fig. 2.6 Two's Complement Binary Adder/Subtractor Configuration

"0" for addition and "1" for subtraction. This ripple through carry (borrow) is a problem when high-speed operation with a large number of bits is required. The delay time of an n-bit carry ripple adder is almost equal to nTc, where Tc is the delay of one carry stage. In the next section we will explain some methods for high-speed addition.

2.3 HIGH-SPEED ADDERS

2.3.1 CARRY LOOK-AHEAD ADDER

As we have seen in the previous section, the delay time of the ripple carry adder is linearly proportional to the size of the input variables. Carry look-ahead (CLA) is a technique which is used to speed up the carry propagation in an adder. The carries to each stage of a parallel adder are calculated simultaneously by additional logic circuitry. As a result the addition time will improve at the cost of using more hardware for the carry look-ahead unit.

Let us denote the addend and augend of an n-bit adder by :

A = An-1 ..... A1A0

B = Bn-1 ..... B1B0

We will let, Ci-1 be the carry input to the i-th stage. The carry input to the first stage (least significant position) is denoted by C-1. Let the Si be the output sum and Ci be the carry of the i-th stage, now we define :

Gi : the i-th carry generate function.

Gi = Ai.Bi

Pi : the i-th carry propagate function.

$P_i = A_i + B_i$

substituting Gi and Pi into equations for sum and carry of the full adder we get:

$S_i = P_i \oplus C_{i-1}$

$C_i = G_i + P_i C_{i-1}$

These equations show that since Pi and Gi are generated simultaneously, for i = 1,2,3, ... , n , then all the sum bits can be computed in parallel if Cn-2, ..., C2, C1, C0 are available. Because of fan-in limitation of CMOS circuits, the number of stages of carry look-ahead adder is usually limited to four. Four bit carry look-ahead equations are:

$C_{-1} = C_{in}$

$C_0 = G_0 + P_0 C_{-1}$

$C_1 = G_1 + P_1 C_0$

$C_2 = G_2 + P_2 C_1$

$C_3 = G_3 + P_3 C_2$

Now if we expand each carry equation in terms of Pi, Gi and Cin we obtain :

$C_0 = G_0 + C_{in} P_0$

$C_1 = G_1 + G_0 P_1 + C_{in} P_0 P_1$

$C_2 = G_2 + G_1 P_2 + G_0 P_1 P_2 + C_{in} P_0 P_1 P_2$

$C_3 = G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + C_{in} P_0 P_1 P_2 P_3$

This set of equations shows that all the carries can be generated simultaneously, and as a result the sum also can be calculated simultaneously.

$S_0 = P_0 \oplus C_{in}$

$S1 = P1 \oplus C0$

$S2 = P2 \oplus C1$

$S3 = P3 \oplus C2$

and C3 is the carry overflow. The functional block diagram of a 4-bit carry look-ahead adder is illustrated in Fig. 2.7. The gate schematic of different blocks of a carry look-ahead adder are shown in Fig. 2.8 .

Theoretically one should be able to expand the CLA unit freely and build CLA adders of any word length. Due to the constraints described before (fan-in and fan-out limitation), however, single-level CLA is applied only to the design of adders of lengths 4 in CMOS circuits.

The total delay time of a 4-bit carry look-ahead adder is the sum of the delay times due to the propagate-generate unit (NAND2), carry look-ahead unit (NAND5), and summation unit (XOR).

2.3.2 RIPPLE CARRY LOOK-AHEAD ADDER

As we mentioned before, the number of stages in the carry look-ahead adder is limited to four. One solution to the high fan in problem is to break the large single CLA unit into a number of smaller CLA units and let the carries ripple between the units. The organization of a 12-bit ripple carry look-ahead adder with three carry look-ahead units each of size 4 is shown in Fig. 2.9. The total delay of this type of adder is the sum of the delays due to propagate/generate unit (NAND2) and sum unit (XOR) plus the delays through all CLA units (3 NAND5).

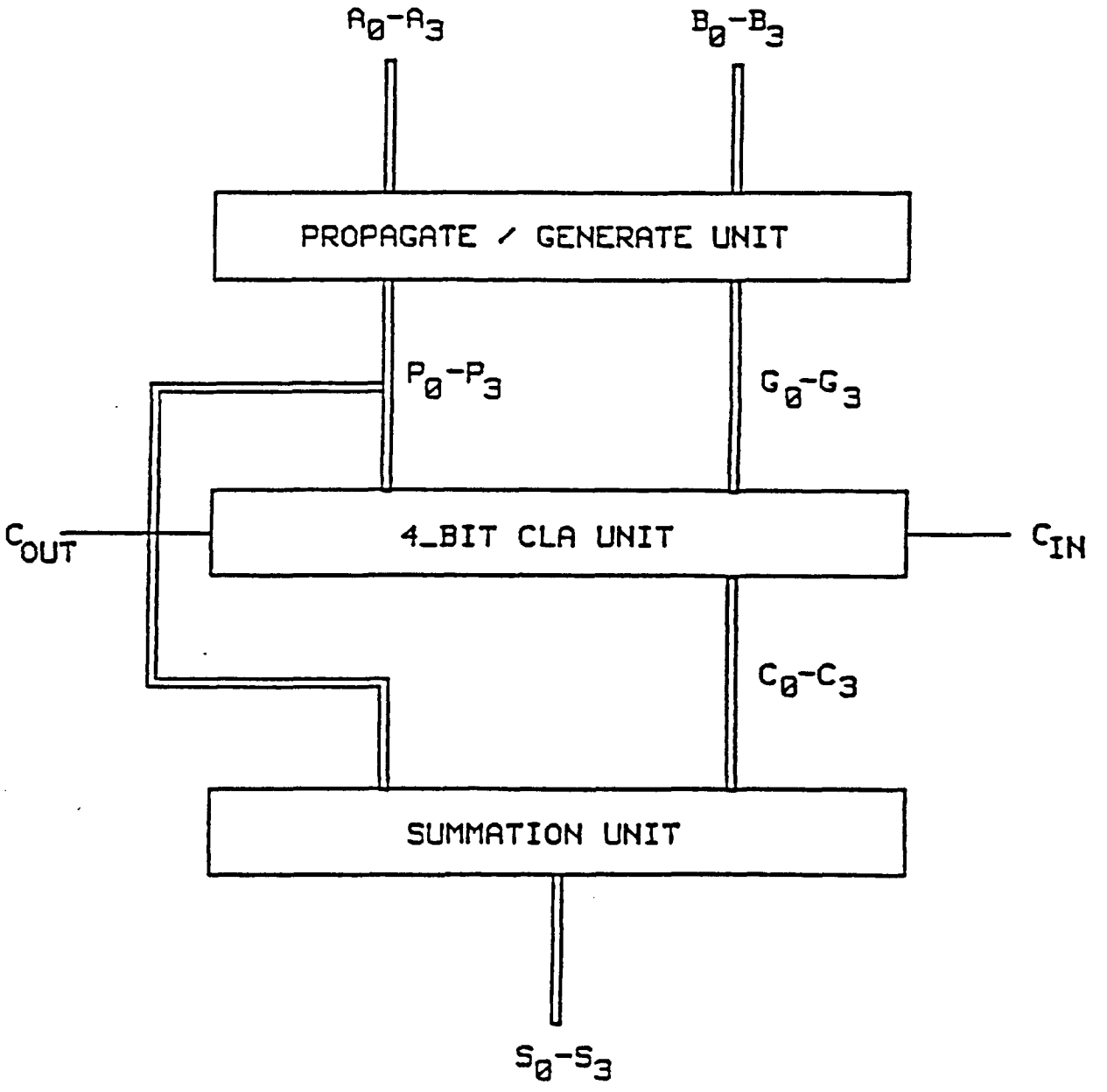


Fig. 2.7 The Functional Block Diagram of a 4-bit Carry Look-Ahead Adder

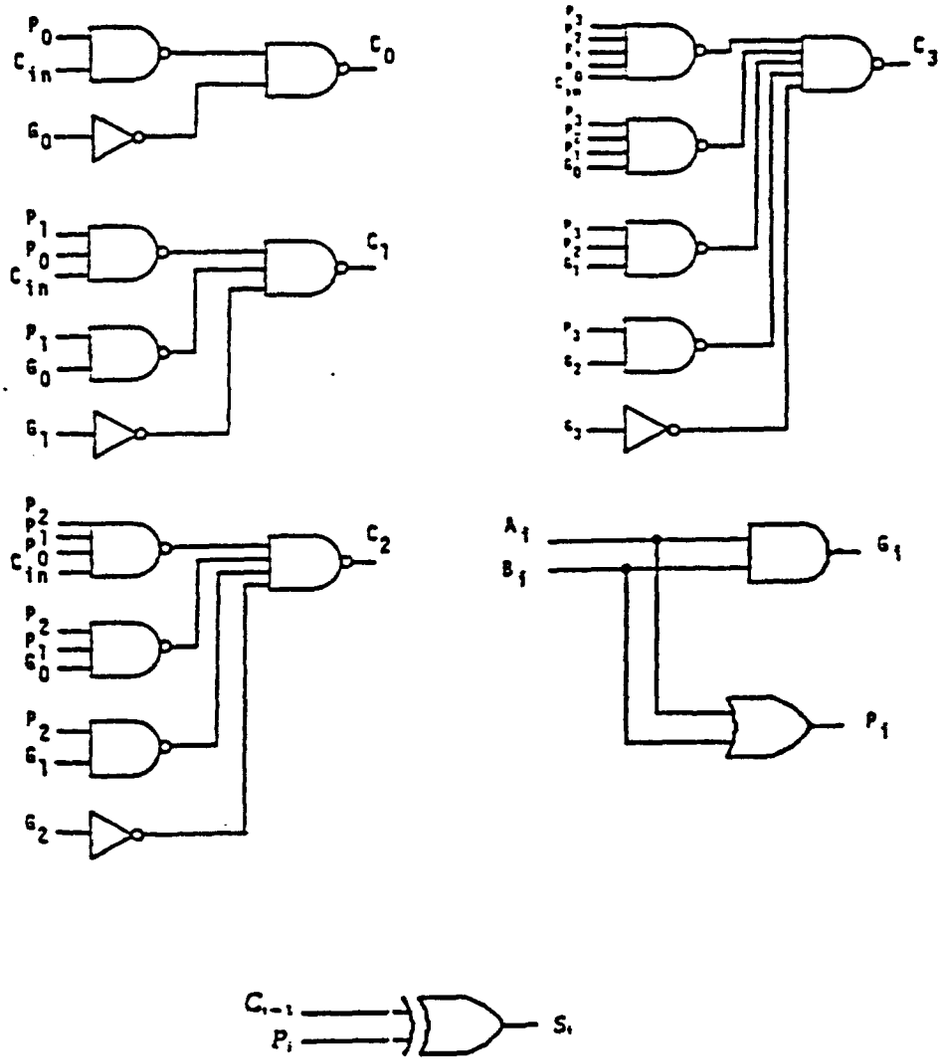


Fig. 2.8 The Schematic Logic of 4-bit Carry Look-Ahead Adder Units

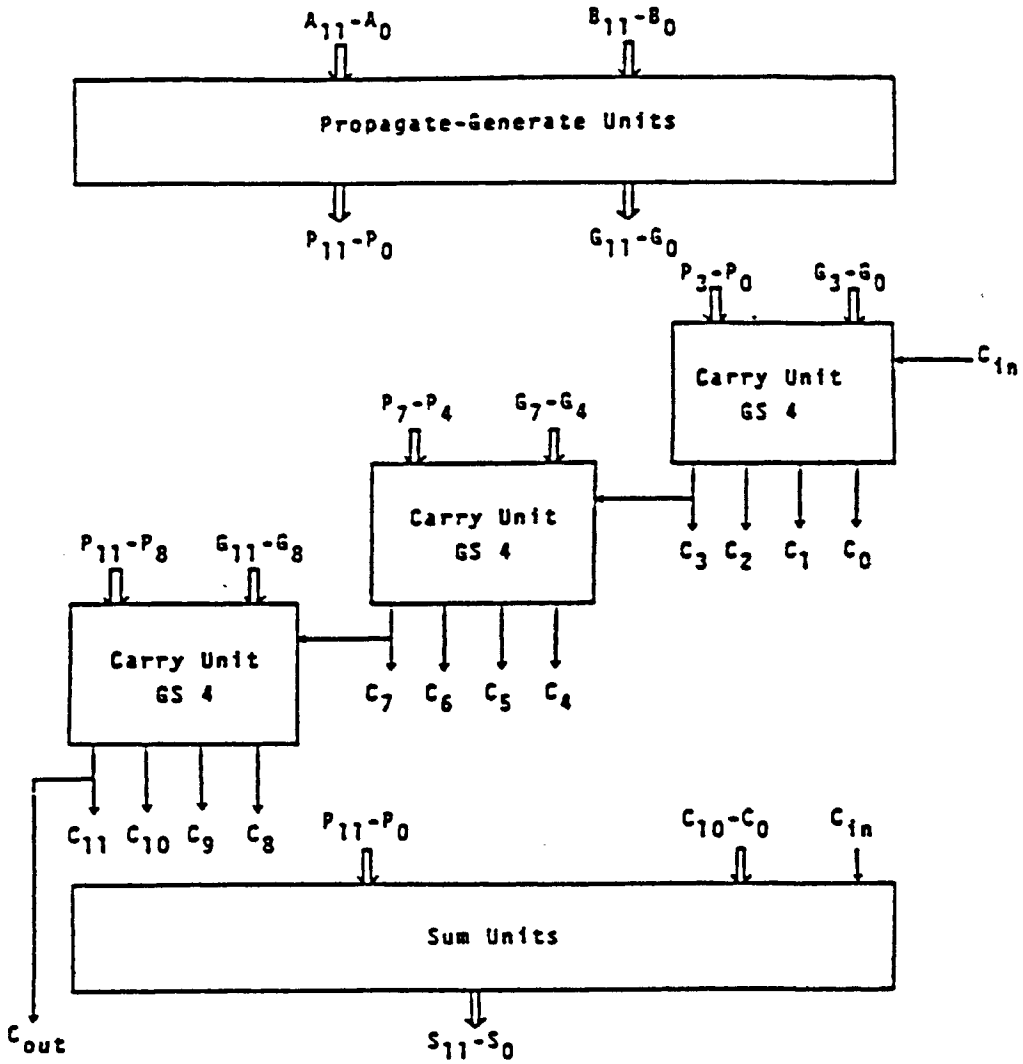


Fig. 2.9 The Organization of a 12-bit Ripple-Carry Look-Ahead Adder

2.3.3 FIRST ORDER CARRY LOOK-AHEAD ADDER

It was noted in the previous section that the delay time in ripple carry look-ahead adder depends on the size of the adder. Another technique called the first order carry look-ahead (FOCLA) adder will be explained [5]. This technique allows the propagation delay to be independent of the size of the adder, but requires more hardware. Let us re-examine the carry equations C3, C7, and C11 of the 12-bit ripple carry look-ahead adder

$$C3 = G3 + G2\ P3 + G1\ P2\ P3 + G0\ P1\ P2\ P3 + Cin\ P0\ P1\ P2\ P3$$

$$C7 = G7 + G6\ P7 + G5\ P6\ P7 + G4\ P5\ P6\ P7 + C3\ P4\ P5\ P6\ P7$$

$$C11 = G11 + G10\ P11 + G9\ P10\ P11 + G8\ P9\ P10\ P11 + C7\ P8\ P9\ P10\ P11$$

Now the question is, how can these carries be calculated simultaneously as was done for a simple CLA adder? We define a set of first order carry propagate/generate functions $P'i$, $G'i$, as follows:

$$P'0 = P0\ P1\ P2\ P3$$

$$G'0 = G3 + G2\ P3 + G1\ P2\ P3 + G0\ P1\ P2\ P3$$

$$P'1 = P4\ P5\ P6\ P7$$

$$G'1 = G7 + G6\ P7 + G5\ P6\ P7 + G4\ P5\ P6\ P7$$

$$P'2 = P8\ P9\ P10\ P11$$

$$G'2 = G11 + G10\ P11 + G9\ P10\ P11 + G8\ P9\ P10\ P11$$

Using these newly defined equations, then C3, C7, and C11 can be written as :

$$C3 = G'0 + Cin\ P'0$$

$$C7 = G'1 + G'0\ P'1 + Cin\ P'0\ P'1$$

$$C11 = \overline{G2} + \overline{G1}\ \overline{P2} + \overline{G0}\ \overline{P1}\ \overline{P2} + Cin\ \overline{P0}\ \overline{P1}\ \overline{P2}$$

From these equations all three carries can be generated simultaneously. The equations for a 12-bit FOCLA adder can be summarized as follows:

Propagate generate unit :

$$Pi = Ai \oplus Bi$$

$$Gi = Ai\ Bi \qquad i = 0, 1, 2, \ldots, 11$$

First order propagate/generate unit

$$\overline{P0} = P0\ P1\ P2\ P3$$

$$\overline{G0} = G3 + G2\ P3 + G1\ P2\ P3 + G0\ P1\ P2\ P3$$

$$\overline{P1} = P4\ P5\ P6\ P7$$

$$\overline{G1} = G7 + G6\ P7 + G5\ P6\ P7 + G4\ P5\ P6\ P7$$

$$\overline{P2} = P8\ P9\ P10\ P11$$

$$\overline{G2} = G11 + G10\ P11 + G9\ P10\ P11 + G8\ P9\ P10\ P11$$

First order carry unit :

$$C3 = \overline{G0} + Cin\ \overline{P0}$$

$$C7 = \overline{G1} + \overline{G0}\ \overline{P1} + Cin\ \overline{P0}\ \overline{P1}$$

$$C11 = \overline{G2} + \overline{G1}\ \overline{P2} + \overline{G0}\ \overline{P1}\ \overline{P2} + Cin\ \overline{P0}\ \overline{P1}\ \overline{P2}$$

Carry unit :

$$C0 = G0 + Cin\ P0$$

$$C1 = G1 + G0\ P1 + Cin\ P0\ P1$$

$$C2 = G2 + G1\ P2 + G0\ P1\ P2 + Cin\ P0\ P1\ P2$$

$$C4 = G4 + C3\ P4$$

$$C5 = G5 + G4\ P5 + C3\ P4\ P5$$

$$C6 = G6 + G5\ P6 + G4\ P5\ P6 + C3\ P4\ P5\ P6$$

$$C8 = G8 + C7\ P8$$

$$C9 = G9 + G8\ P9 + C7\ P8\ P9$$

$$C10 = G10 + G9\ P10 + G8\ P9\ P10 + C7\ P8\ P9\ P10$$

Summation unit :

$$Si = Pi \oplus Ci-1 \qquad i = 0, 1, 2, ..., 11$$

The block diagram of a 12-bit first order carry look-ahead adder is shown in Fig. 2.10 .

2.3.4 CARRY SELECT ADDER

The carry select adder is another high-speed addition technique which is of great value for the addition of large word length numbers [4]. In this system of addition, the addend and augend are broken into subaddend and subaugend sections that are added twice. The first addition with a 0 carry-in and the second with a 1 carry-in. These two sections produce two subsums. The correct value of carry-in selects the appropriate subsum (2-input multiplexer). The selection of the carry input to each section is generated in a carry selection unit. Let us consider a 16-bit carry select adder. First, the 16-bit word is divided into 4 sections each of size 4-bit. Each 4-bit section adder could either be a ripple-carry adder or a carry-lookahead adder. The logical expressions for carry selection circuits are:

$$C3 = Cin\ \overset{1}{C3} + \overset{0}{C3}$$

$$C7 = \overset{1}{C7}\ C3 + \overset{0}{C7}$$

$$= \overset{1}{C7}\ \overset{1}{C3}\ Cin + \overset{1}{C7}\ \overset{0}{C3} + \overset{0}{C7}$$

$$C11 = \overset{1}{C11}\ C7 + \overset{0}{C11}$$

$$= \overset{1}{C11}\ \overset{1}{C7}\ \overset{1}{C3}\ Cin + \overset{1}{C11}\ \overset{1}{C7}\ \overset{0}{C3} + \overset{1}{C11}\ \overset{0}{C7} + \overset{0}{C11}$$

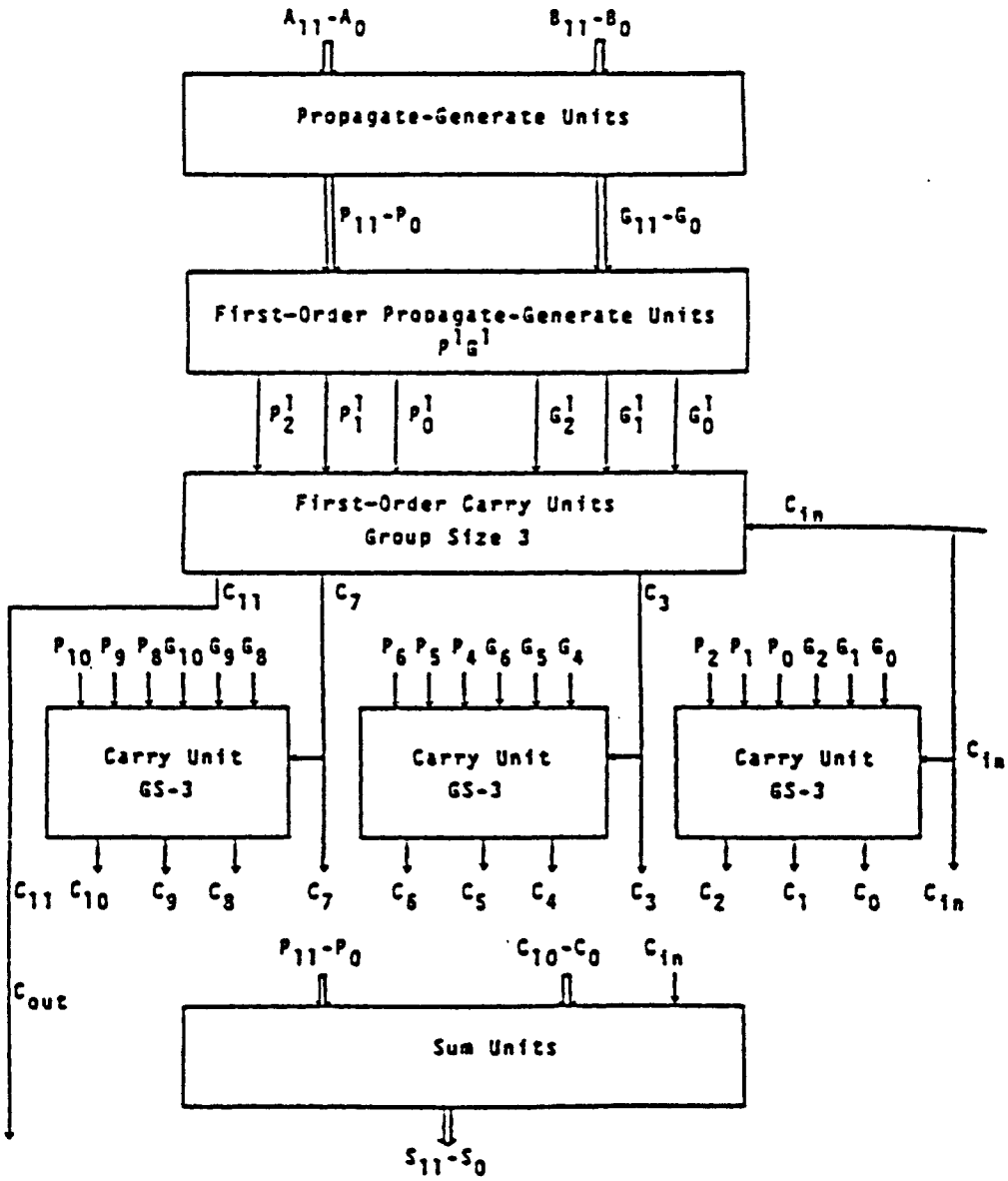


Fig. 2.10 The Functional Block Diagram of 12-bit First-Order Carry Look-Ahead Adder

Figure 2.11 shows the basic scheme of the 16-bit carry select adder. The superscript "0" and "1" shows the carry obtained from the sections of "zero" and "one" forced carry-in respectively.

2.4 PARALLEL MULTIPLIER

The multiplication of two n-bit integers, A and B, will create a 2n-bit product, P = A x B. Where A is called the multiplicand and B is the multiplier. Consider two unsigned binary integers :

A = am-1 .... a1a0

B = bn-1 .... b1b0

With values Av and Bv respectively

$$Av = \sum_{i=0}^{m-1} (a_i \, 2^i)$$

$$Bv = \sum_{i=0}^{n-1} (b_i \, 2^i)$$

In binary, the product of A and B results in a (m+n)-bit number P.

$$Pv = Av \, Bv$$

$$= [\sum_{i=0}^{m-1} (a_i \, 2^i)][\sum_{j=0}^{n-1} (b_j \, 2^j)] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [(a_i \, b_j) \, 2^{i+j}]$$

$$= \sum_{k=0}^{m+n-1} (p_k \, 2^k)$$

All partial product terms (ai bj) are generated in parallel by mxn AND gates. The design of an 8x8-bit unsigned array multiplier is selected to demonstrate the parallel

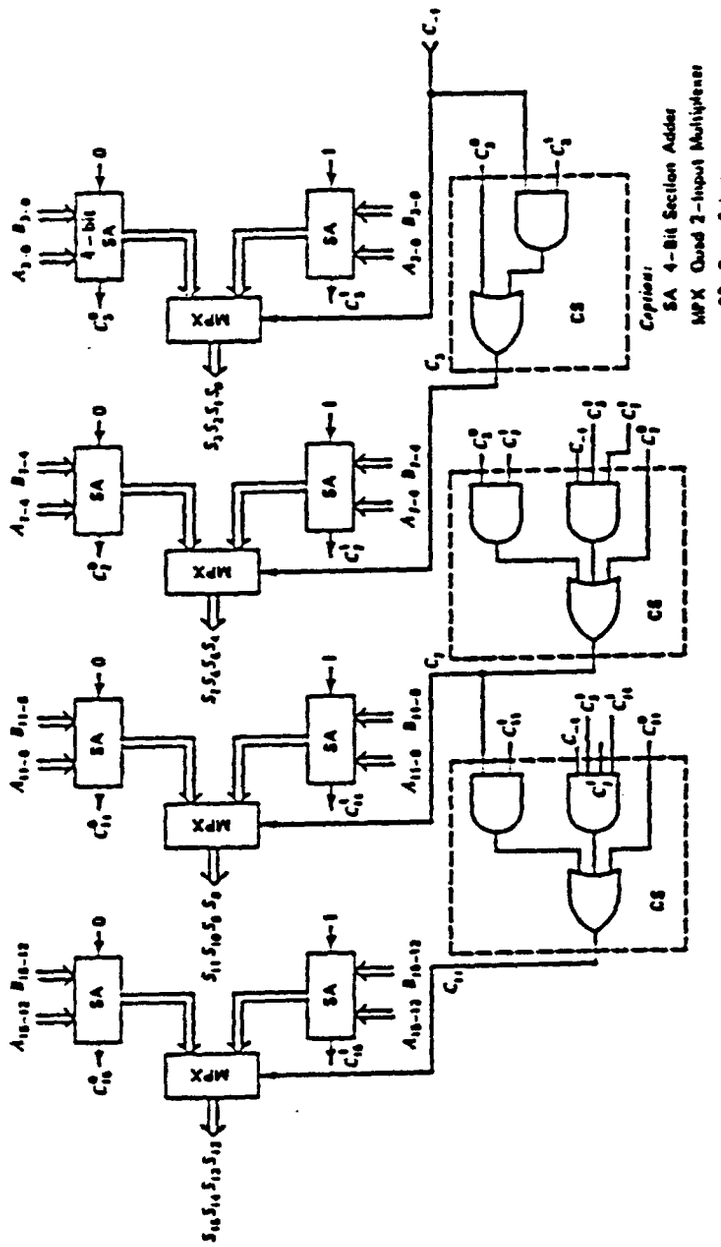


Fig. 2.11 The Organization of a 16-bit Carry-Select Adder

multiplication architecture using carry-save adders. The advantage of this approach is that design issues are minimal and layout is highly modular. The logic diagram of an 8x8-bit parallel multiplier is shown in Fig. 2.12 [6]. The implementation of such a multiplier of size nxn-bits requires (n-1)(n-1)-1 Full adders, n Half adders, and nxn AND gates. Therefore the entire multiplier is designed using three unit cells: a full adder, a half adder, and an AND gate cell. The last stage of the array multiplier is an (n-1)-bit ripple-carry adder which can be replaced by a carry lookahead adder to improve the speed of the operation.

2.5 PIPELINED ARITHMETIC

2.5.1 THE CONCEPT OF PIPELINE SYSTEM

The computing power of a machine is determined by two major factors, one is throughput rate (bandwidth) and the other is latency [4]. Throughput is the rate at which new data can be fed to the processor. Latency is the time required for the data to traverse the processor. For a system that performs one operation at a time (such as the adders that have been discussed so far), throughput rate is the inverse of latency.

In most digital signal processing applications the throughput rate of the processor is the critical factor, not the latency time. In conventional designs, an increase in the throughput rate of arithmetic processors has been achieved by reducing the latency with faster logic circuitry. For example, high-speed adders have been designed

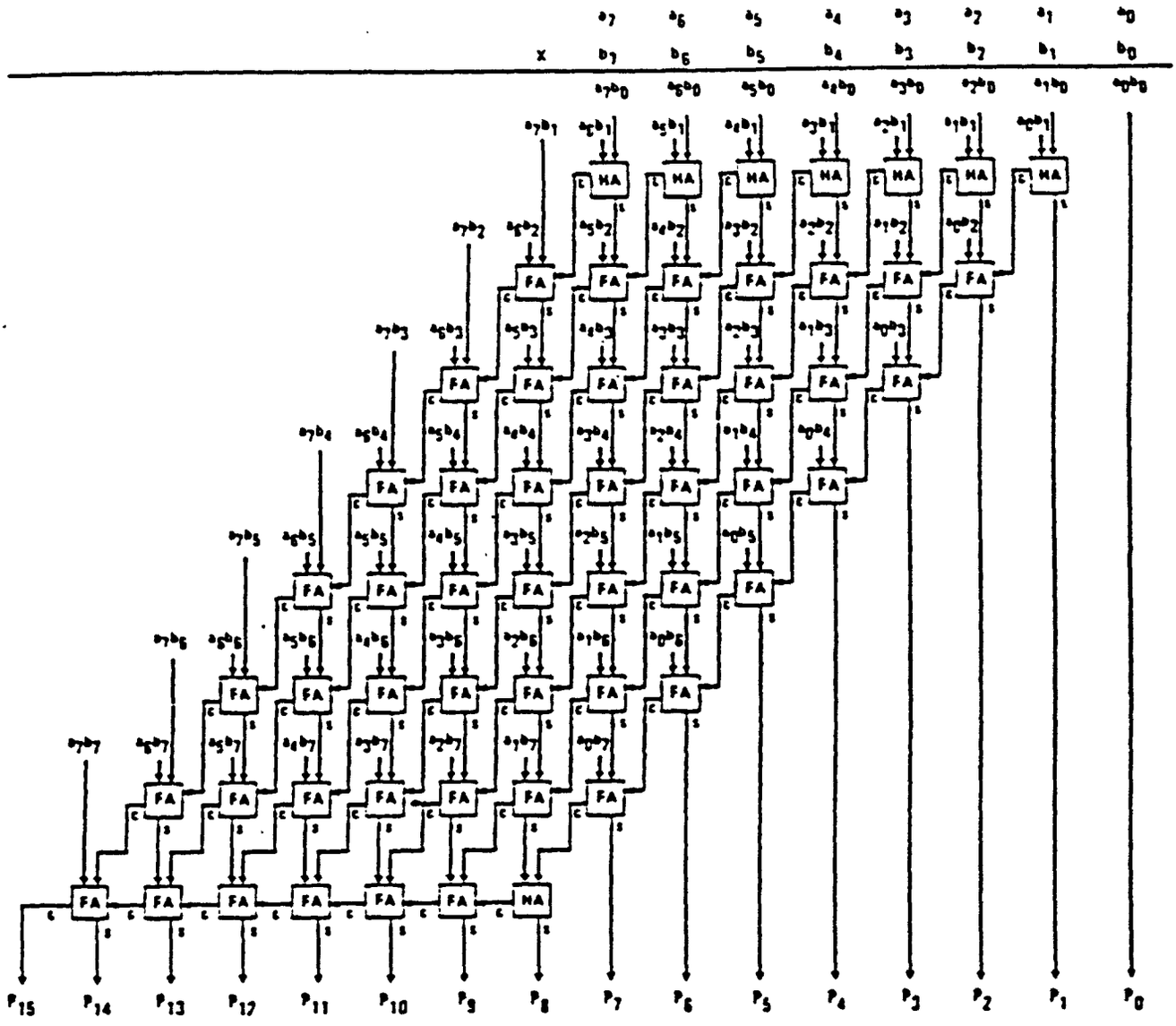


Fig. 2.12 The schematic Block Diagram of an 8x8-bit Parallel Multiplier

to reduce the propagation delay time.

A simple solution to the throughput rate problem is to allow simultaneous execution of many tasks by multiple arithmetic units. Parallel processing with straight hardware duplication, however, may not be economical or cost effective. Pipelining allows a significant increase in throughput rate with only moderate increase in hardware. Pipeline computing refers to the subdivisions of the total computational workload into individual tasks, so that they can be executed in an overlapped fashion through a high-speed arithmetic pipeline under certain precedence constraints. The functional organization of a pipelined arithmetic unit is illustrated in Fig. 2.13 . The pipeline is characterized by the succession of stages. In a pipelined structure, successive stages are interfaced with data latches (synchronized registers), which hold the input and output bit pattern of the successive stages. Different stages may have different delay times. In order to regulate the pipeline operation, the synchronizing clock pulse should have a clock period of

$$T > \text{Max} \{ Ti + Tl \} \qquad i \in [1 , k]$$

Where Ti is the delay time of the i-th stage and Tl is the delay time of a single latch. Therefore the throughput rate of the pipelined system is determined by the maximum delay of one stage plus the delay time of a latch. The latency of such a system is k clock cycles.

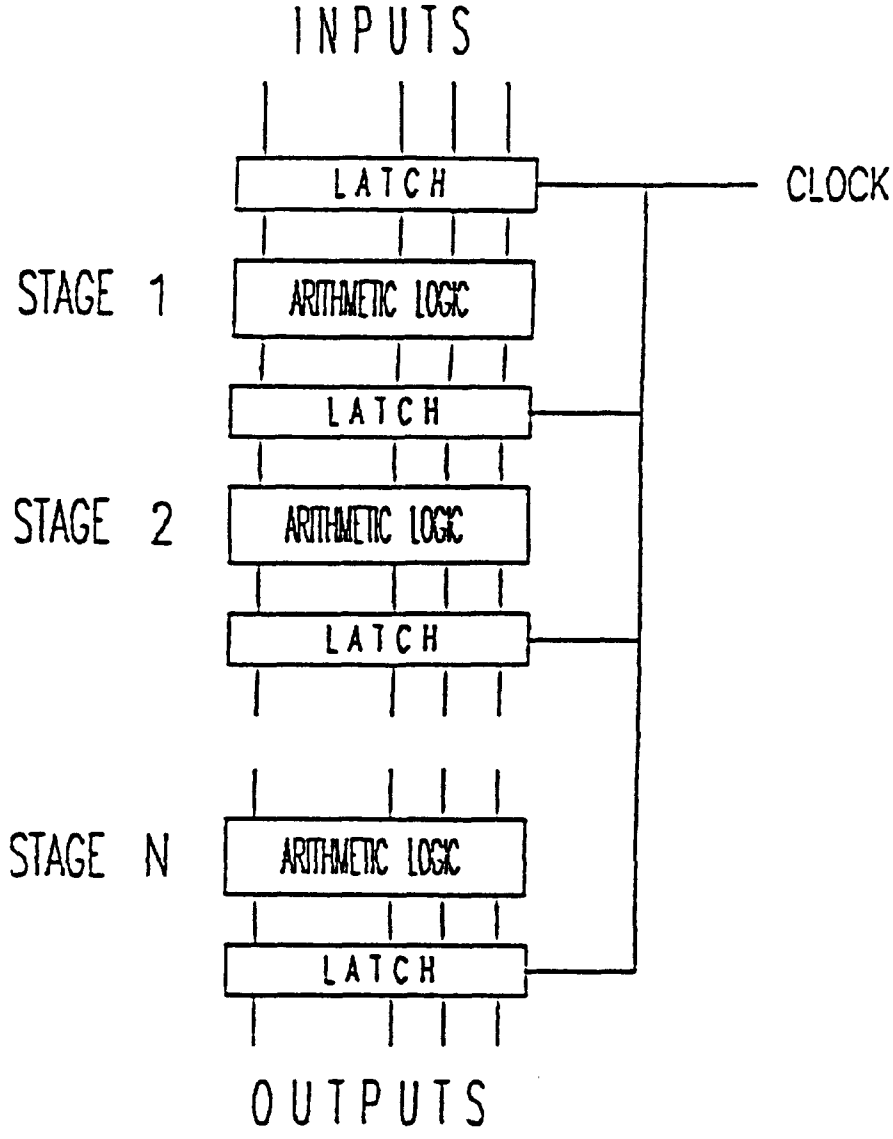


Fig. 2.13 The Functional Organization of an Arithmetic Pipeline with k Stages

2.5.2 PIPELINED CARRY-SAVE ADDER (HALF ADDER ARRAY)

Pipelined adders remove the problem of carry propagation at the cost of extra hardware. Fig. 2.14 shows the architecture of a fully pipelined adder using an array of half adders. The pipelining is achieved by placing latches between each stage of operation. In this way the inputs can arrive separated by only one clock cycle. The maximum clock frequency of operation is determined by the delay of a half adder plus the delay of a single latch.

2.5.3 PIPELINED RIPPLE-CARRY ADDER (FULL ADDER ARRAY)

The general architecture of a pipelined ripple-carry adder (FA array) is shown in Fig. 2.15. This design requires less hardware than the similar structure using an array of half adders (almost 25% reduction in transistor count). As before, throughput rate of this structure is determined by the delay time of a full adder plus the delay of a latch.
A 10-bit pipelined ripple-carry adder was implemented using 3um CMOS technology. Appendix B (Fig. B-5) shows the chip layout of this adder.

2.5.4 PIPELINED CARRY LOOKAHEAD ADDER

In section 2.3.3, the organization of the first order (two-level) carry lookahead adder was discussed. This adder can be converted to pipelined version by placing latches between the different stages. Figure 2.16 shows the architecture of a 16-bit pipelined carry lookahead adder. The adder has five stages:

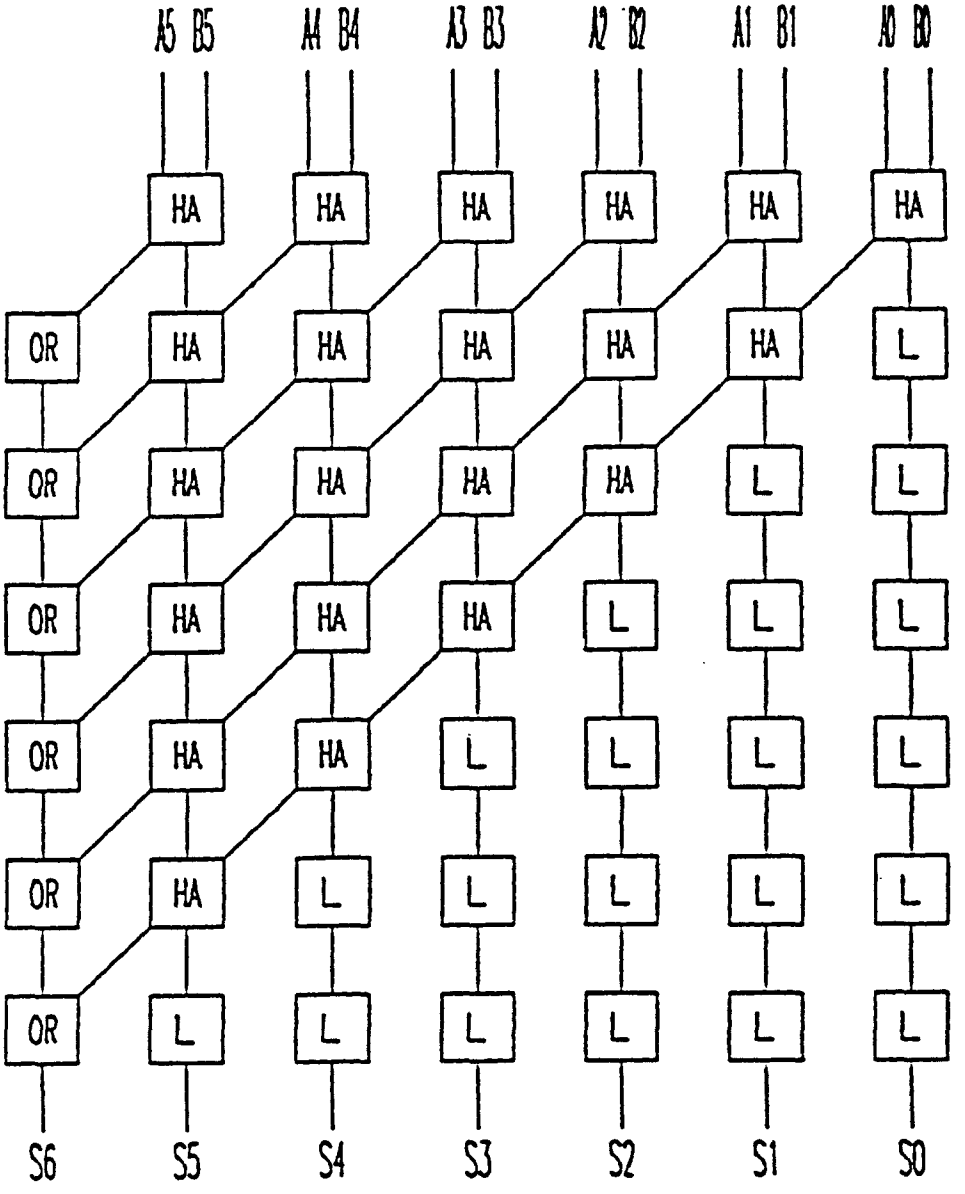


Fig. 2.14 Block Diagram of a 6+6 -bit Pipelined Binary Adder (HA Array)

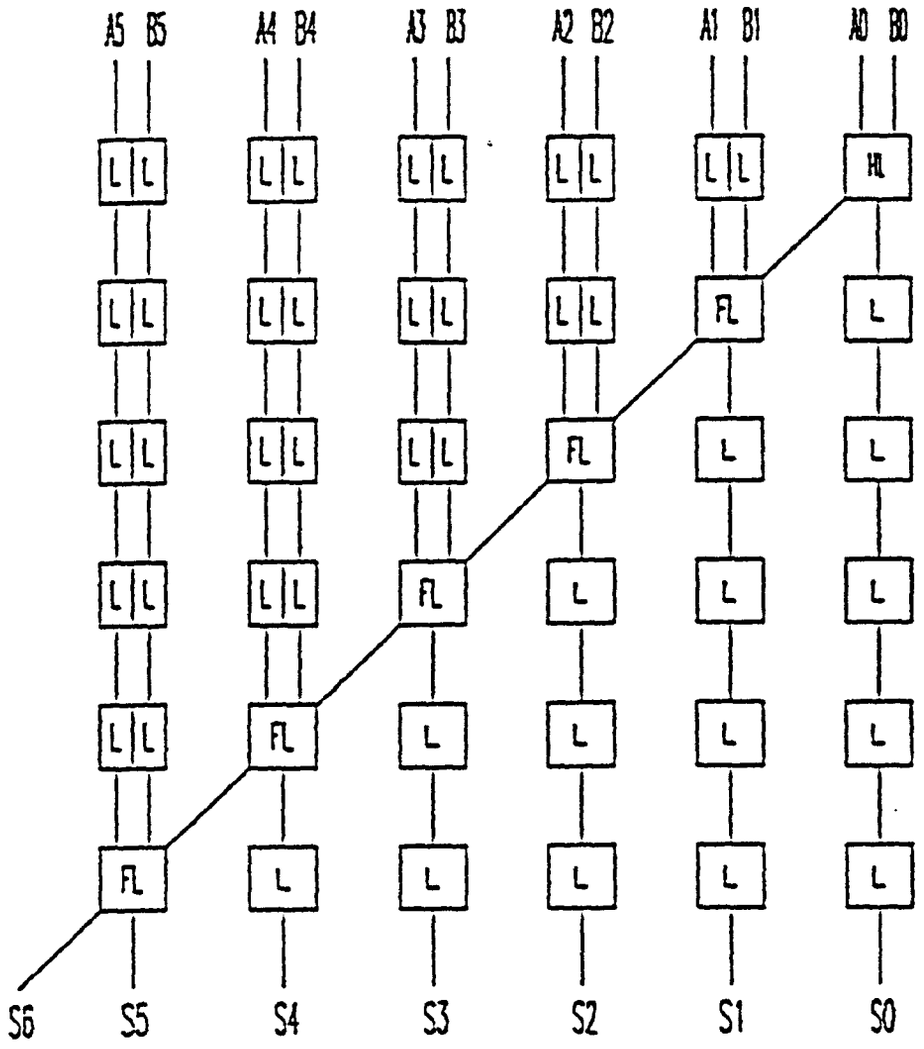


Fig. 2.15 Block Diagram of a 6+6 -bit Pipelined Binary Adder (FA Array)

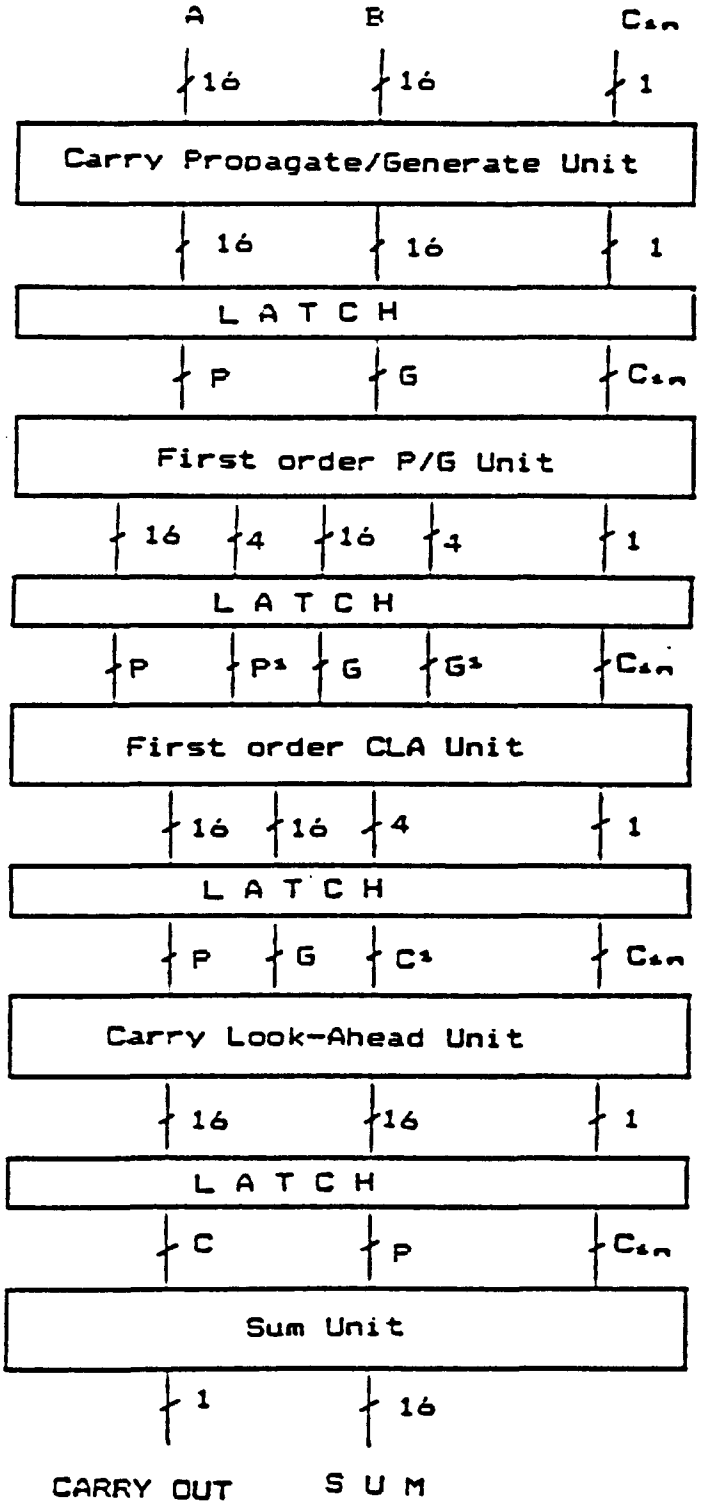


Fig. 2.16 Architecture of a 16-bit Pipelined Carry Look-Ahead Adder

Stage 1 - Propagate / Generate Unit:

Each pair of input bits (A, B) is used to produce the generate (G) and propagate (P) bits according to the following equations

$P = A \oplus B$

$G = A \cdot B$

Stage 2 - First Order Propagate / Generate Unit:

Each group of four generate and propagate signals from the previous stage are used to produce the first order propagate and generate vectors according to the following equations:

$P'0 = P0\ P1\ P2\ P3$

$G'0 = G3 + G2\ P3 + G1\ P2\ P3 + G0\ P1\ P2\ P3$

$P'1 = P4\ P5\ P6\ P7$

$G'1 = G7 + G6\ P7 + G5\ P6\ P7 + G4\ P5\ P6\ P7$

$P'2 = P8\ P9\ P10\ P11$

$G'2 = G11 + G10\ P11 + G9\ P10\ P11 + G8\ P9\ P10\ P11$

$P'3 = P12\ P13\ P14\ P15$

$G'3 = G15 + G14\ P15 + G13\ P14\ P15 + G12\ P13\ P14\ P15$

Stage 3 - First order Carry Lookahead Unit:

First order propagate / generate vectors are used with the carry input Cin to produce the intermodular carry signals according to the following equations :

$C3 = G'0 + Cin\ P'0$

$C7 = G'1 + G'0\ P'1 + Cin\ P'0\ P'1$

$C11 = G'2 + G'1\ P'2 + G'0\ P'1\ P'2 + Cin\ P'0\ P'1\ P'2$

$C15 = G'3 + G'2\ P'3 + G'1\ P'2\ P'3 + G'0\ P'1\ P'2\ P'3 + Cin\ P'0\ P'1\ P'2\ P'3$

Stage 4 - Carry Lookahead Unit :

The final carries are produced in carry lookahead unit

according to the following equations

$C0 = G0 + Cin\ P0$

$C1 = G1 + G0\ P1 + Cin\ P0\ P1$

$C2 = G2 + G1\ P2 + G0\ P1\ P2 + Cin\ P0\ P1\ P2$

$C4 = G4 + C3\ P4$

$C5 = G5 + G4\ P5 + C3\ P4\ P5$

$C6 = G6 + G5\ P6 + G4\ P5\ P6 + C3\ P4\ P5\ P6$

$C8 = G8 + C7\ P8$

$C9 = G9 + G8\ P9 + C7\ P8\ P9$

$C10 = G10 + G9\ P10 + G8\ P9\ P10 + C7\ P8\ P9\ P10$

$C12 = G12 + C11\ P12$

$C13 = G13 + G12\ P13 + C11\ P12\ P13$

$C14 = G14 + G13\ P14 + G12\ P13\ P14 + C11\ P12\ P13\ P14$

Stage 5 - Sum Unit

In the sum unit, the carries are combined with the generate signals to produce the sum bits

$Si = Pi \oplus Ci-1$

As before, pipelined registers are used between the stages. The pipelined carry lookahead adder has the advantage of smaller latency time compared to the pipelined ripple carry adder. For a 16-bit pipelined ripple-carry adder, 16 clock cycles are required before the first result is output while the pipelined carry lookahead adder requires only 5 clock cycles. The disadvantage of the pipelined carry lookahead adder is its irregular structure compared to the pipelined ripple -carry adder (HA & FA array). The throughput rates of both adders are almost the same.

2.5.5 PIPELINED MULTIPLIER

Most digital signal processing algorithms require the use of high-speed multipliers. In applications where a multiplier is operating on a input data stream, a pipelined multiplier is used to improve the throughput rate of the operations. The maximum clock rate of a pipelined multiplier is determined by the delay due to one stage, which is equal to the delay of a full adder plus a register. Fig. 2.17 shows the schematic block diagram of a multiplier array using a carry-save adder (CSA) technique [3]. It is readily observed that data flows vertically from one stage to the next. The registers, which are placed at the outputs of each individual cells, make the architecture fully pipelined. This method results in a high-throughput rate operation. Further improvement in speed is still possible by replacing each full adder by two half-adders and then pipelining the half adders. In this way the throughput rate is determined by the half-adder delay plus a register. Since the chip area will increase considerably the area-time product of this design will be higher.

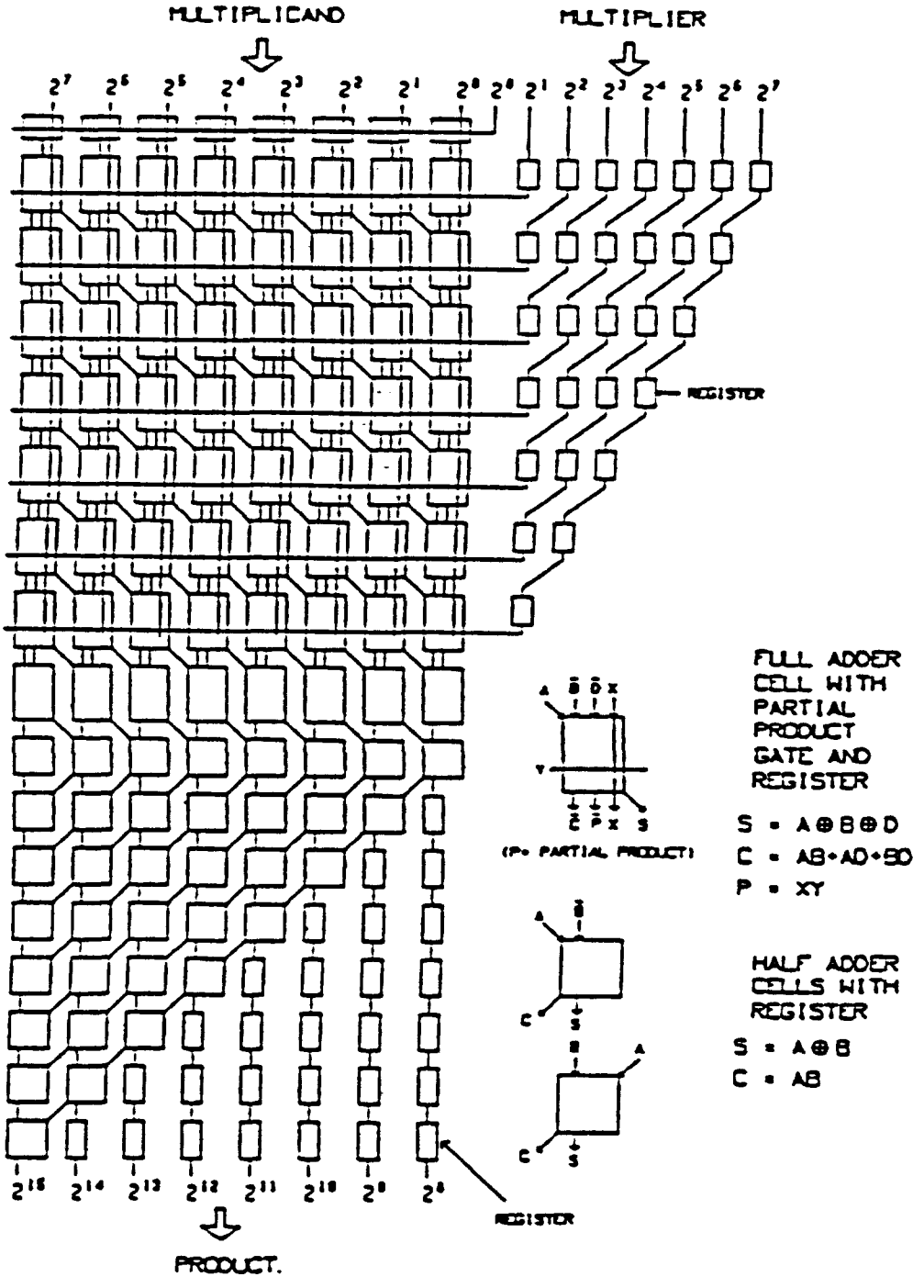


Fig. 2.17 Block Diagram of an 8x8-bit Parallel Pipelined Multiplier

CHAPTER 3

RESIDUE ARITHMETIC AND DESIGN OF HIGH-SPEED RNS ADDER

3.1 INTRODUCTION

This chapter describes the representation of a number in the Residue Number System (RNS), and will also discuss the operations of addition, subtraction, and multiplication using the RNS representation. VLSI implementations of RNS adders are discussed and four different realization approaches are presented: Binary adder, look-up table, hybrid, and counter-based method. In the final part of this chapter a pipelined Binary ripple-carry adder is compared with a pipelined RNS adder (ripple-carry-based) and the advantages of the RNS adder over the Binary adder are shown.

3.2 RESIDUE REPRESENTATION

The Residue Number System is a nonweighted integer number system which is described by an N-tuple of integers [7] :

$$m_1, m_2, m_3, \ldots\ldots\ldots\ldots, m_N$$

called moduli (each number is called a modulus). The representation of an integer X in RNS takes the form of an N-tuple of residues,

$$X = \{ x_1, x_2, x_3, \ldots\ldots\ldots, x_N \}$$

Where x_i is the remainder of X when divided by m_i. The following set of N equations define x_i

$$X = q_i m_i + x_i \qquad i = 1,2,3,\ldots\ldots\ldots, N$$

By definition x must be positive and less than m . So qi is the integer value of the quotient X/mi which is denoted by [X/mi]. The quantity xi is designated as residue of X modulo mi (also known as X mod mi) or ! X !mi . Therefore we can write

$$X = mi\ [\ X/mi\ ] + xi$$

The dynamic range of X is represented by

$$0 \leq X < M$$

$$M = m1.m2. \ldots . mN$$

The residue representation of a number is unique but the converse of the statement is true only if :

(1) The set of moduli are pairwise relative prime

(2) Only numbers within the dynamic range are considered

Example 1

Consider a Residue Number System with moduli m1 = 3 and m2 = 5. What is the residue representation of X = 13 ?

For m1 = 3 we obtain [13/3] = 4 and for modulus 5 the integer value is [13/5] = 2 . Hence

$$x1 = X - m1\ [X/m1] = 13 - 3 * 4 = 1$$

$$x2 = X - m2\ [X/m2] = 13 - 5 * 2 = 3$$

Therefore the residue representation of 15 is (1 , 3)

Table 3.1 shows the residue representation of the numbers -4 to 20 for moduli 3 , 5 .

Examination of table 3.1 shows that residue representation is periodic (in this case the period is 15). If the Residue Number System is restricted to a single period, ambiguity in this system will be prevented.

Integer	Residue Digits (moduli 3, 5)	Integer	Residue Digits (moduli 3, 5)
-4	2, 1	9	0, 4
-3	0, 2	10	1, 0
-2	1, 3	11	2, 1
-1	2, 4	12	0, 2
0	0, 0	13	1, 3
1	1, 1	14	2, 4
2	2, 2	15	0, 0
3	0, 3	16	1, 1
4	1, 4	17	2, 2
5	2, 0	18	0, 2
6	0, 1	19	1, 4
7	1, 2	20	2, 0
8	2, 3		

Table 3.1

3.3 RESIDUE ARITHMETIC

For the residue number system with moduli m1, m2, ..., mN, let X and Y be represented by residue digits.

$$X \quad <-----> \quad \{ x1, x2, ..., xN \}$$

$$Y \quad <-----> \quad \{ y1, y2, ..., yN \}$$

Addition, subtraction, and multiplication of residue numbers X and Y are given by

$$|X + Y|_M <-----> \{ |x1 + y1|_{m1}, |x2 + y2|_{m2}, \ldots |xN + yN|_{mN} \}$$

$$|X.Y|_M <-----> \{ |x1.y1|_{m1}, |x2.y2|_{m2}, \ldots |xN.yN|_{mN} \}$$

The above equations imply that addition, subtraction, and multiplication in residue number system can be performed completely independently and, hence, in parallel on each residue digit. The absence of a carry (or borrow) between

digits in residue arithmetic gives an inherent speed advantage to the system.

Note that in the above equations each operation (addition, subtraction, and multiplication) is obtained modulo M. Hence if, for example, sum X + Y exceeds M, an ambiguity will arise.

Example 3.2 :

For the moduli 3 and 5, add, subtract, and multiply X = 2 and Y = 7.

Solution:

```
  Moduli                    3     5
-------------------------------------
    7   <---------->   { 1  ,  2 }
  + 2   <---------->   { 2  ,  2 }
  ----                 ------------
  ! 9 !  = 9  <--->    { 0  ,  4 }
     !15
```

```
  Moduli                    3     5
-------------------------------------
    7   <---------->   { 1  ,  2 }
  - 2   <---------->   { 1  ,  3 }
 -----                 ------------
  ! 5 !  = 5  <---->   { 2  ,  0 }
     !15
```

```
  Moduli                    3      5
--------------------------------------
    7   <---------->   { 1  ,  2 }
  X 2   <---------->   { 2  ,  2 }
 ------               --------------
  ! 14 !  = 14  <--->  { 2  ,  4 }
     !15
```

The validity of these operations may be checked by looking at the table 3.1.

3.4 VLSI IMPLEMENTATION OF RNS ADDERS

In this section four different realization approaches for modulo m adders are analyzed. The implementation methods are as follows:

I - Binary-based (random logic) RNS adders

II- Look-up table implementation

III- Hybrid method

IV - Counter-based RNS adder

3.4.1 BINARY-BASED (RANDOM LOGIC) RNS ADDER

To achieve a high throughput rate, the RNS adder is implemented with combinational logic. The architecture of a general RNS adder is shown in figure 3.1 [2]. Two binary adders are used to perform the RNS addition. The first adder computes S1 = A + B and the second adder computes S2 = A + B - M. The carry overflow which results from the second adder indicates whether S1 is the correct answer or S2. If carry overflow is zero the correct answer is S1, otherwise S2 is the right answer. Any of the types of high-speed binary adders, discussed in chapter 2, can be used for the implementation of this residue adder. The maximum throughput rate is achieved by using pipelined ripple-carry or carry look-ahead adder structures. In section 3.5 we give details of the architecture of the pipelined ripple-carry-based RNS adder.

3.4.2 LOOK-UP TABLE RNS ADDER

This method is based on storing the RNS addition in a look-

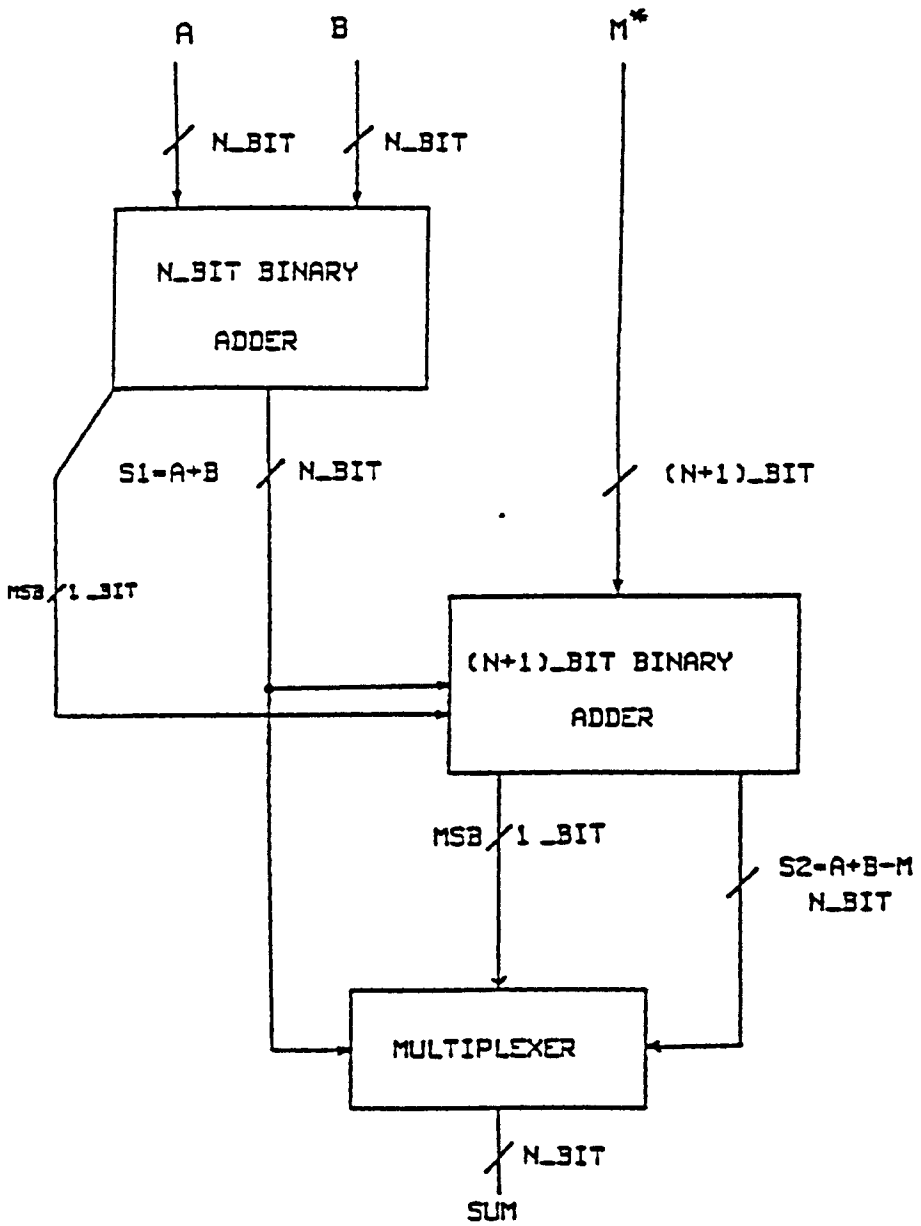


Fig. 3.1 Modulo m RNS Adder

A: ADDEND

B: AUGEND

M*: 2'S COMPLEMENT OF THE MODULUS

up table. The total number of memory locations required for a word length of k-bits is $k(2^{2k})$ entries. For large moduli, a large capacity memory is required in this approach and this limits the speed of operation. One of the advantages of the look-up table method is the large savings in hardware for fixed operands. Since constants can be added, subtracted, or multiplied simply by changing the RNS addition table, these are often referred to as "free operations" [8].

3.4.3 HYBRID METHOD

This approach combines both random logic binary adder modules and look-up table modules [9]. Fig 3.2 shows an RNS adder using the hybrid method. It consists of a k-bit binary adder and a (k+1)-bit address ROM. The ROM is used to correct the output of the binary adder and to provide free operations with constants.

3.4.4 COUNTER-BASED RNS ADDER

In this approach, residue addition is performed by a ring counter [10]. The schematic diagram of the adder is shown in Fig. 3.3. The adder consists of a counter-pulse generator, a sum counter, and a buffer counter. The addition time of two m-bit numbers is performed in 2m clock cycles. The ring counter is constructed by D-type flip-flops cascaded in a feed-back shift register form. The number of flip-flops is equal to m (modulus). It is obvious that X mod m is obtained by shifting the data X times. The time chart of the adder is shown in Fig 3.4, for the operation 16 + 27.

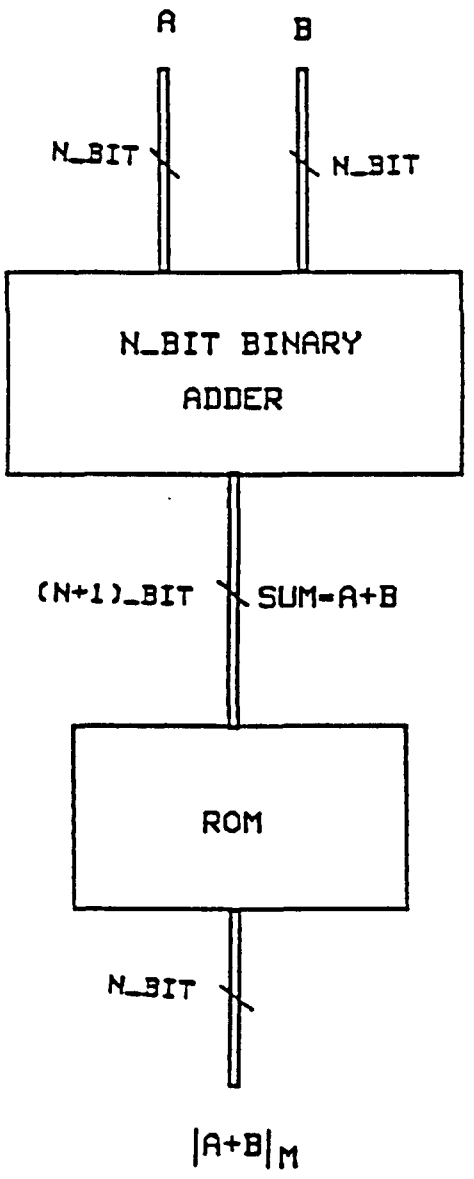


Fig. 3.2 An RNS Adder Using the Hybrid Approach

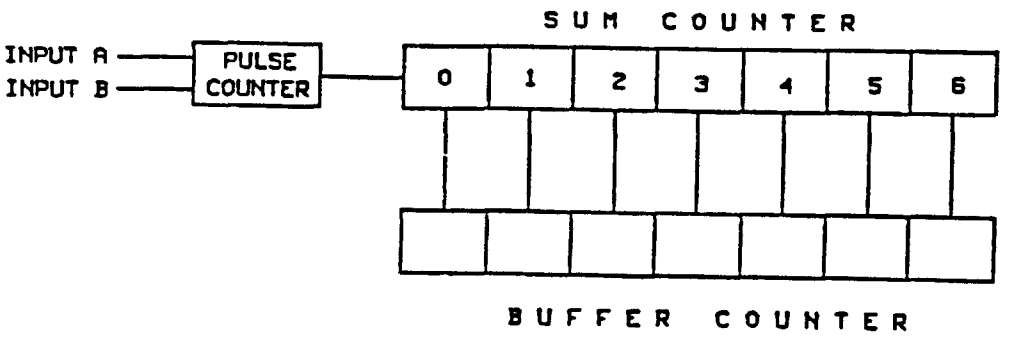


Fig. 3.3 A Counter-Based RNS Adder

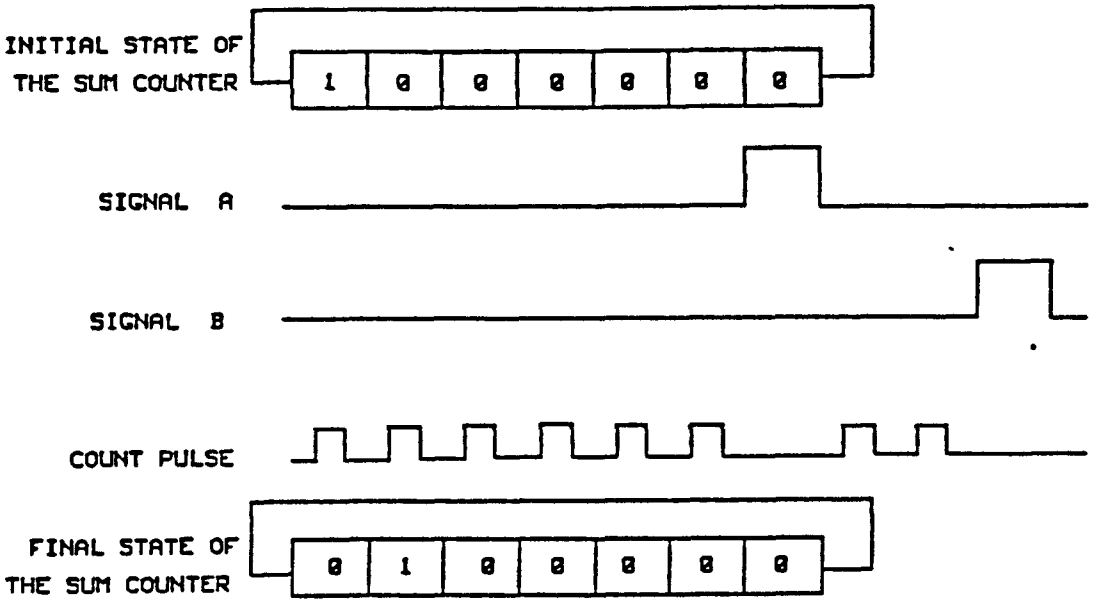


Fig. 3.4 Time Chart of the RNS Adder of Fig. 3.3

3.5 PIPELINED RIPPLE-CARRY RNS ADDER

The schematic block diagram of the pipelined ripple-carry-based modulo-M adder is shown in Fig. 3.5 . The carry save adder technique is utilized to obtain the maximum throughput rate which reduces to the delay of a single full adder plus a register. The two-operand pipelined adder performs the binary addition of the two numbers S1 = A + B. The three-operand pipelined adder performs the binary addition of A, B, and Mc, where Mc is the 2's complement of the base (M). The carry overflow (Co) which is generated by the three-operand adder is used as a control signal to a 2N-->N multiplexer to enable the selection of the correct output.

Higher throughput rate is obtained by using half adders in the design of pipelined RNS adder. Figure 3.6 shows the schematic diagram of such an adder.

A 5-bit pipelined RNS adder (FA array) has been implemented using 3um CMOS technology. Appendix B shows the chip layout of this adder.

3.6 COMPARISON OF THE PIPELINED RIPPLE-CARRY BINARY AND RNS ADDERS

In this section it will be shown that for a wide-dynamic range of operation, the RNS adder offers advantages over the binary adder in four areas: hardware complexity; speed; testability; latency time. The factor linking these advantages is that RNS arithmetic permits a wide-dynamic arithmetic range to be broken into several independent small-dynamic ranges.

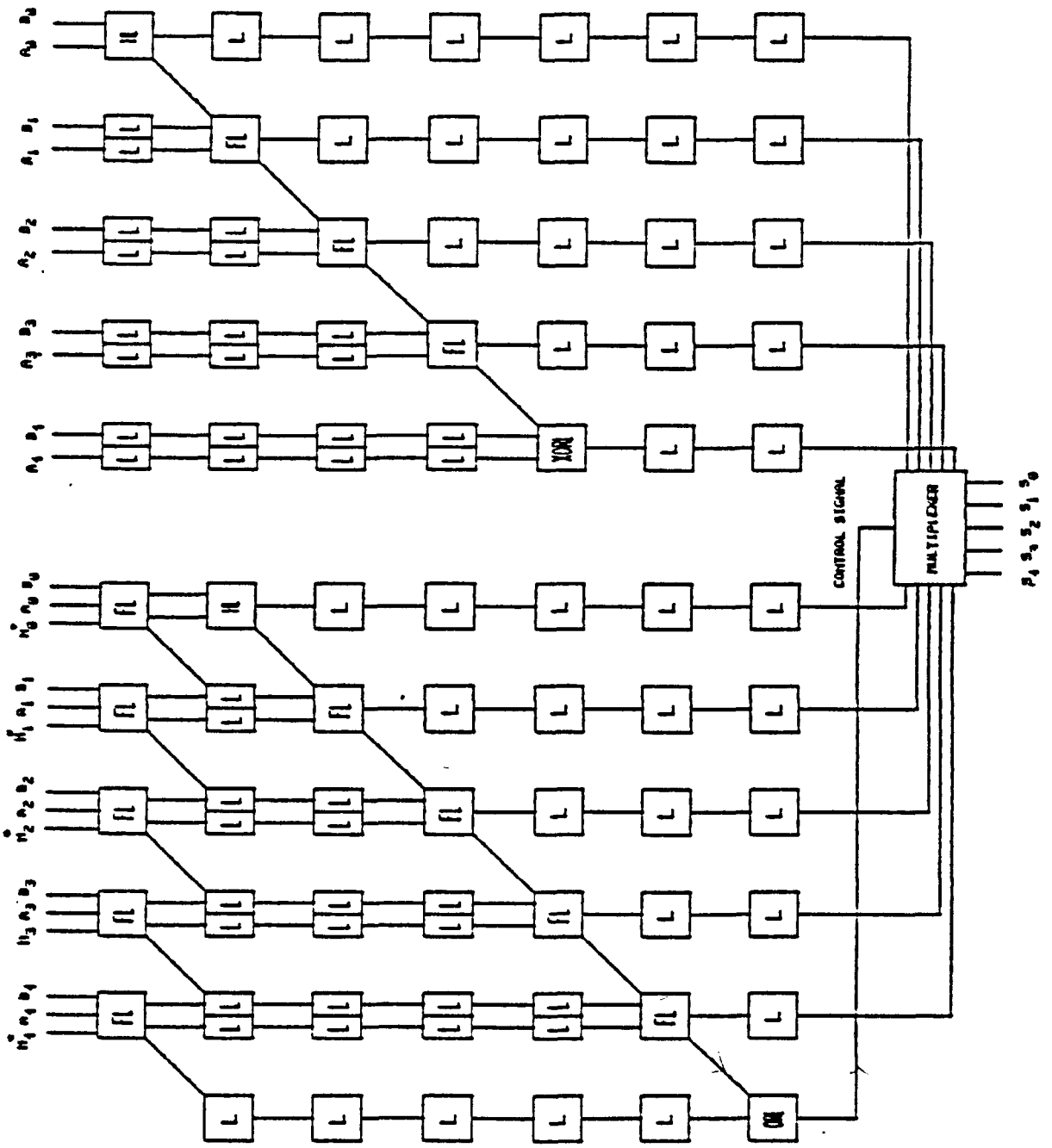


Fig. 3.5 A 5-bit Pipelined Ripple-Carry RNS Adder (FA Array)

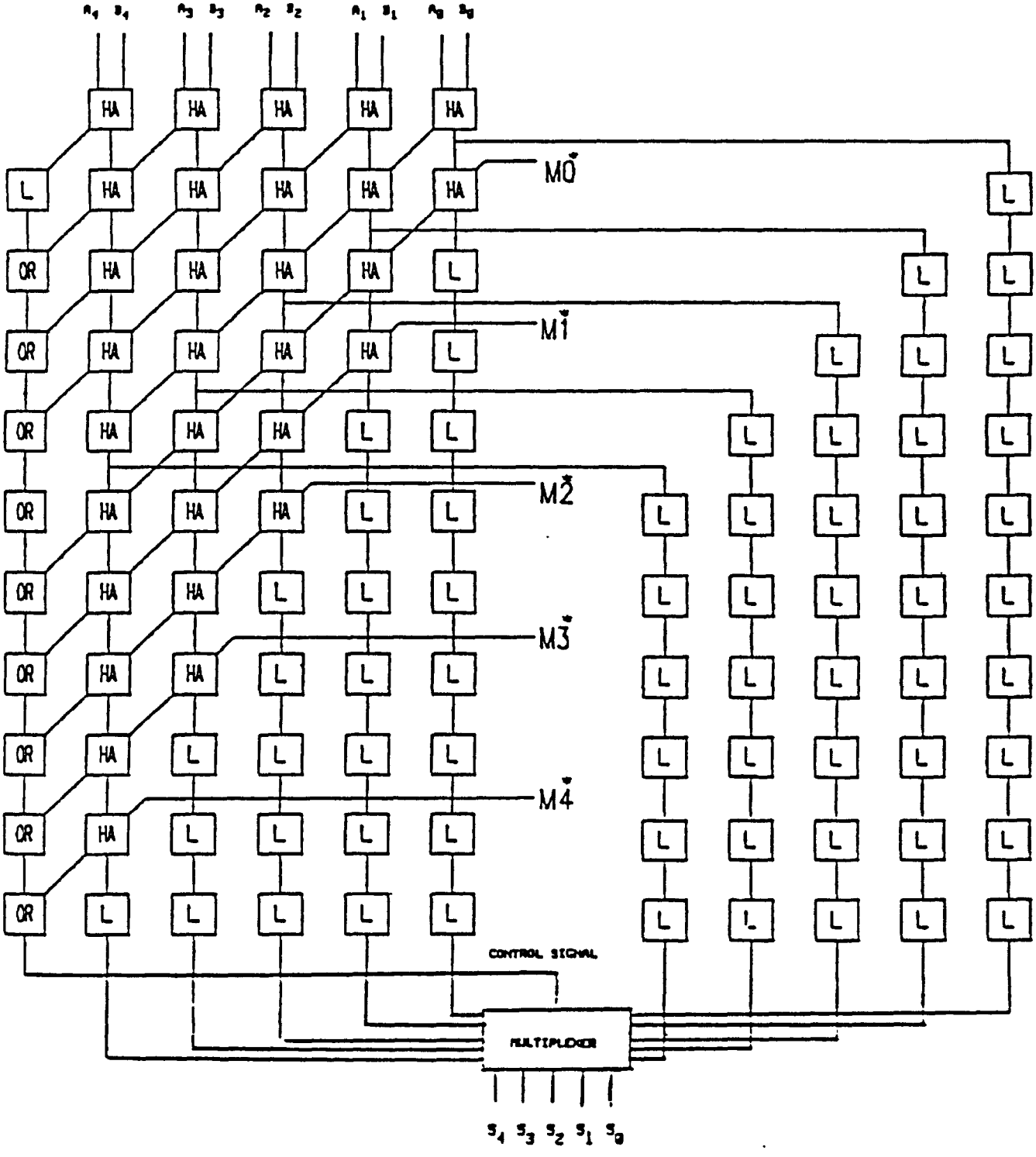


Fig. 3.6 A 5-bit Pipelined Carry Save RNS Adder (HA Array)

3.6.1 HARDWARE COMPLEXITY

Consider the pipelined ripple-carry Binary adder illustrated in Fig. 2.15. It is assumed that each cell is realized as a combinational logic circuit. The system parameters are defined as:

N : Dynamic range of the adder (in bits)

Nfa : The total number of Full adders

Nl : The total number of latches

Tfa : Delay time of the Full adder cell

Tl : Delay time of the latch

Tlat.: Latency time of the system

Fclk : maximum clock frequency

For an N-bit pipelined ripple-carry binary adder we need

$Nfa = N$

$Nl = N(3N + 1)/2$

As we can see from these equations the number of the latches is proportional to N^2. For large values of N, the total number of latches will be very large. One possible way of reducing the number of the latches is to partition N such that

$N < n1 + n2 + n3 + \ldots + nk$

This is a natural effect of using the Residue Number System, where operations are performed over several rings in parallel.

Consider the pipelined system illustrated in Fig 3.5. Let us define the following parameters

ni : Dynamic range of the i-th residue adder

k : Total number of modules

(Nfa)i : Number of full adders in the i-th module

(Nl)i : Number of latches in the i-th module

(Nmux)i : Number of multiplexers in the i-th module

For an ni-bit pipelined RNS adder we need

(Nfa)i = 3ni

(Nl)i = 3ni(ni + 1)

(Nmux)i = ni

It should be mentioned that the modulo (2^m) addition is performed by an ordinary m-bit binary adder with carry overflow discarded.

So the pipelined RNS adder requires;

$$Nfa = \sum_{i=1}^{k} [(Nfa)i]$$

$$Nl = \sum_{i=1}^{k} [(Nl)i]$$

$$Nmux = \sum_{i=1}^{k} [(Nmux)i]$$

We will show by an example in the next section that the comparison between the RNS and Binary adders depend on dynamic-range; the higher the dynamic range, the greater the savings.

3.6.2 THROUGHPUT RATE

In the pipelined structure the synchronization of the data transfer is made by the use of a global clock. One of the problems with this type of global timing is the clock skew, which becomes intolerable for very-large-scale arrays.

Controlling clock skew in VLSI systems is a big problem because the clock transition time is fairly long compared to the propagation delay through a fast MOS gate. The following quotation is from A. Fisher and H.T. Kung [11];

" Unfortunately, large clocked systems can be difficult to implement because of the inevitable problem of the clock skews and the delays which can be especially acute in VLSI system as the feature size shrinks"

The clock skew problem is alleviated by using a pseudo two-phase clock design at the cost of less speed and extra area due to the second runners.

Clock skew is not a significant problem in RNS because RNS implementations require several small-scale arrays which can also be clocked with two phases (single clock).

3.6.3 TESTABILITY

One of the critical factors in VLSI design is testing complex digital circuits. Consider, for example, a 30-bit binary adder. To test this adder exhaustively a sequence of (2^{*0}) inputs must be applied to test the circuit. Assuming one had the pattern and applied them at a rate of 1-us per pattern, the testing will take about 36 thousand years[12]! It is clearly not feasible to do a complete functional test of this circuit. There are some techniques which enable one to generate a test vector which is of reasonable size and also covers about 90% of the faults. Now consider the same 30-bit addition in RNS. Several modules of 5 and 6-bit RNS adders will be adequate to do the job. The test vectors (

exhaustive testing) would be a few sequences of (2^{10}) inputs. The same pattern tester would be able to test this adder in a few milliseconds. This ease of testing is again due to working with small dynamic ranges within the RNS.

3.6.4 LATENCY TIME

The latency time of an N-bit pipelined ripple-carry Binary adder is

Tlat. = N*Tclk

The latency time of the pipelined RNS adder depends on the module with the highest dynamic range (n)

Tlat. = (n+1)*Tclk

It is obvious that n<<N and therefore latency time of the RNS adder is generally much less than the binary adder.

3.7 EXAMPLE

In this example it will be shown that a 20-bit pipelined RNS adder has advantages over its binary counterpart.

The dynamic range of 20-bit is equal to 1048580. A Residue Number System with moduli 31, 16, 15, 13, and 11 has a dynamic range of 1063920; the systems are virtually equivalent in dynamic range. The necessary hardware to realize a 20-bit Pipelined Binary adder is:

Nfa = 20

Nl = 610

The Pipelined RNS adder with moduli (31, 16, 15, 13, 11) needs;

Nfa = 55

N1 = 296

Nmux = 17

It should be noted that the modulo 16 adder is simply a 4-bit binary adder.

In order to make a comparison between the two systems assume that the area of a register is of unit area. The area of the full adder is almost equal to two unit areas and the area of a multiplexer is almost half the unit area. Therefore:

Area of the 20-bit Pipelined Binary adder = 650 unit

Area of the 20-bit Pipelined RNS adder = 414 unit

This means a saving of 37 percent in area for RNS adder.

Latency time the of 20-bit Pipelined Binary adder = 20*Tclk

Latency time of the 20-bit Pipelined RNS adder = 6*Tclk

and this is less than 1/3 of its Binary counterpart.

The improvement in testability and clock skew in RNS adders is evident because of the smaller size of the arrays.

In the same manner it can be shown that the use of RNS for 30-bit addition, results in about a 50 percent saving in area.

3.8 SUMMARY

In this chapter it has been shown that for a wide-dynamic range of operations, pipelined RNS adders offer the following advantages over their binary counterparts:

(I) - Hardware complexity

The chip area of a pipelined binary adder is proportional to N^2, whereas the chip area of a pipelined RNS adder is proportional to $\sum_{i=1}^{K} ni^2$. By proper selection of the moduli

one can have

$$\sum_{i=1}^{K} ni^2 < N^2$$

Which means a saving in chip area for the RNS adder.

(II) - Testability

For pipelined binary and RNS adders, the computer run time to do the test generation and fault simulation is approximately proportional to N^3 and $\sum_{i=1}^{K} ni^3$, respectively [12]. As before by proper selection of moduli an improvement in testability of RNS adders can be achieved since

$$N^3 > \sum_{i=1}^{K} ni^3$$

(III) THROUGHPUT RATE

In the pipelined architecture, clock skew increases with the size of the array. As clock skew increases, the throughput rate decreases. Since in RNS ni << N, clock skew becomes insignificant and hence the array can be clocked at higher speeds.

(IV) Latency time

The latency time of pipelined binary adders depends on the number of bits of the bigger operand, N, whereas the latency time of the pipelined RNS adder depends on the module with the highest dynamic range, n. Since n<<N, therefore, latency time of the RNS adder is generally much less than that of the binary adder.

CHAPTER 4

CIRCUIT AND LOGIC DESIGN WITH CMOS

4.1 INTRODUCTION

CMOS has emerged as the most suitable technology for VLSI design and will be the dominant technology for the next decade [13]. The greatest advantage of CMOS over NMOS is its inherent low power characteristics. Since the heat generated by the power dissipation within the chip is difficult to remove from the package and because the performance of the MOS transistors decreases as the temperature of the chip increases, the design of NMOS VLSI circuit becomes quite complex [14]. NMOS has had some advantages in terms of speed and silicon area needed to produce the same functionality, but with shrinking feature size and the utilization of dynamic circuit design, those advantages are evaporating. As transistor dimensions are reduced, the current delivered by PMOS is approaching the current provided by NMOS of the same size. So small CMOS circuits are not much slower than their NMOS counterparts [15]. Another advantage of CMOS is better noise margin characteristics.

In the next Section we are going to examine different CMOS logic families.

4.2 CMOS LOGIC TECHNIQUES

There are two fundamental approaches to the implementation of logic circuits in CMOS VLSI design.

(I) Static logic where electrical connections and feedback are used to transfer and maintain the logic levels.

(II) Dynamic logic where logic levels are transferred by means of a clock and charge storage on capacitive nodes is employed to retain the logic levels between the clock periods.

Some advantages of static logic are:

- Good noise margin
- No need for clock driver circuits
- Logic level retention at D.C.
- Predictable performance at high temperature

Replacement of static cell by dynamic can offer potential improvements in three areas:

- Power dissipation
- Silicon area
- Speed

Some disadvantages of dynamic logic circuits are :

- Poor noise margin
- Need for clock driver circuits
- Logic retention is not achieved at D.C.

4.2.1 STATIC CMOS COMPLEMENTARY LOGIC

In the conventional static CMOS technique (fully complementary CMOS logic) the logic function of each gate is implemented by two combinational networks. Figure 4.1(a) shows the structure of the general CMOS static gate. The top PMOS network acts as a good switch to transmit undegraded logic 1 to the output and the bottom NMOS network provides

good logic 0 switching.

The CMOS network that performs AND/OR INVERT (AOI) function for one 3-input AND, and one 2-input AND (32 AOI), is shown in Fig. 4.1(b). Several points may be noted from this example. Firstly the NMOS network is the logical dual of the PMOS network. Secondly, for all input combinations there is always a path from Vdd (logic 1) or Vss (logic 0) to the output and the full supply voltages appear at the output. All CMOS static gates may be designed as ratioless circuits [30]. Thirdly, for any input combination there is never a path from Vdd to Vss, except for the very brief period when the output or inputs are undergoing transitions.

4.2.2 PSEUDO-NMOS LOGIC

One of the major disadvantages of the fully complementary CMOS approach is that CMOS circuits occupy more area than NMOS circuits. For example the 32AOI function could be made with 6 transistors in NMOS or pseudo-NMOS, as shown in Fig. 4.2(b). General pseudo-NMOS logic is shown in Fig. 4.2(a). Pseudo-NMOS is equivalent to NMOS except that the depletion or enhancement NMOS load is replaced by a P-device. The main problem with this gate is the high static power dissipation that occurs whenever the NMOS network is in a low impedance state.

In complementary CMOS circuits for every gate driven by the output there is a connection to both an N-device and a P-device. As a result the capacitive load on gates of a fully complementary CMOS circuit is at least a factor of two

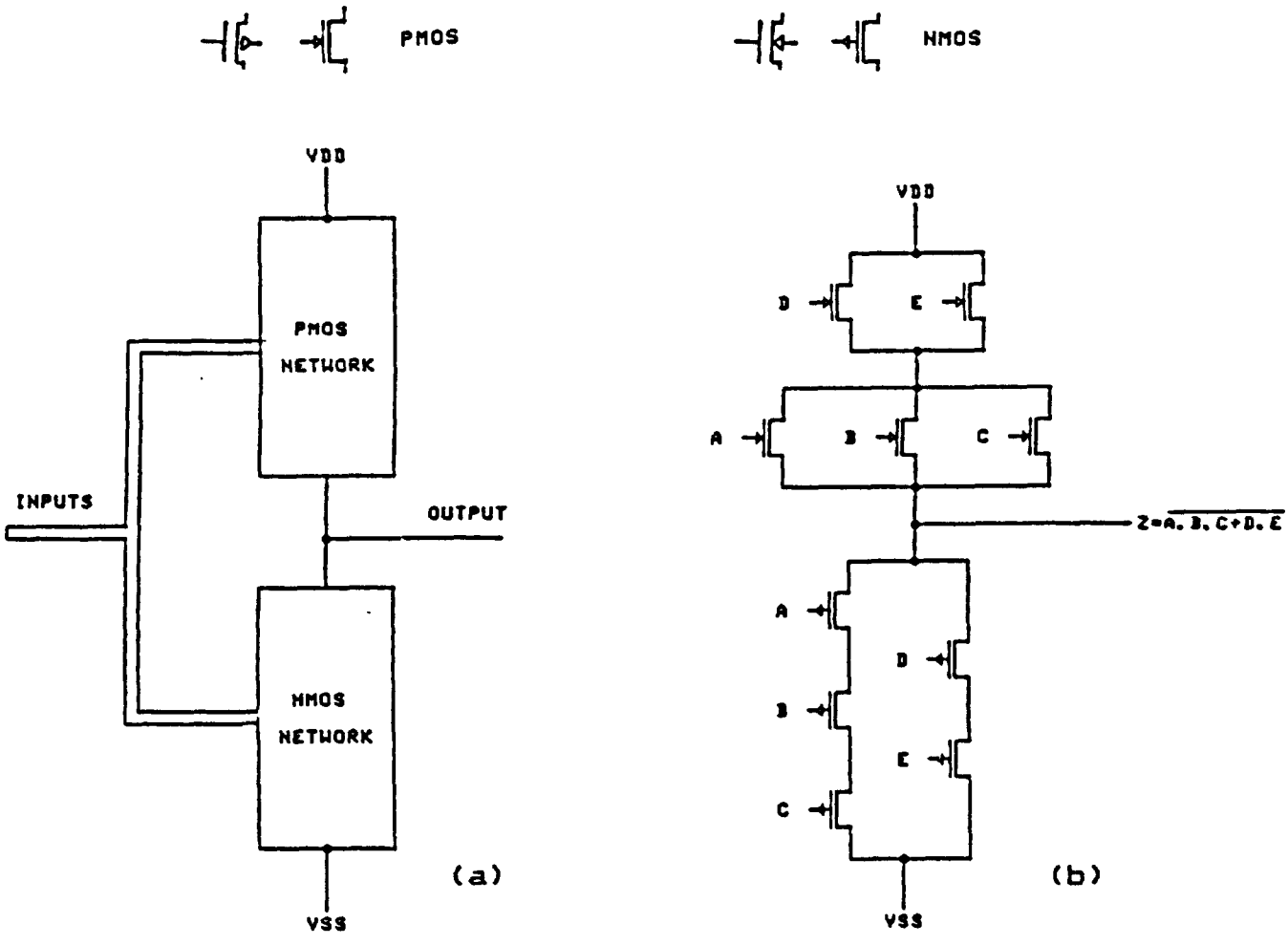
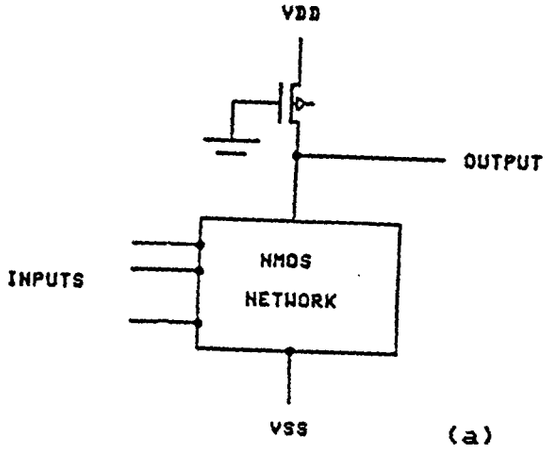


Fig. 4.1 (a) General CMOS Complementary Logic
(b) Fully Complementary CMOS 32AOI Gate



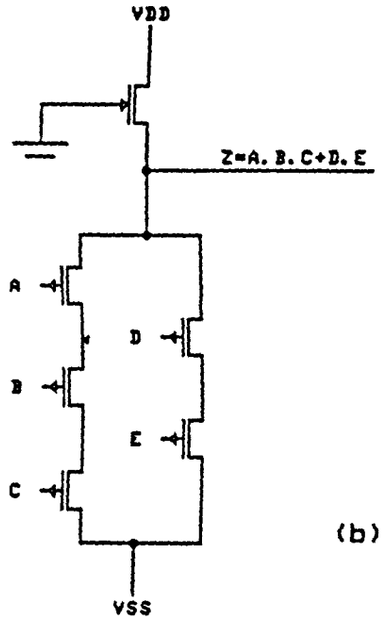


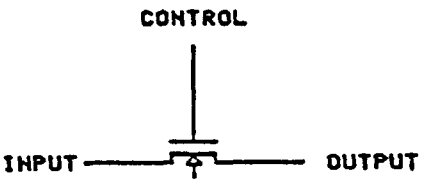
Fig. 4.2 (a) General Pseudo-NMOS Logic
(b) Pseudo-NMOS 32AOI Gate

higher than the load on a pseudo-NMOS circuit. However, in pseudo-NMOS the pull-up device is always on even when the gate is pulling down. This, in turn, slows the pull down. The gain ratio of the P-transistor load to N-transistors has to be selected such that the correct switching is ensured. Usually the gain of the pull-up is chosen smaller than half the pull-down(Ratioed Logic). As a result the speed of the pseudo-NMOS and static CMOS are close. Pseudo-NMOS has less load capacitance but also smaller pull-up gain. The trade-off of choosing one or the other is between the low power of the CMOS and the low area of the pseudo-NMOS.

4.2.3 PASS TRANSISTOR LOGIC (TRANSMISSION GATE)

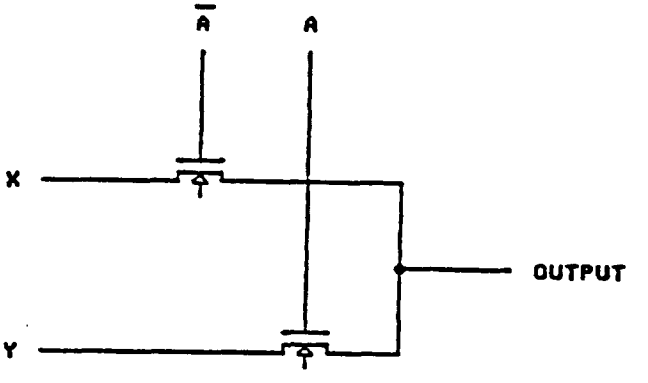
Pass transistor logic is a popular technique in NMOS circuits. Mead and Conway used pass transistors in the design of latches, flip-flops, multiplexers and some combinational logic structures. Pass-network implementation of a logic function results in a faster gate with less area and power dissipation than using conventional logic design. Formal methods for deriving pass transistor logic have been presented for NMOS [16]. Figure 4.3(a) shows the pass gate and its logical function. Figure 4.3(b) shows the basic logic functions - AND, NAND, OR, NOR and XOR - that can be implemented with the assistance of pass transistors.

In CMOS, these structures can be replicated by using a full transmission gate for each original N-transistor. One of the advantages of transmission gates over pass transistors is the ability to pass both logic 1 and 0 without degradation.



PASS_GATE TRUTH TABLE		
INPUT	CONTROL	OUTPUT
0	0	UNDEFINED
1	0	UNDEFINED
0	1	0
1	1	1

(a)



OUTPUT	X	Y
$\overline{A}B$	0	B
$\overline{AB}$	1	$\overline{B}$
A+B	B	1
$\overline{A+B}$	$\overline{B}$	0
$A \oplus B$	B	$\overline{B}$

(b)

Fig. 4.3 (a) Pass-Transistor logic Model
(b) Pass Transistor Structures for Basic Logic Functions - AND, NAND, OR, NOR, and XOR

Figure 4.4 shows the schematic of a novel 6-transistor transmission gate Exclusive-OR gate [30]. The same logic function when realized in conventional static CMOS requires 14 devices. Transmission gates can be used in the design of a half adder, Fig. 4.5 , which results in a very compact circuit. This logic technique can also be used in the design of flip-flops. The schematic of a Pseudo 2-phase flip-flop is shown in Fig. 4.6 . It uses a 2-phase nonoverlapping NMOS clocking scheme (Mead and Conway [34]) and adds complementary clocks (clk1b & clk2b). Note that clk1.clk2=0 for all time. During clk1, the first transmission gate is closed, thereby the input level is stored on the gate capacitance of the inverter. During clk2, the first transmission gate is open and the second transmission gate is closed, causing the inverse of the stored logic value to be placed on the gate of the second inverter. The delays between the clocks are chosen to ensure that, for worst case clock skew, the two clocks do not overlap.

A 2-phase dynamic flip-flop is shown in Fig. 4.7 . A clock race condition, similar to that experienced in pseudo 2-phase clocking, can arise in this structure. If clkb is delayed from the clk signal, we see that the first TG NMOS device can be turned on at the same time as the second TG N-transistor. So the clock skew must be carefully studied and minimized.

4.2.4 CLOCKED CMOS LOGIC (C^2MOS)

The clocked CMOS circuitry is basically composed of

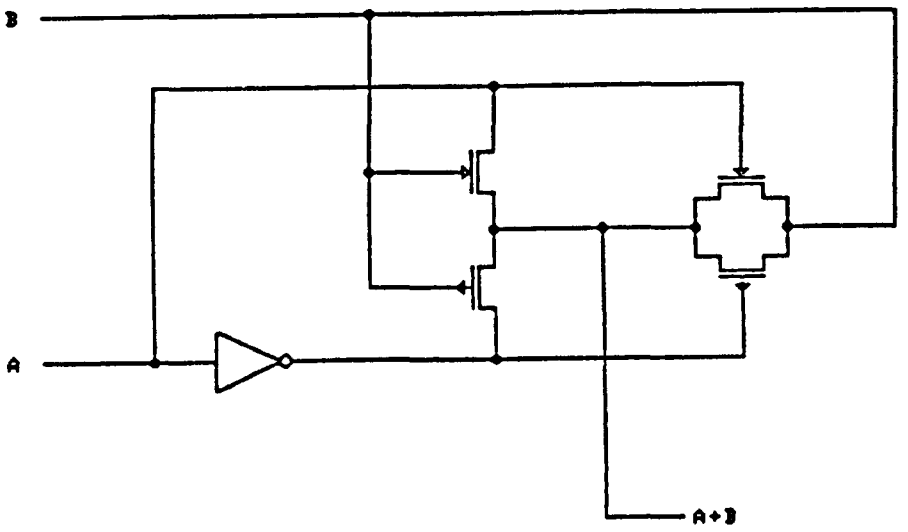


Fig. 4.4 Transmission Gate (TG) XOR Circuit

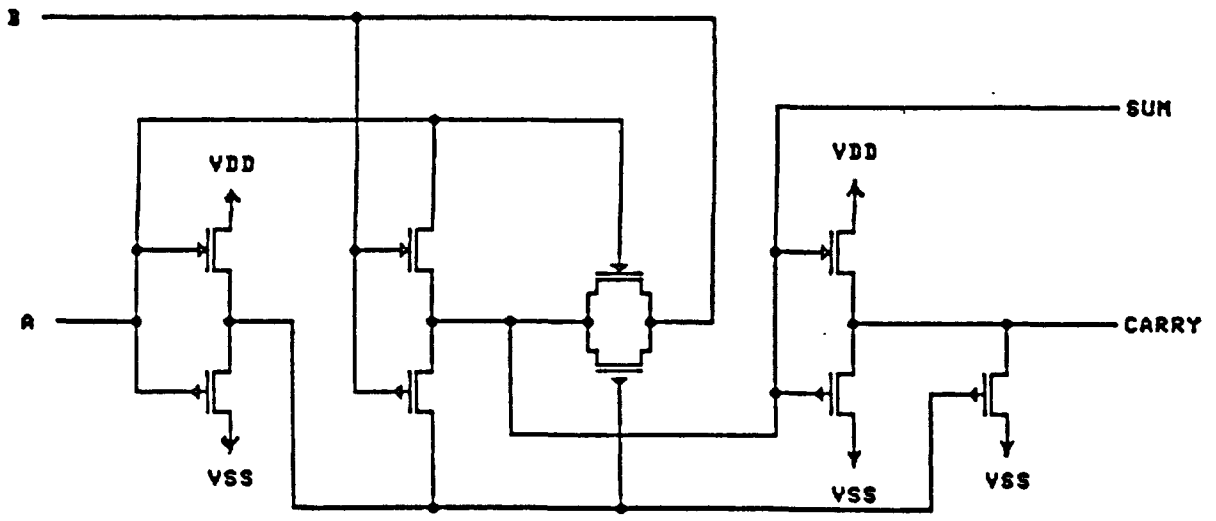


Fig. 4.5 Transmission Gate Half Adder Circuit

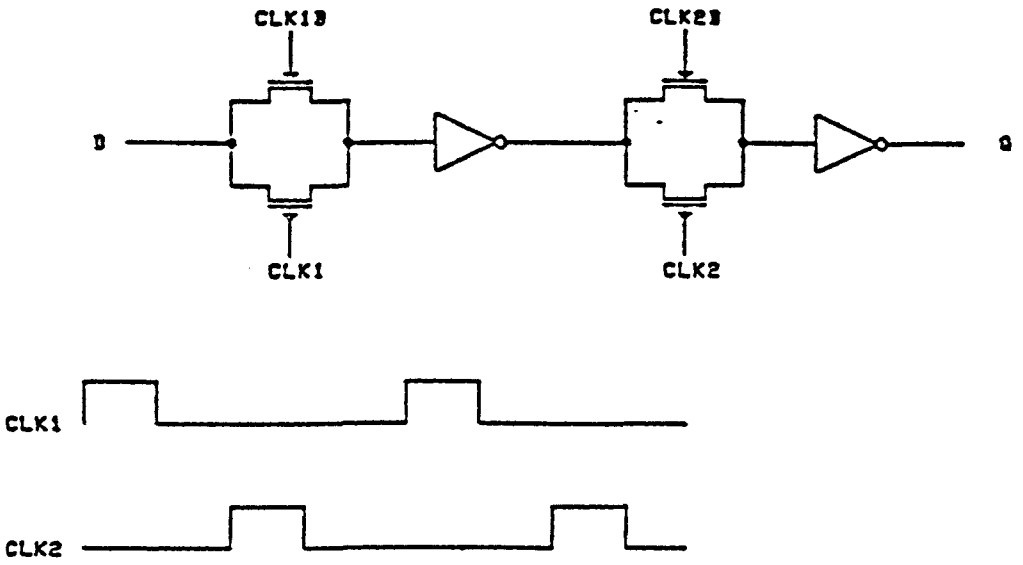


Fig. 4.6 Pseudo Two-Phase TG Flip-Flop

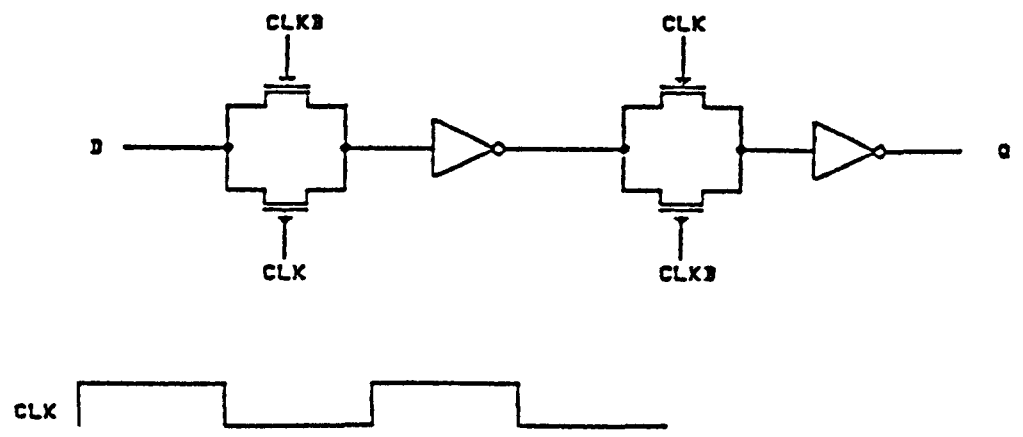


Fig. 4.7 Two-Phase TG Flip-Flop

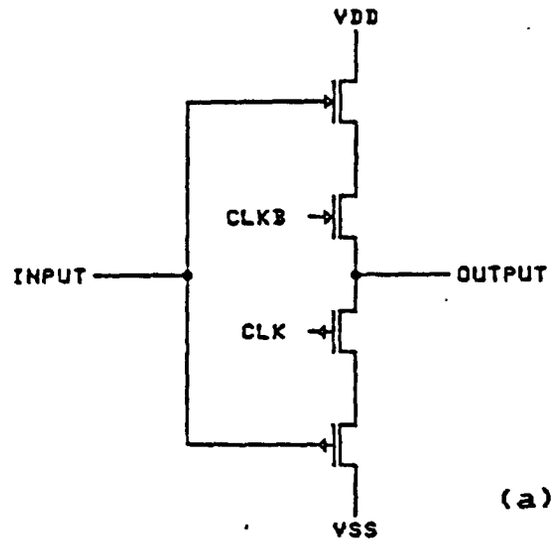
conventional static CMOS and two extra MOS switches connected in series [17]. Two types of clocked CMOS inverter are shown in Fig. 4.8. In Figure 4.8(b) the MOS switches are sandwiched between the CMOS inverter while in Fig. 4.8(a) the MOS switches are connected on both sides of the conventional CMOS inverter. The operation of the clocked CMOS inverter is as follows. Whenever clk = 1 the circuit acts as a conventional CMOS inverter. When clk = 0, the output holds the previous value.

When the input data makes a transition and the clock pulses are not supplied to the clocked CMOS inverter of Fig. 4.8(b), some of the charges stored in the output will be transferred from the output capacitance to the parasitic capacitances Ca or Cb. This is called charge sharing and happens whenever there is a phase difference between the clock pulses and input data pulses. The circuit of Fig. 4.8(a) does not have such problem.

The clocked CMOS circuit of (32 AOI) is shown in Fig. 4.9. The main use of this logic technique is to form clocked circuits that incorporate latches or interface with other dynamic logics. It will be shown later that clocked CMOS latches and shift registers are more compact than conventional static CMOS counterparts.

4.2.5 BASIC DYNAMIC CMOS LOGIC

Dynamic CMOS circuits generally work based on the principle of precharging a node to a particular level (usually high for NMOS), while the current path to the other level (ground



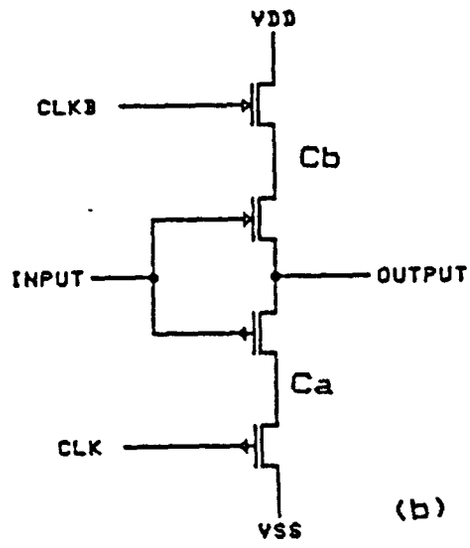


Fig. 4.8 (a) Clocked CMOS Inverter (without Charge Sharing)
(b) Clocked CMOS Inverter (with Charge Sharing)

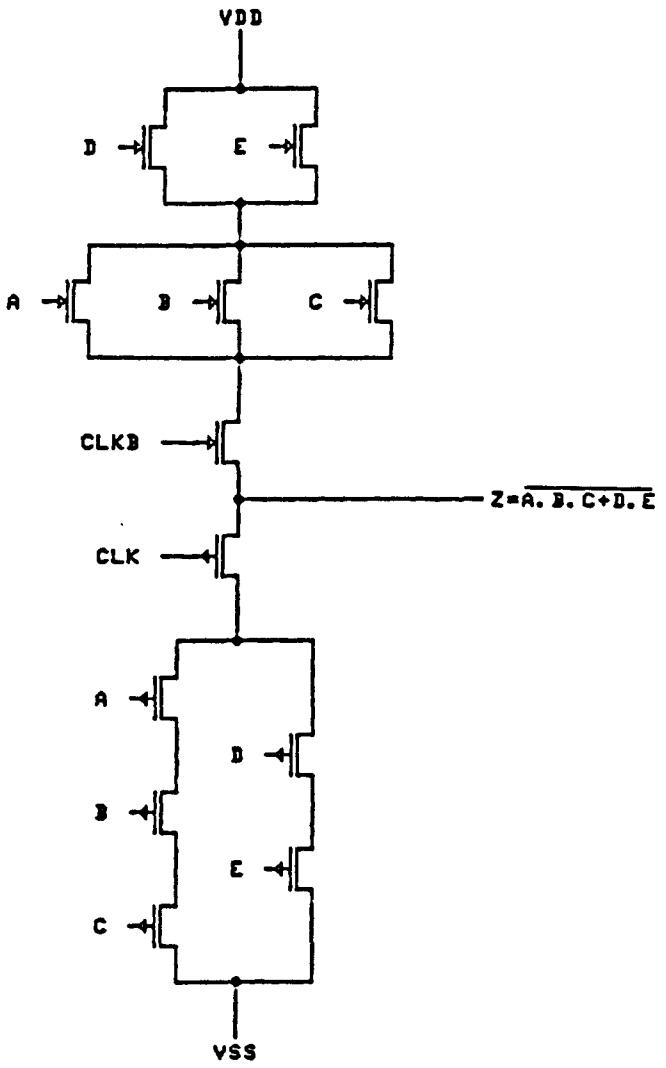


Fig. 4.9 Clocked CMOS 32AOI Gate

for NMOS) is turned off [18]. During the precharge phase the inputs to the circuit change. The next phase of operation is the evaluate phase. Here the path to Vdd is turned off by a clock and the path to ground (Vss) is turned on. Depending on the state of the inputs, the output either will be pulled down or will float at the high level. The fundamental characteristic of dynamic logic is that it employs charge storage on capacitive 'soft' nodes to retain logic levels between the clock periods. Charge leakage from such a soft node limits the minimum clock frequency. Figure 4.10(a) shows a dynamic NOR2 gate. The discharge timing diagram of the soft node obtained from SPICE is shown in Fig. 4.10(b). The time constant of this curve is a few milliseconds. The typical value for the minimum clock frequency of this circuit is about 1-2 KHz.

The schematic circuit of the dynamic CMOS gate for the (32AOI) function is given in Fig. 4.11 . One of the advantages of the dynamic CMOS circuits is the reduced silicon area. Whereas there are 2n transistors in a conventional static CMOS gate, the dynamic style needs only n+2 transistors. The load capacitance of dynamic CMOS circuits is the same as the pseudo-NMOS gate but full pull-down current is available. The pull-up time is improved by virtue of the precharge transistor, and power dissipation is closer to static CMOS than static pseudo-NMOS. Since in dynamic CMOS each gate must be precharged high every cycle, even if its output is to continue low, there is some power penalty compared to static CMOS.

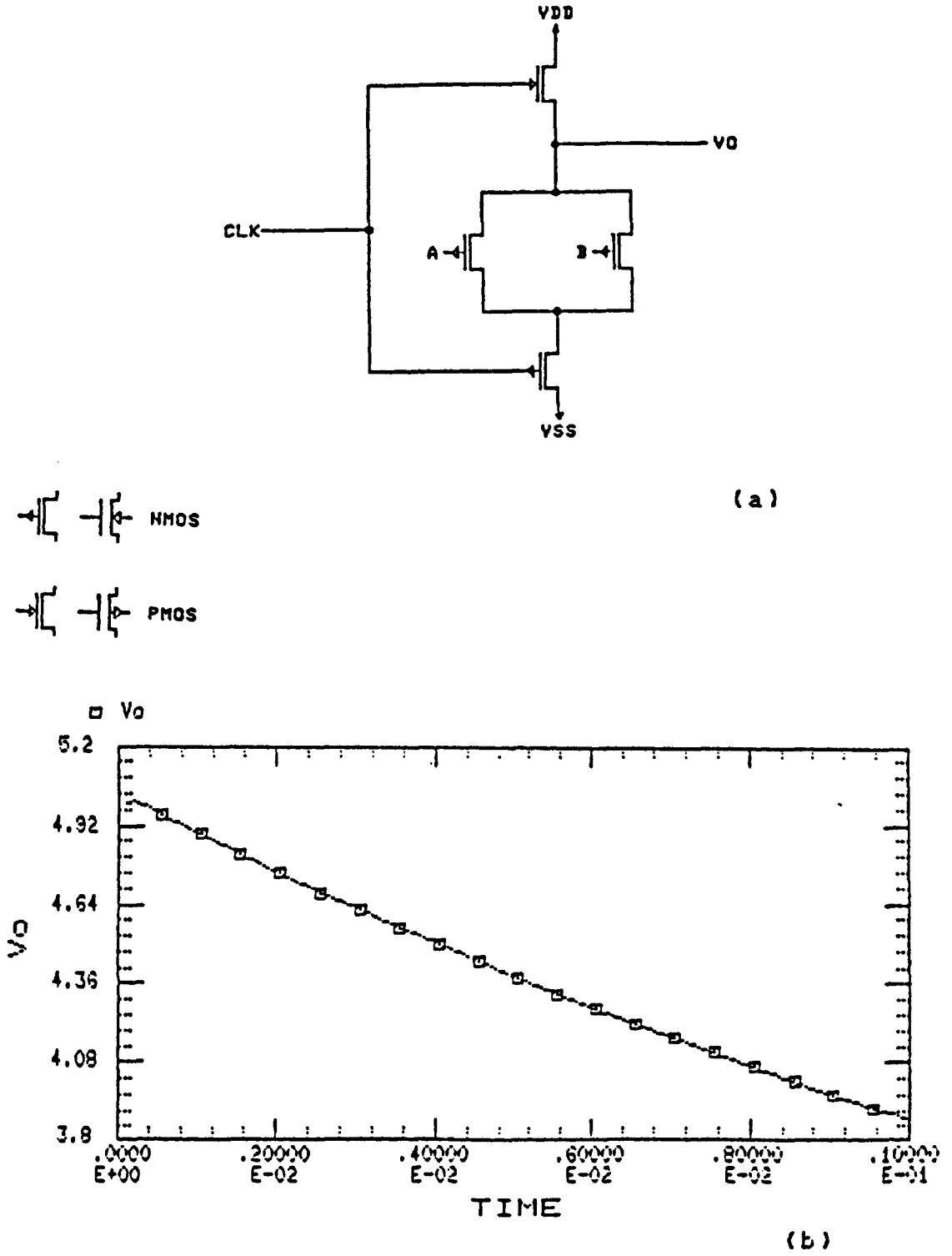


Fig. 4.10 (a) Dynamic CMOS NOR2 Gate
(b) Discharge Timing Diagram of the Precharged Node (Vo), Obtained from SPICE

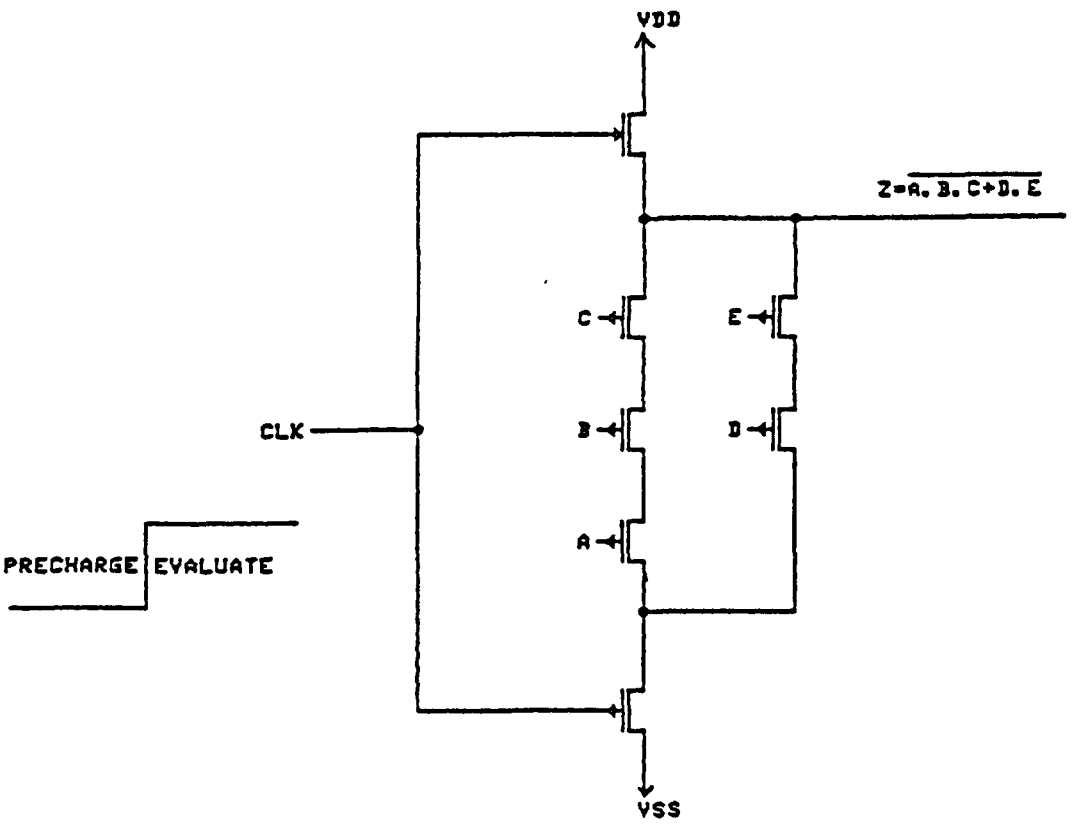
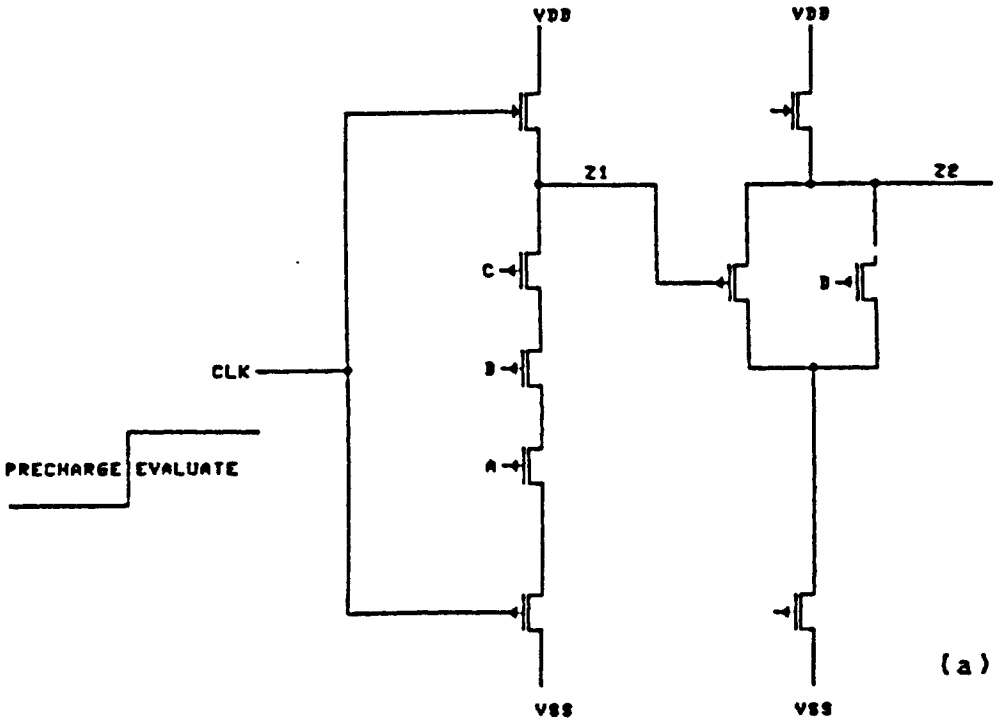


Fig. 4.11 N-Type Dynamic CMOS 32AOI Gate

This structure has two significant problems. First, if the inputs do not change during the precharge phase, the charge redistribution could corrupt the output node voltage. Second, single phase dynamic CMOS gates cannot be cascaded. Consider the two-stage dynamic CMOS circuit shown in Figure 4.12(a). During the precharge phase the output nodes Z1 and Z2 are charged to Vdd. Suppose that output Z1 of the first gate is supposed to go low (Vz1=0) and output Z2 of the second gate has to stay high during the evaluate phase (Vz2=1). Since there is a finite delay involved in this transition (Z1 --> 0), the precharged node Z1 will discharge the output node Z2 of the following gate before the first gate is correctly evaluated. This is called the internal delay race problem. Figure 4.12(b) shows the waveforms associated with Fig. 4.12(a) obtained from SPICE for the critical case when A=B=C=1 and D=0. The glitch problem in this circuit is seen on the output node Z2.

4.2.6 DOMINO CMOS LOGIC

A simple modification to the dynamic CMOS circuit allows a single clock to precharge and evaluate a cascaded set of dynamic logic blocks. The internal race problem is solved in DOMINO CMOS by placing a static inverter after every dynamic block; as shown in Fig. 4.13(a) [19]. During the precharge phase (clk=0), the output nodes of the dynamic gates are precharged high and the output of all the static inverters is low. Consequently, all the N-channel transistors fed from these inverters, will be turned off during the precharge



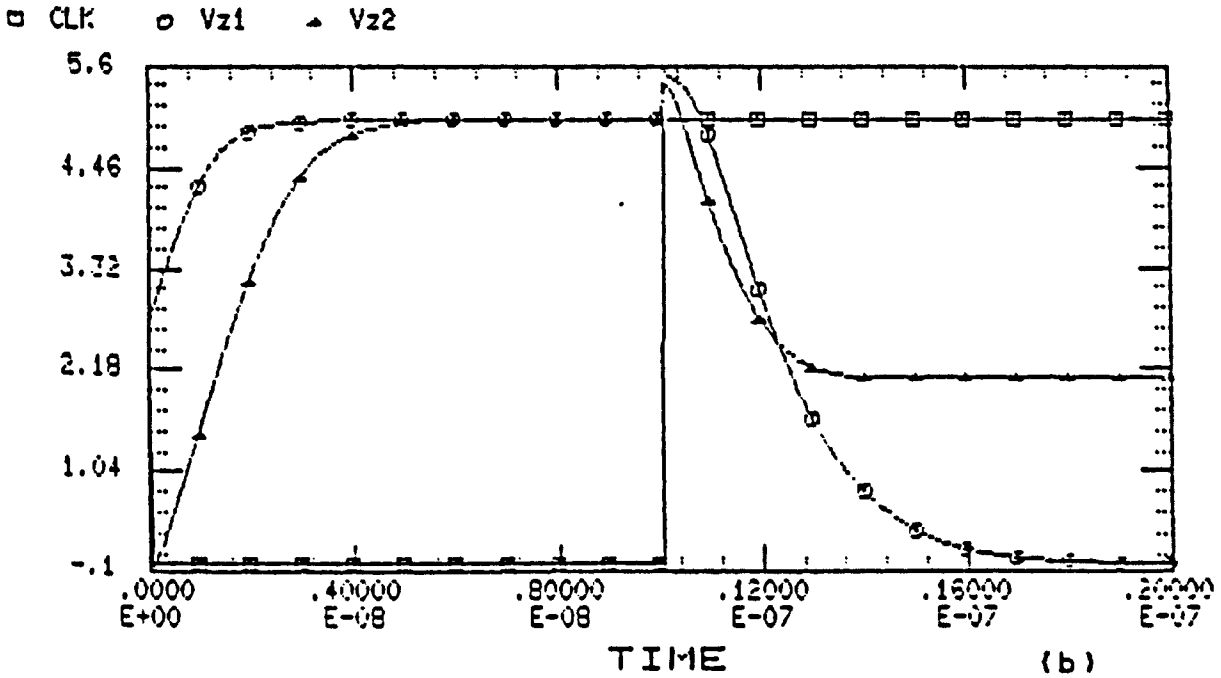


Fig. 4.12 (a) Cascaded Dynamic CMOS Logic
(b) SPICE Simulation for A=B=C=1 & D=0, Showing the Internal Delay Race Problem

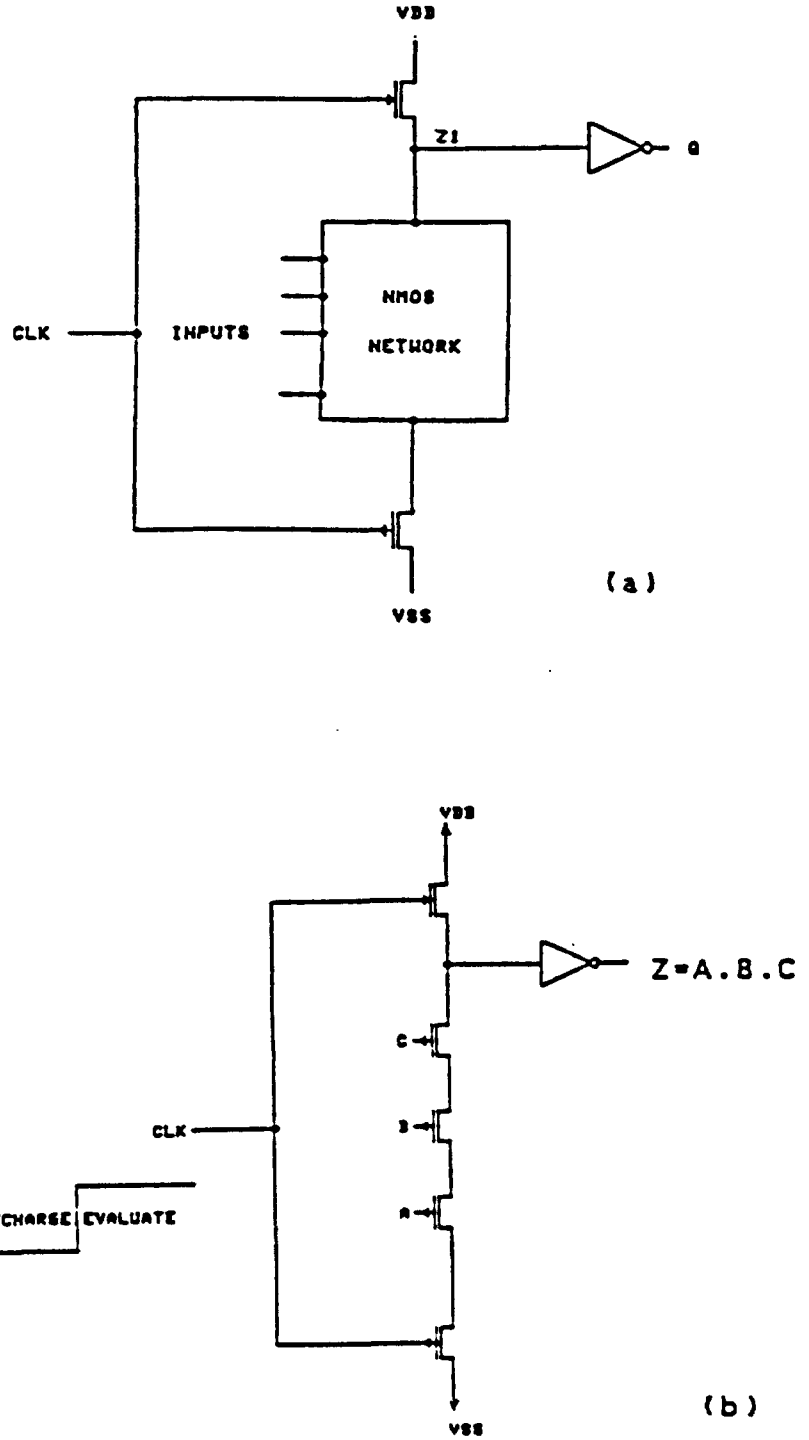


Fig. 4.13 (a) Domino CMOS Logic
(b) Domino CMOS AND3 Gate

phase. During the evaluation phase, the output will conditionally discharge, causing the output of the static inverter to conditionally go high. During the evaluation, the dynamic gate can make at most one transition (1 to 0), the inverter can only make a single transition (from 0 to 1), as a result there can be no glitches at any nodes in the circuit. A single DOMINO circuit gate is shown in Fig. 4.13(b). It is a dynamic CMOS gate driving an inverter buffer. Only the output of the static inverter is fed to the other gates of the circuit, the output of the dynamic gates goes only to the inverting buffer. During precharge all DOMINO outputs turn off the transistors they drive, which guarantees that there can be no glitches at any nodes in the circuit. Therefore all gates may be switched from precharge to evaluate with the same clock edge. An example of a two-stage DOMINO CMOS circuit is shown in Fig. 4.14. During precharge, nodes Z1 and Z3 are high so nodes Z2 and Z4 are low. Assume the input values A=B=C=1 and D=0. During the evaluate phase node Z1 goes low, causing node Z2 to go high. Since D=0, nodes Z3 and Z4 maintain their previous values. The chain of actions propagating in such a fashion is reminiscent of the behavior of a row of dominoes toppling into one another, hence the name DOMINO logic [19].

Replacing a static cell by a dynamic one can result in a significant saving in silicon area. In DOMINO logic all the P-channel transistors are eliminated, at the expense of one PMOS and one additional NMOS. This leads to a saving in silicon area and reduction of the output capacitance, which

results in higher speed and lower DC power dissipation. Only a small amount of power is needed to precharge the output high every cycle (if the output was pulled down in the previous cycle). One limitation of DOMINO circuit techniques is that only noninverting gates are possible. This limits the logic flexibility and implies that logic inversion has to be performed at the inputs or outputs of the blocks of domino logic. Another drawback is that each gate must be buffered. Finally, in common with clocked CMOS circuits, charge redistribution can be a problem. Depending on the situation, the effect of these problems can be minimized. For example, in arithmetic logic units, the necessary XOR gates may be implemented conventionally and driven by the last domino circuit. The buffer is often needed anyway to achieve maximum speed. Charge redistribution problem and its effect on the performance of domino CMOS logic will be studied in section 4.3 . Several solutions to the charge-sharing problem are also presented.

4.2.7 NORA CMOS TECHNIQUE (N-P DYNAMIC LOGIC)

The schematic circuit of NORA technique is shown in Fig. 4.15 [35]. The logic functions are implemented using blocks of NMOS and PMOS transistors. During the precharge phase of operation the first stage (NMOS network) is precharged high and the second stage is precharged low. The PMOS transistors of the second stage will be turned off during the precharge. Therefore, alternately cascading NMOS and PMOS logic blocks will allow the circuit to operate race free (NO RAce). Some

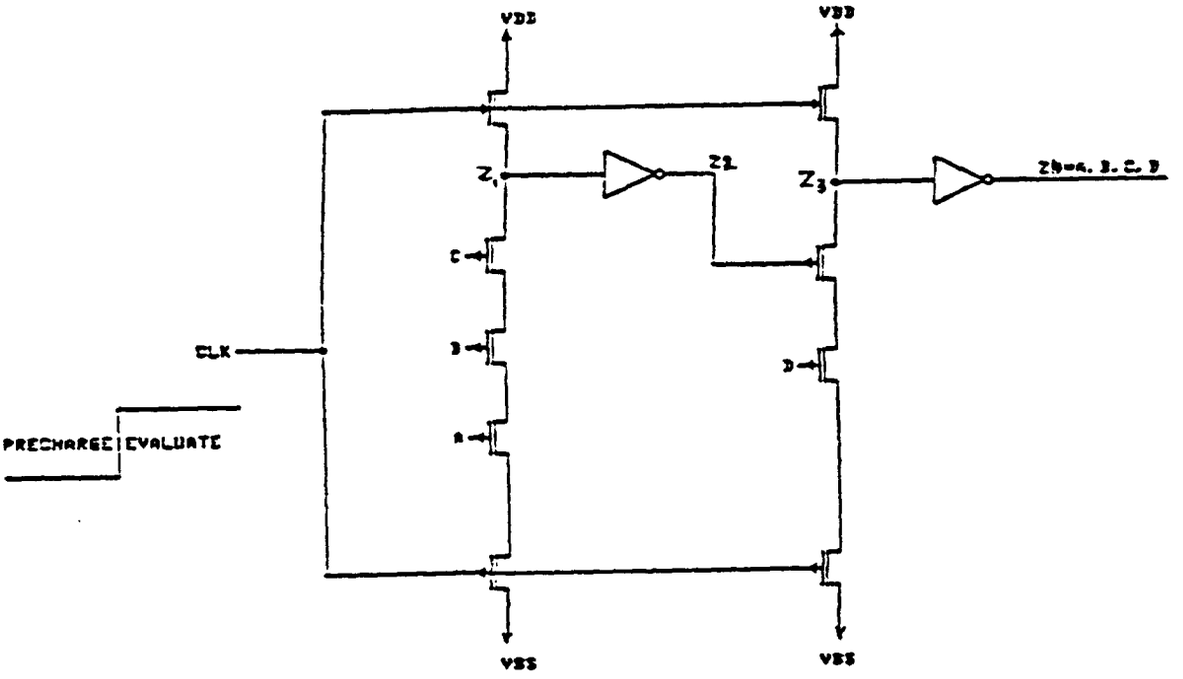


Fig. 4.14 Two-Stage Domino CMOS Gate

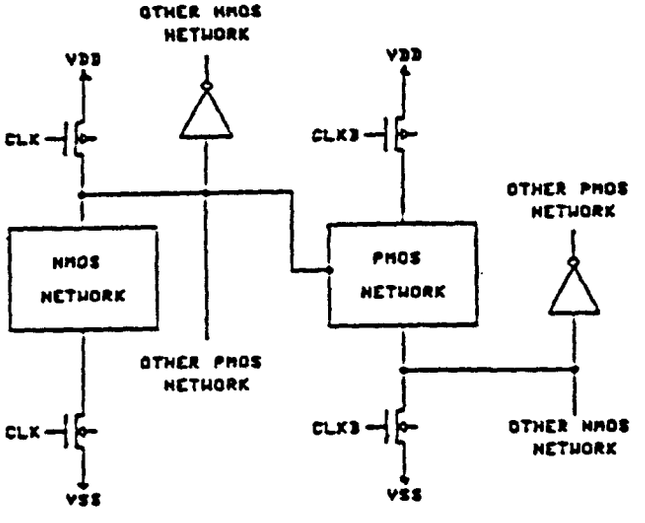


Fig. 4.15 N-P Dynamic CMOS Logic

advantages of this circuit over Domino logic are:

I - The ability to implement inverting function

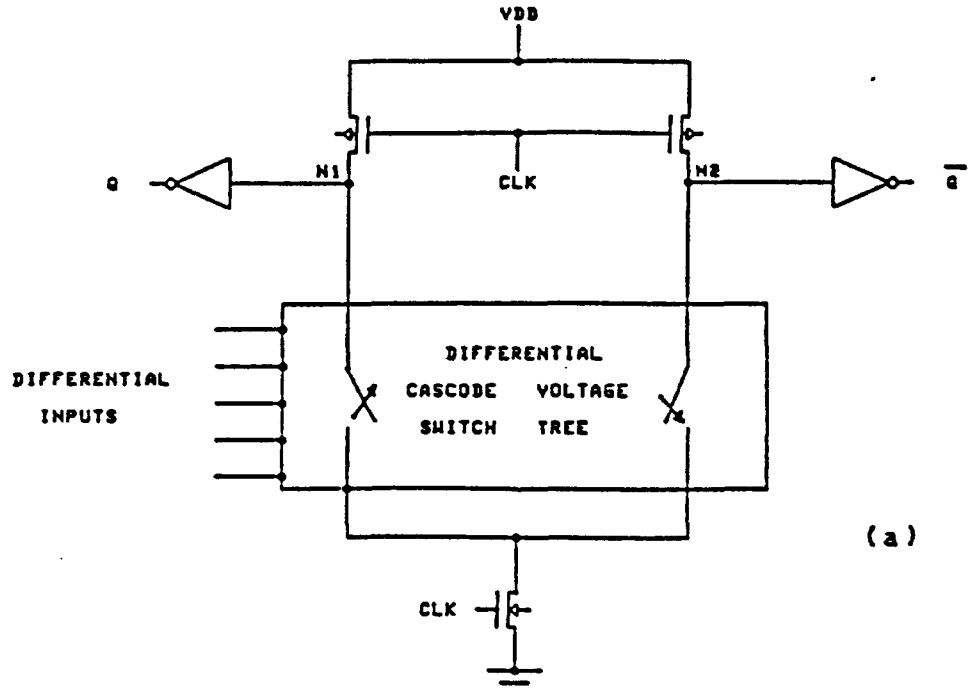
II - The absence of inverters on the gate outputs

Problems with this logic technique includes the low-speed of PMOS network, charge redistribution, and reduced noise margins.

4.2.8 DIFFERENTIAL CASCODE VOLTAGE SWITCH LOGIC (DCVSL)

The Differential Cascode Voltage Switch Logic (DCVSL) [20] is a complete logic family because it provides complementary outputs. This property of having the true and the complement of the output function (differential signals) is a free self-test feature [23]. The DCVSL circuit concept is illustrated in Fig 4.16(a). This logic family has also two phases of operations, precharge and evaluate. During the precharge phase the transistors MP1 and MP2 are on while the transistor MN1 is off. Nodes N1 and N2 will be charged to Vdd and outputs from the inverters are at 0v. When the clock goes high (evaluate phase) the transistor MN1 is on and MP1 and MP2 are off. Depending on the differential inputs, either node N1 or N2 is pulled down by the Differential Cascode Voltage Switch trees. The true and complement of the output function will be ready in the evaluate phase. Since the inputs drive only the NMOS devices, the load capacitance of DCVSL is smaller by a factor of two or more,than its static CMOS counterpart.

The DCVS tree can be designed by intuition or using some existing algorithms [25]. A method of designing the DCVS



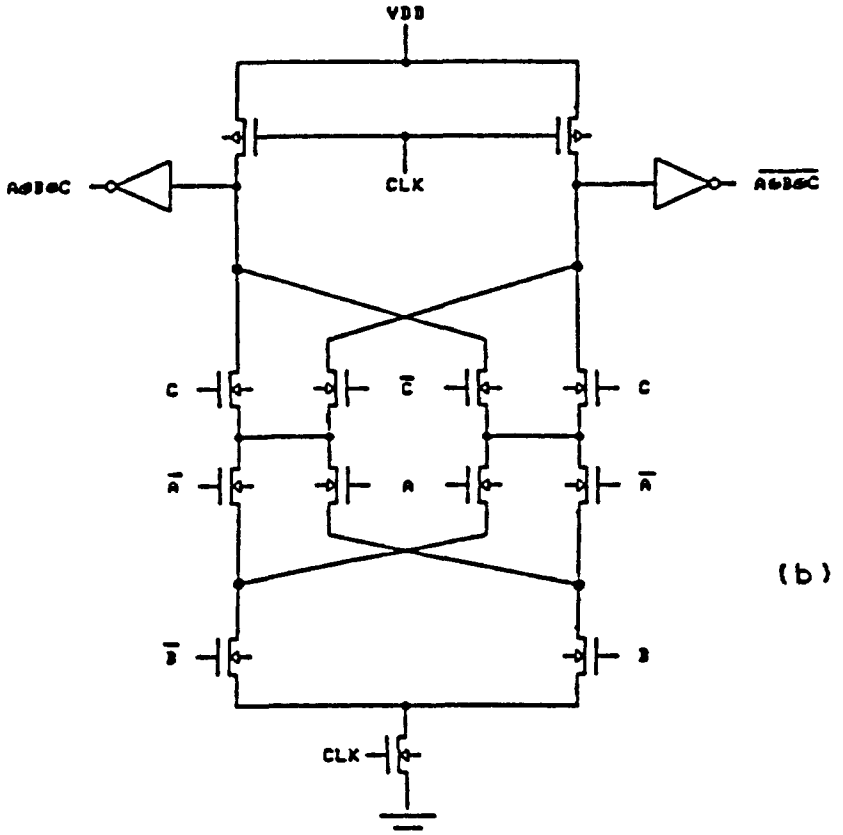


Fig. 4.16 (a) Differential Cascode Voltage Switch Logic (DCVSL)
(b) Three-Input XOR DCVSL Gate

tree is discussed in section 4.4. Fig. 4.16(b) shows a 3-input XOR gate. This example shows the efficiency of a DCVS tree. Device redundancy is reduced by the functional power of the DCVS tree.

4.2.9 LATCHED DOMINO CMOS LOGIC

Latched Domino CMOS is another alternative circuit technique to alleviate the inversion problem in domino logic [21]. Latched Domino (LDOMIO) produces double rail (differential) output signals from single rail (single-ended) input signals for any gates. Ldomino gates can be used as an interface between blocks of static CMOS and Domino or DCVSL logic. It can drive Domino gates directly but cannot be driven by Domino gates. Ldomino provides differential outputs but, unlike DCVSL, differential inputs are not required. Fig. 4.17 illustrates a basic latched Domino gate. It consists of a Domino gate and an additional unbalanced sense amplifier/latch circuitry. The cross-coupled NMOS devices Ms1 and Ms2 with PMOS feedback devices Mf1 and Mf2 form the sense amplifier/latch. The capacitance at node N1 is larger than that of node N2, due to the parasitic capacitances of the NMOS combinational network which implements the logic function. This provides the necessary imbalance in the latch.

During the precharge phase, both nodes N1 and N2 are charged to Vdd and the outputs of the static inverters, Q and QB are set low. The inputs of the gate must be applied during the precharge phase for glitch-free operation. When the clock

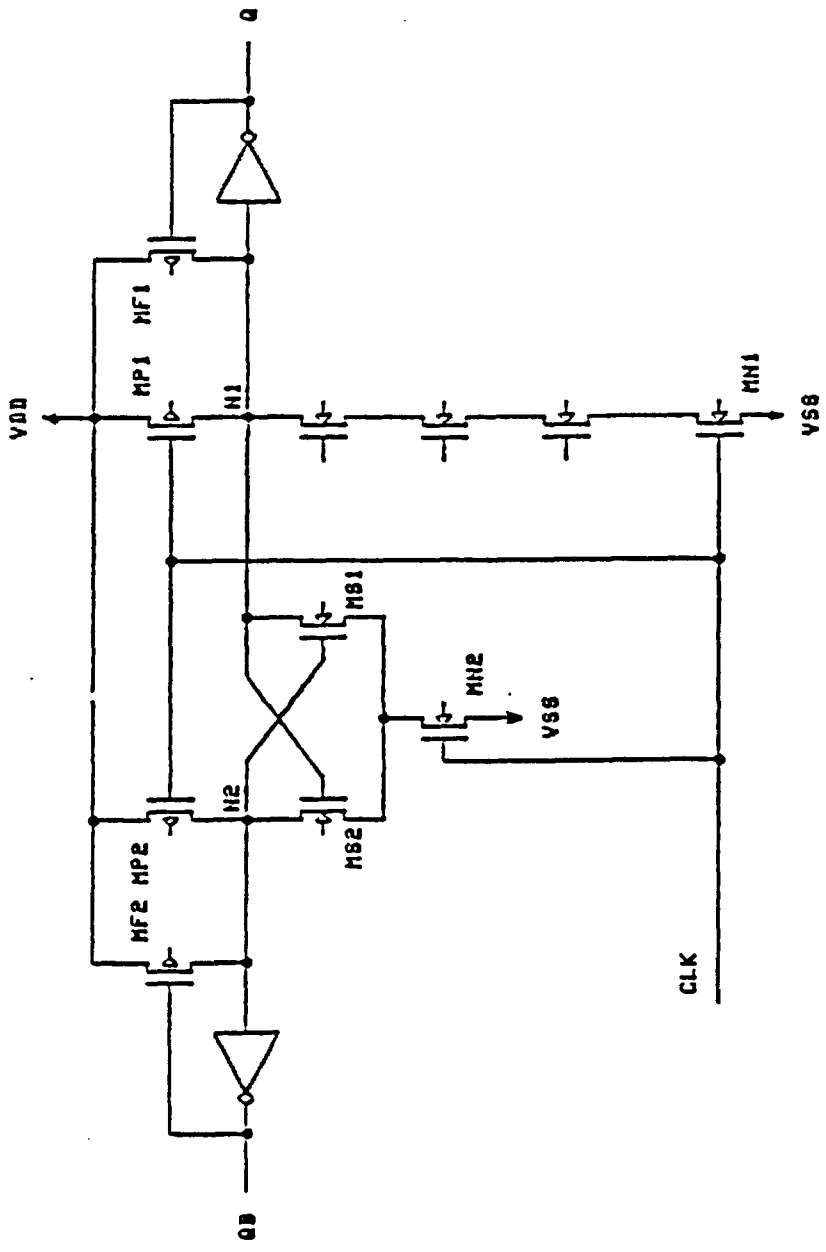


Fig. 4.17 Basic Latched Domino Gate

signal goes high, depending on the input signals, the node N1 either stays high or is discharged. If the node N1 stays high, the sense amplifier/latch will discharge node N2. Output Q stays low while QB (output complement) rises to logic 1. If the node N1 is discharged, it overpowers the unbalanced latch and N2 will remain high. Output Q goes high while QB stays at its logic zero.

The devices in the Ldomino circuit have to be carefully dimensioned to optimize the performance. Since the sense amplifier is deliberately unbalanced, it is not very sensitive to process variations. By making the channel width of the device Ms1 wider than Ms2 we can increase the imbalance. If the degree of imbalance is slight, both devices Ms1 and Ms2 will simultaneously conduct (during evaluate phase) for a long time and will destroy the noise margin. The following steps are suggested [21] for device dimensions in Ldomino gate;

(1) The inverter dimensions are selected for the required rise time and ratioed to obtain a logic 1 noise margin about 0.5v less than logic 0 noise margin.

(2) All the N-channel devices have minimum channel width.

(3) Precharge devices are not critical and are selected for the proper precharge period.

(4) The P-channel feedback devices Mf1 and Mf2 must have a W/L ratio of about 0.3 times the W/L ratio of N-channel devices.

4.2.10 SAMPLE-SET DIFFERENTIAL LOGIC (SSDL)

Sample-Set Differential Logic (SSDL) which is an improved CMOS logic circuit using a differential cascode voltage switch (DCVS) tree has been proposed recently [22]. This logic technique allows the use of several transistors in series in the DCVS tree without a significant penalty in speed. The SSDL circuit operation consists of a sample and a set phase. Figure 4.18(a) shows the basic structure of the SSDL circuit. The true and complement inputs are applied to the DCVS tree (combinational NMOS network) and the SSDL gate generates both true and complement outputs. The SSDL circuit is a two-phase (single clock) dynamic logic which operates with a clock signal (clk) and its complement (clkb). During the sample phase (clock low) transistors Mp1, Mp2 and Mn1 are turned on and nodes N1 and N2 are precharged high. Due to the differential nature of the DCVS tree, a small voltage imbalance results between the internal nodes N1 and N2. During the set phase (clock high) the sense amplifier is activated and a voltage imbalance between nodes N1 and N2 causes the flip-flop to switch, rendering the full Vdd-Vss swing. The speed advantage of the SSDL circuit over DOMINO and DCVSL techniques is obtained through the reduced delay of the sense amplifier path (Mn2, Mn3, and Mn4) rather than the path through the series NMOS transistors. Fig. 4.18(b) shows a 3-input XOR SSDL gate.

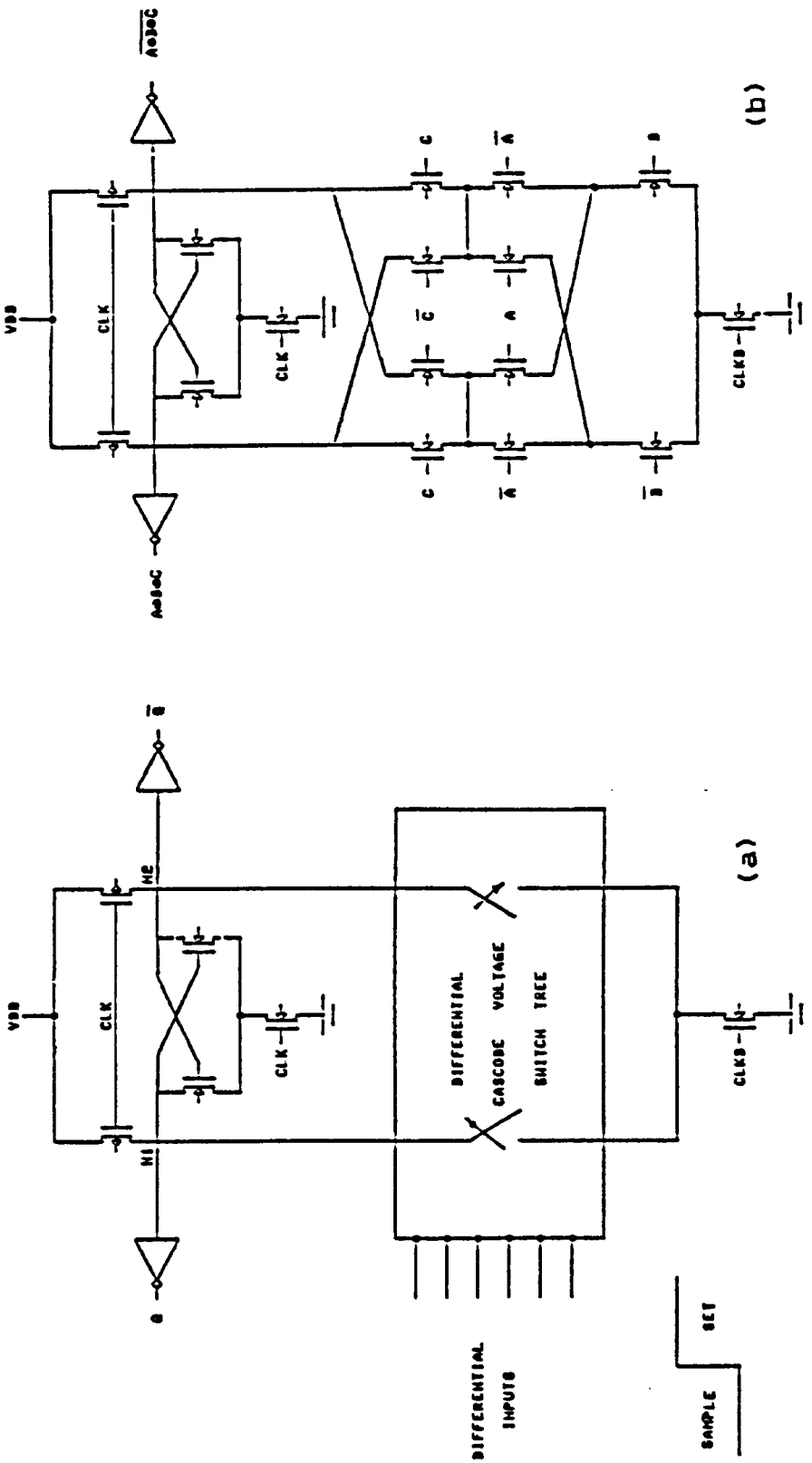


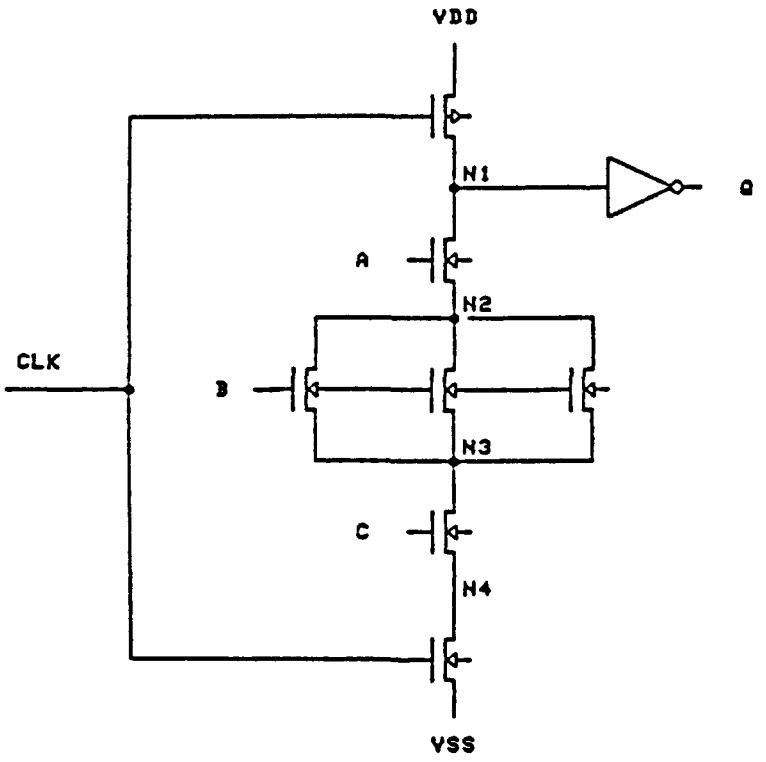
Fig. 4.18 (a) Basic Sample-Set Differential Logic (SSDL)
(b) Three-Input XOR SSDL Gate

4.3 CHARGE REDISTRIBUTION IN DOMINO CMOS LOGIC

Domino CMOS logic presents a charge redistribution problem that occurs between internal parasitic capacitances in the gate. Consider the circuit diagram of the Domino gate shown in Fig. 4.19(a). During the precharge phase, the charge is stored only at the top node N1 and the nodes N2, N3, N4 are not charged. Therefore, during the evaluate phase, there may be an electrical path to several discharged nodes (causing charge sharing) without an electrical path to ground. This charge redistribution results in a voltage drop at node N1 which is at logic 1. If the charge removed from node N1 is sufficient to reduce the voltage at node N1 below the inverter threshold voltage Vinv, then the inverter will falsely switch and cause other gates to switch [23]. The condition for false switching is

$$Vdd \frac{CN1}{CN1 + Ci} < Vinv$$

Where CN1 is the parasitic capacitances at node N1 and Ci is the uncharged internal nodes capacitances. Fig. 4.19(b) shows an example of the charge redistribution . We observe the Domino gate of Fig. 4.19(a) for two full cycles. During the first cycle inputs A,B and C are set to logic 1, causing the discharge of the internal nodes. During the second precharge phase (clk=0), the node N1 is charged to Vdd. In the second cycle the inputs are set to A = B = 1 and C = 0, creating an electrical path between the nodes N1, N2 and N3. This situation causes charge redistribution among the nodes N1, N2 and N3. The input combination A = B = 1 , C = 0 is



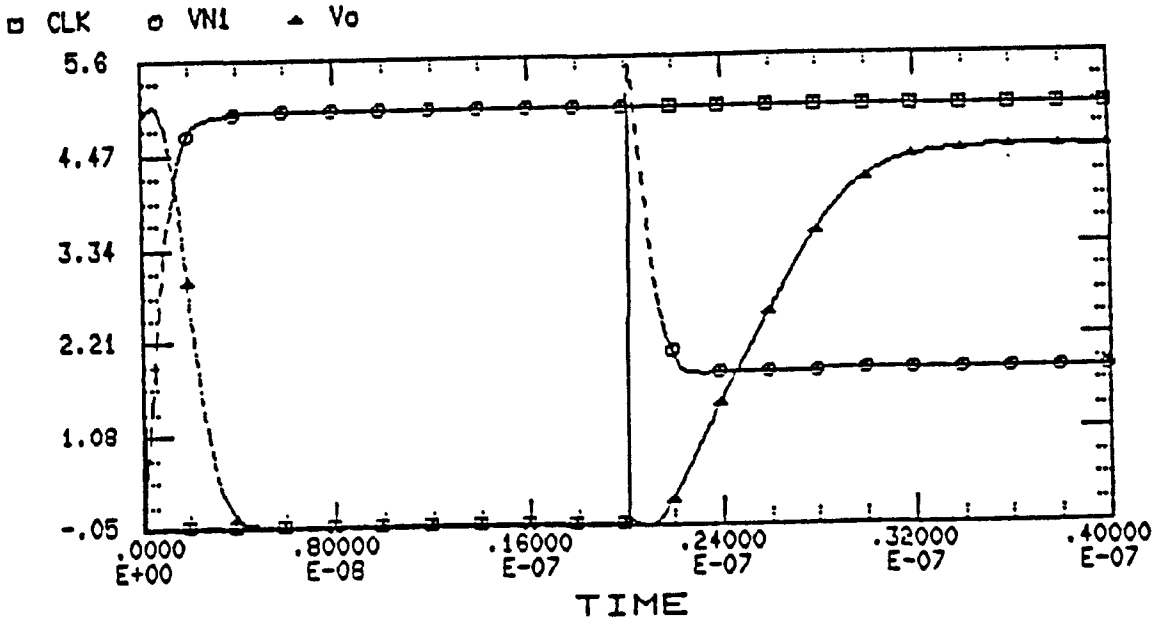


Fig. 4.19 (a) Domino Circuit
(b) Waveforms Associated with Domino Logic, Fig 4.19(a). The Effect of Charge Sharing is Seen on Node N1

supposed to produce logic zero value at the output. However, because of charge sharing at node N1 the voltage drops to 1.7v. This produces an erroneous value of logic one at the output.

4.4 NOISE MARGIN OF DOMINO GATES

The noise margin of a Domino gate is determined by both the dynamic and static parts. The logic 1 noise margin at node N1 (Fig. 4.19(a)) which is referred to as the internal noise margin (VNM1), is determined by two factors [24]:

1) The noise margin of the inverting buffer

2) The charge redistribution between the precharged node and the discharged internal nodes.

The logic 1 noise margin of the static inverter is derived using an approach followed in [30]. The resulting expression for the unity gain point at node N1 is

$$Vu1 = \frac{-b2 + \sqrt{[\, b2^2 - 4b1*b3 \,]}}{2*b1}$$

where

$$b1 = 1/2 - 1/8\ (BR + 1)^2$$

$$BR = BP/BN$$

and BP, BN are the transconductance factors of the PMOS and NMOS devices.

$$b2 = 1/4\ (BP - 1)(BR*VDP/2 + VtOn) - Vton(BR + 1)/2 + BR*VDP$$

$$VDP = Vdd + VtOp$$

Where Vton and VtOp are the threshold voltages of the NMOS and PMOS devices with zero source to substrate voltage.

$$b3 = VtOn(BR*VDP + VtOp)/2 - (BR*VDP + VtOn)^2/8 - BR*VDP*VDP/2$$

The logic 1 noise margin of the static inverter is given by

$$NM1 = Vdd - Vu1$$

After charge redistribution is complete, VN1 drops by a value of D(V) which depends on the initial values of the voltages on the precharged node and the other internal nodes participating in charge sharing as well as the value of the parasitic capacitances of these nodes. The internal noise margin at node N1 can be expressed as

$$VNM1 = NM1 - D(V)$$

The logic zero noise margin at node N1 is the logic zero noise margin of the static inverter.

4.5 METHODS OF IMPROVING THE CHARGE-REDISTRIBUTION PROBLEM IN DOMINO CMOS GATES

In this section we examine the techniques used to alleviate the charge redistribution problem. One method to reduce charge sharing is to ensure that the signals which go high during the precharge phase and are stable during the evaluate phase, are connected to the transistors which are closet to the output node. Other techniques include the control of layout dependent capacitances, incorporating P-channel feedback devices and multiple precharging of the internal nodes. Also a novel circuit technique using a NOR buffer to solve the charge redistribution problem that occurs in Domino gates with large fan-in is presented.

4.5.1 CONTROL OF LAYOUT DEPENDENT CAPACITANCES

One method of reducing the glitch sensitivity is to increase the capacitance of the precharge node and reduce the parasitic capacitances on the other internal nodes. Since the widths of the N-channel devices in the switching function block are selected to meet the required delay specification of the gate, not much can be done to reduce the internal capacitances. The precharge node capacitance can be increased by making the size of the transistors in the inverting buffer larger. This will increase the drive capability of the circuit. Increasing the size of the output inverter increases the delay time; this leads to a tradeoff between speed and the internal noise margin.

4.5.2 P-CHANNEL FEEDBACK TRANSISTOR

Another method to alleviate the charge redistribution problem is to place an additional P-channel transistor, Mf, in parallel with the precharge transistor [23]. As illustrated in Fig. 4.20, the inverter-transistor combination acts as a latch and replenishes any charge that is lost from the precharged node due to charge redistribution. The feedback transistor, Mf, serves as a load transistor when node N1 is forced to ground. Therefore, the operation is now that of a ratioed circuit. The aspect ratio (Wf/Lf) of Mf has to be selected such that the effective gain of M1 to M6 in series is significantly larger than the gain of Mf.

Another impact of the feedback device is on performance. The

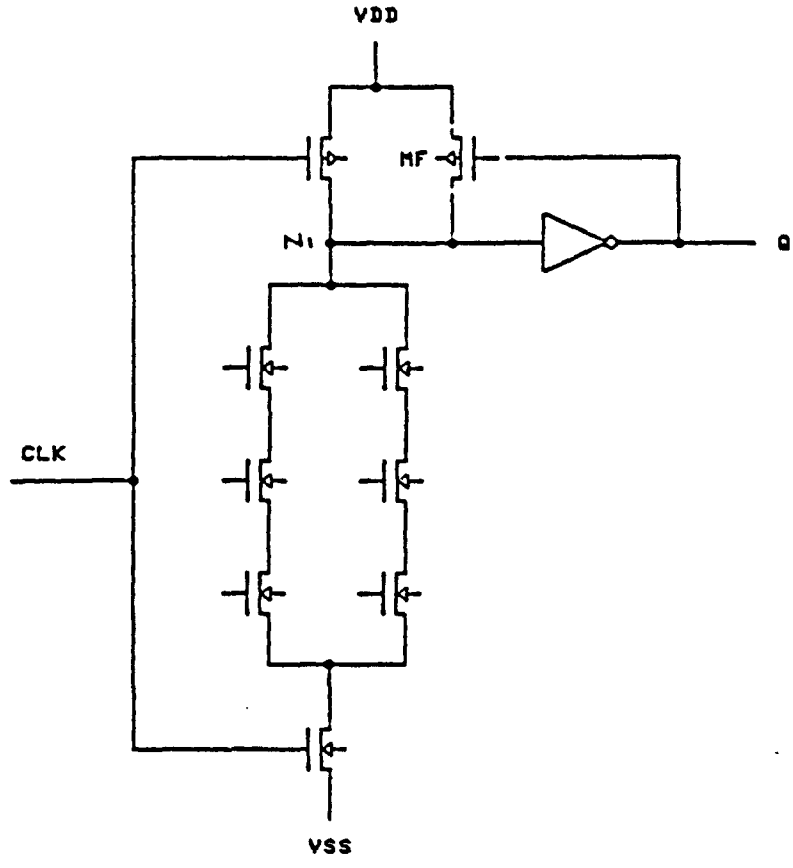


Fig. 4.20 A Domino Circuit Using PMOS Feedback Device to Reduce Charge Sharing Effect

delay increases as a larger feedback device (large gain) is used. Transistor Mf also helps alleviate transient noise problem.

4.5.3 MULTIPLE PRECHARGING OF INTERNAL NODES

A very simple and effective way to alleviate the charge redistribution problem is to precharge the internal nodes. Figure 4.21 illustrates this technique applied to a complex AND/OR gate. The precharge devices Mp2 and Mp3 can have minimum channel length and width since they are used to precharge a small internal capacitance. Multiple precharging of the internal nodes can eliminate the charge sharing problem at the expense of extra devices which increase layout area and routing complexity in the circuit.

4.5.4 THE USE OF NOR BUFFER

A Domino gate with large fan-in (e.g. 8-input AND) contains a large number of NMOS transistors in series. This results in a severe charge sharing problem and a long gate delay. A novel method of designing Domino gate with large fan-in has been presented in [24]. Figure 4.22(a) shows an 8-input AND gate. This circuit does not function properly since there are too many NMOS devices in series. Fig. 4.22(b) shows the 8-input AND gate using a 2-input NOR buffer. As we can see, by splitting the 8 series transistors into two separate branches of 4 transistors and using a NOR2 gate instead of an inverter, a high-speed Domino AND8 becomes realizable. The noise margin will improve with increasing fan-in. A 16-input Domino AND gate is realizable by using a NOR4 buffer.

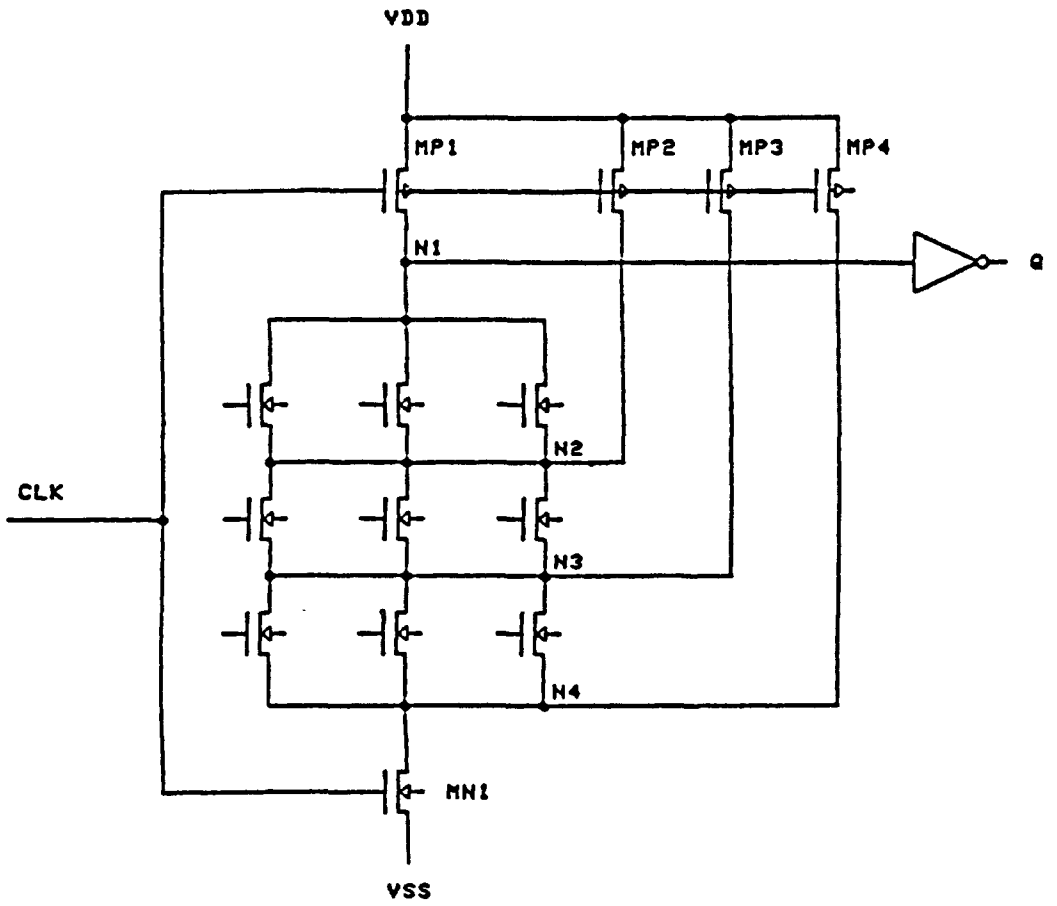


Fig. 4.21 A Domino Circuit Using Multiple precharging Devices to Prevent Charge Sharing

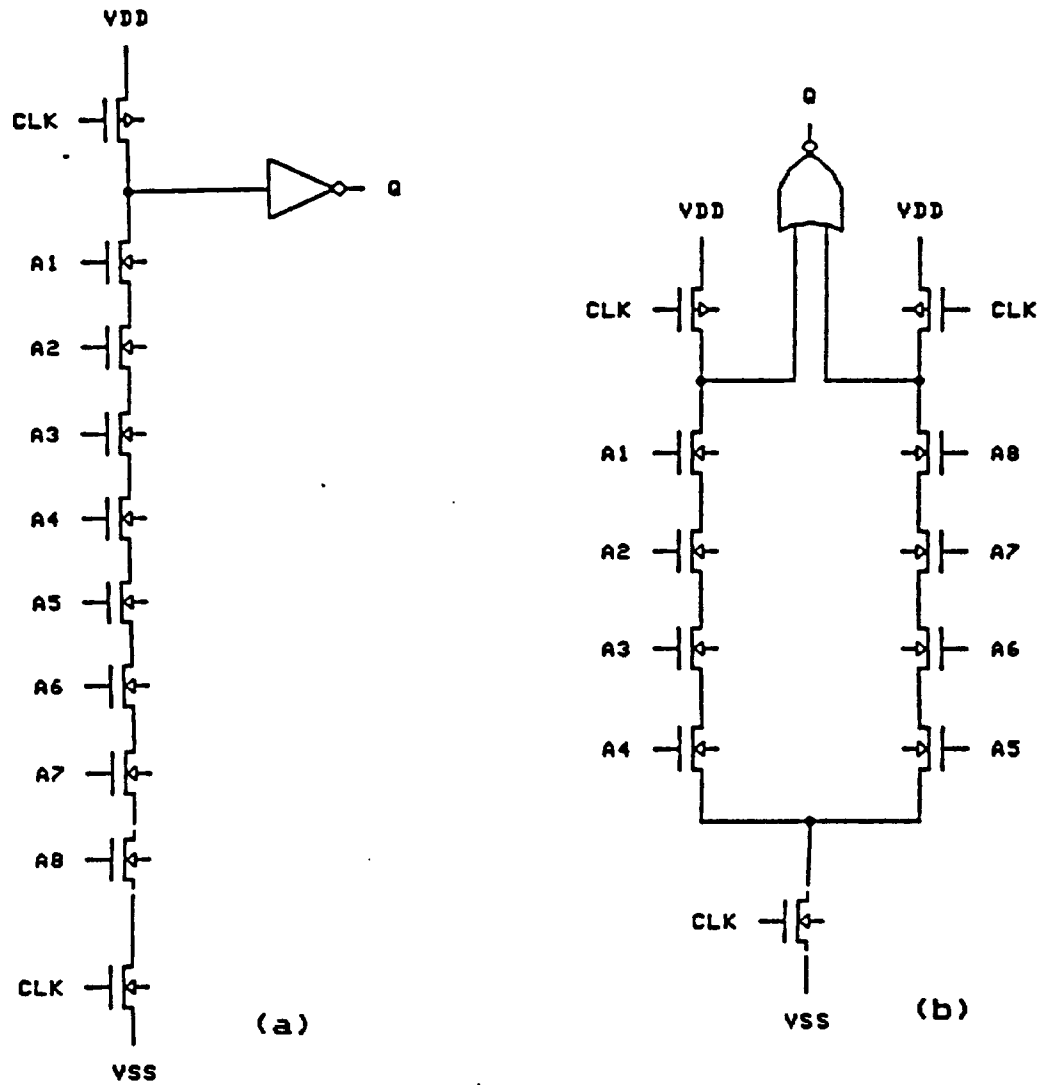
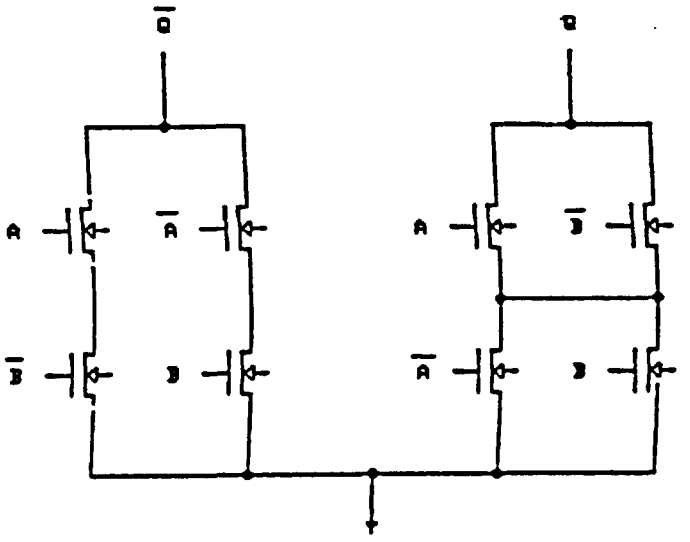


Fig. 4.22 (a) A Domino 8-Input AND Gate
(b) NOR Buffered 8-Input AND Gate

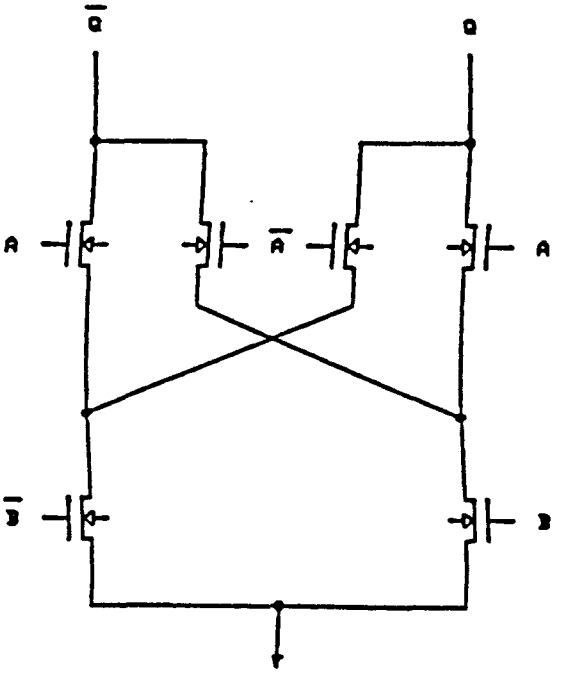
4.6 DESIGN OF DIFFERENTIAL CASCODE VOLTAGE SWITCH TREE

Differential cascode voltage switch (DCVS) trees is the main part of any DCVSL circuits. It consists of two complementary NMOS switches (binary decision trees). The differential Cascode Voltage Switch (DCVS) Tree is designed such that (1)When the input vector X = (x1,....xn) is the true vector of the switching function f(x), node N1 is disconnected from ground and node N2 is connected to ground by a unique conducting path through the tree; (2) When X = (x1,.....xn) is the false vector of f(x) the reverse holds [25]. Fig. 4.23(a) shows the basic form of a two-input DCVSL XOR gate. The DCVSL trees can be further minimized by using logic minimization algorithms. Fig. 4.23(b) shows a different tree structure for the same function. Note that only 6 NMOS transistors are required while in the previous design (Fig. 4.23 (a)) 8 devices are needed. The functionality of this circuit can be verified by trying all the possible combinations of the inputs. Since between a boolean expression and a DCVS tree structure a unique one-to-one correspondence does not exist, a logic function might be realized with several different tree structures.

In this section we are going to explain a simple and practical method for constructing DCVS trees [25]. This procedure utilizes a Karnaugh mapping technique and is a very efficient method for functions of up to five variables. Consider the carry-out function of a full adder which is given by the following boolean function



(a)



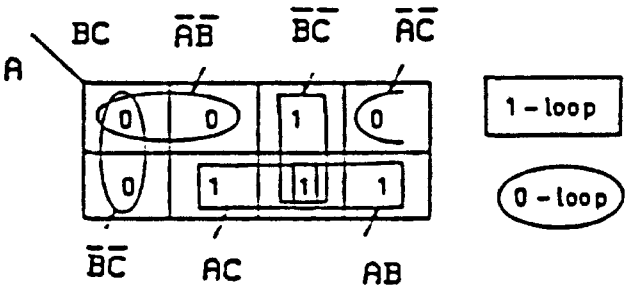
(b)

Fig. 4.23 (a) DCVS 2-Input XOR
(b) Logic Minimized DCVS 2-Input XOR

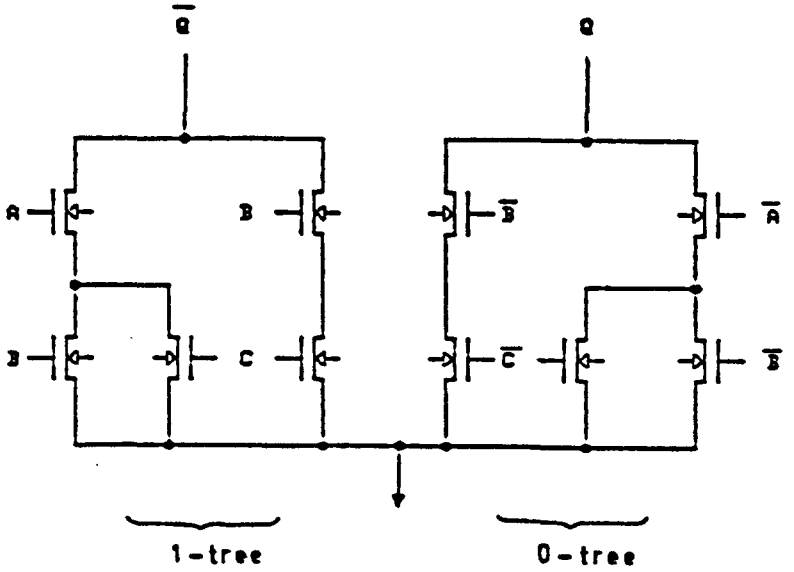
Co = A B + B Ci + A Ci

To construct the corresponding DCVS tree we start with the Karnaugh map, shown in Fig. 4.24(a). The minimal cover for 1 and 0-cells are shown in the K-map. The resulting DCVS trees of this function is illustrated in Fig. 4.24(b). The tree attached to node N1 (carry-out complement) is derived from the 1-cells and is called the 1-tree. Similarly, the tree connected to N2 (carry-out) is derived from 0-cells and is called 0-tree. Since the 1-cells and 0-cells have been grouped separately, the 1- and 0-trees are disjointed. This tree structure shows that 10 NMOS devices are required to implement the carry-out function. The K-map procedure can be used to explore the maximum commonality between the two trees. This leads to the minimization of the device count.

The 10-cell (01-cell) is defined as the cell obtained from grouping of a 1-cell (0-cell) and a 0-cell (1-cell). The K-map of the previous example is shown in Fig. 4.25(a) which has three types of loops, namely 0-, 1-, and 10-loops. The DCVS tree resulting from this K-map consists of a shared tree corresponding to 10-loops and two branches corresponding to the 1- and 0-loops (Fig. 4.25(b)). Note that in this circuit only 8 NMOS transistor are required, which is two devices less than Fig. 4.24(b). Generally the tree sharing results in a reduction of the number of devices.

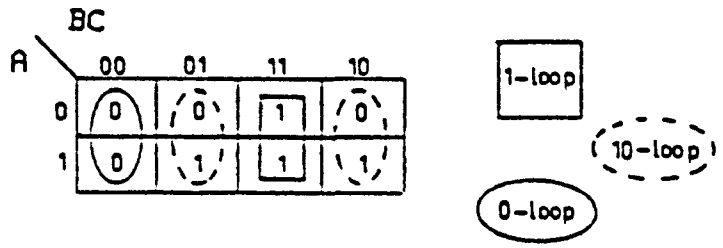


(a)

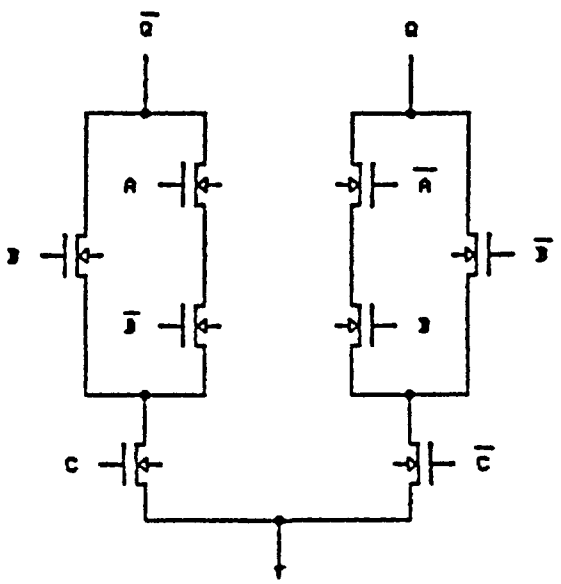


(b)

Fig. 4.24 (a) K-map Of the Carry-Out Function of a Full Adder
(b) DCVS Implementation of the Carry-Out Function of a Full Adder



(a)



(b)

Fig. 4.25 (a) K-map of the Carry-Out Function of a Full Adder (Different Encirclement)
(b) The DCVS Tree of the Carry-Out Resulting From Fig. 4.25(a)

4.7 ANALYSIS AND OPTIMIZATION OF THE SSDL GATE

To analyze the electrical behavior of the SSDL circuit , we replace the DCVS tree by its simplified equivalent circuit, Figure 4.26. As we know, there should be a unique path from either node N1 or N2 to ground. We refer to this path as the ON-SIDE of the switch logic. The other node is disconnected from the ground by the OFF transistors. We refer to this as the OFF-SIDE of the switch logic. The analysis of the SSDL gate will be considered for the worst case operation which happens when:

(I)- The node which has to be pulled down (high) is at high (low) voltage.

(II)- All the inputs to the off-side are high except the last one, which is low

$B_i = 1$ for $i = 1, 2, 3, \ldots, k-1$

$B_k = 0$

4.7.1 DERIVATION OF THE DELAY TIME

The three components of the delay time, which may determine the maximum clock frequency of the SSDL gate, are:

(I) - Charging time of the node N1 (which has no path to ground)

(II) - Partial-charging time of the node N2 (which has a path to ground)

(III)- Latch-up time of the sense amplifier

The first two components of the delay belong to the sample-phase and the third one belongs to the set-phase of operation. The clock phase should be symmetrical to allow

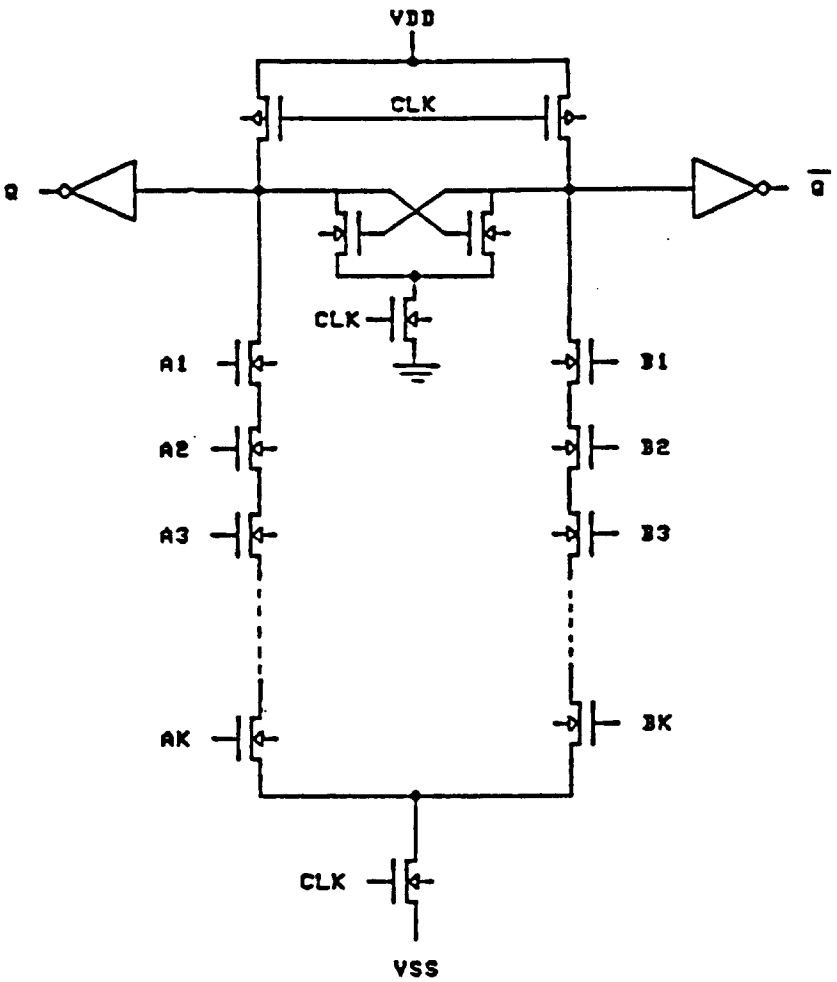


Fig. 4.26 Equivalent Circuit of SSDL Gate

different stages of the SSDL gate to be cascaded efficiently. Therefore only the dominant delay component (the slowest one) determines the maximum clock frequency of the gate operation. The first component of the delay time, which is the delay due to precharging of all parasitic capacitances, is slower than the second component of the delay. Therefore the sample-phase is analysed for the determination of this delay.

4.7.1.1 SAMPLE PHASE OF OPERATION

During the sample phase the Off-side of the switch logic is not conducting and therefore the corresponding node (N2 in our analysis) will be charged up to the supply voltage. Fig. 4.27(a) shows the equivalent circuit of the Off-side during the precharge for the worst case. In this case the precharge transistor Mp2 has to charge all the parasitic capacitances CN2 ,C1 , C2 , ... The RC equivalent circuit of this circuit is shown in Fig. 4.27(b).

To measure the precharging time of the voltage at node N2 we use the definition of the delay [26] i.e. we take the first-order moment of impulse response (inertia) as the delay.

$$Td = \int_0^{\infty} h(t)\,t\,dt \qquad 4\text{-}1$$

From the extended Elmore's definition of the delay [27], the delay of the RC network is:

$$\overrightarrow{Td} = \overrightarrow{R}\,\overrightarrow{C}\,[\,V(\infty) - V(0)\,] \qquad 4\text{-}2$$

Considering figure (4b) the charging time for node N2 is

$$Td1 = Rp2\,\{\sum_{i=1}^{K} Ci\,[\,Vi(\infty) - Vi(0)\,]\,\} \qquad 4\text{-}3$$

{Vi(0)} are the initial conditions of the voltages which will be assumed zero. {Vi(∞)} are the steady state value of the voltages. See Appendix (D) for more detail on Vi(∞).

4.7.1.2 SET-PHASE OF THE OPERATION

At this phase of the operation Mn1, Mp1 and Mp2 will turn off and Mn4 starts conducting; this activates the sense amplifier. The voltage imbalance on nodes N1 and N2 acts as an initial condition for the cross coupled differential amplifier. This inequality of the voltages will cause one of the cross coupled transistors to draw more current than the other. The positive feedback in the regenerative switch helps the lower voltage node to be pulled down even lower . The voltage imbalance will grow until it becomes comparable to supply voltage. The objective is to design this flip-flop so that there is no voltage excursion on the higher voltage node, while the lower voltage node discharges to ground.

The schematic of the sense amplifier is shown in Fig. 4.27(b) In the analysis of the SSDL circuit during the sample phase we have shown that one of the nodes (e.g. N2) will be pulled up to Vdd and the other one (N1) will be at a voltage less than Vdd (Vdd - V). By the time the clock goes high, its transition will be coupled through coupling capacitances existing between the gate and source of the transistors Mp1 and Mp2 . The effect is the addition of charges on nodes N1 and N2. Due to the symmetrical structure of the SSDL circuit the load capacitances on node N1 and N2 are equal so are the corresponding coupling capacitances. As

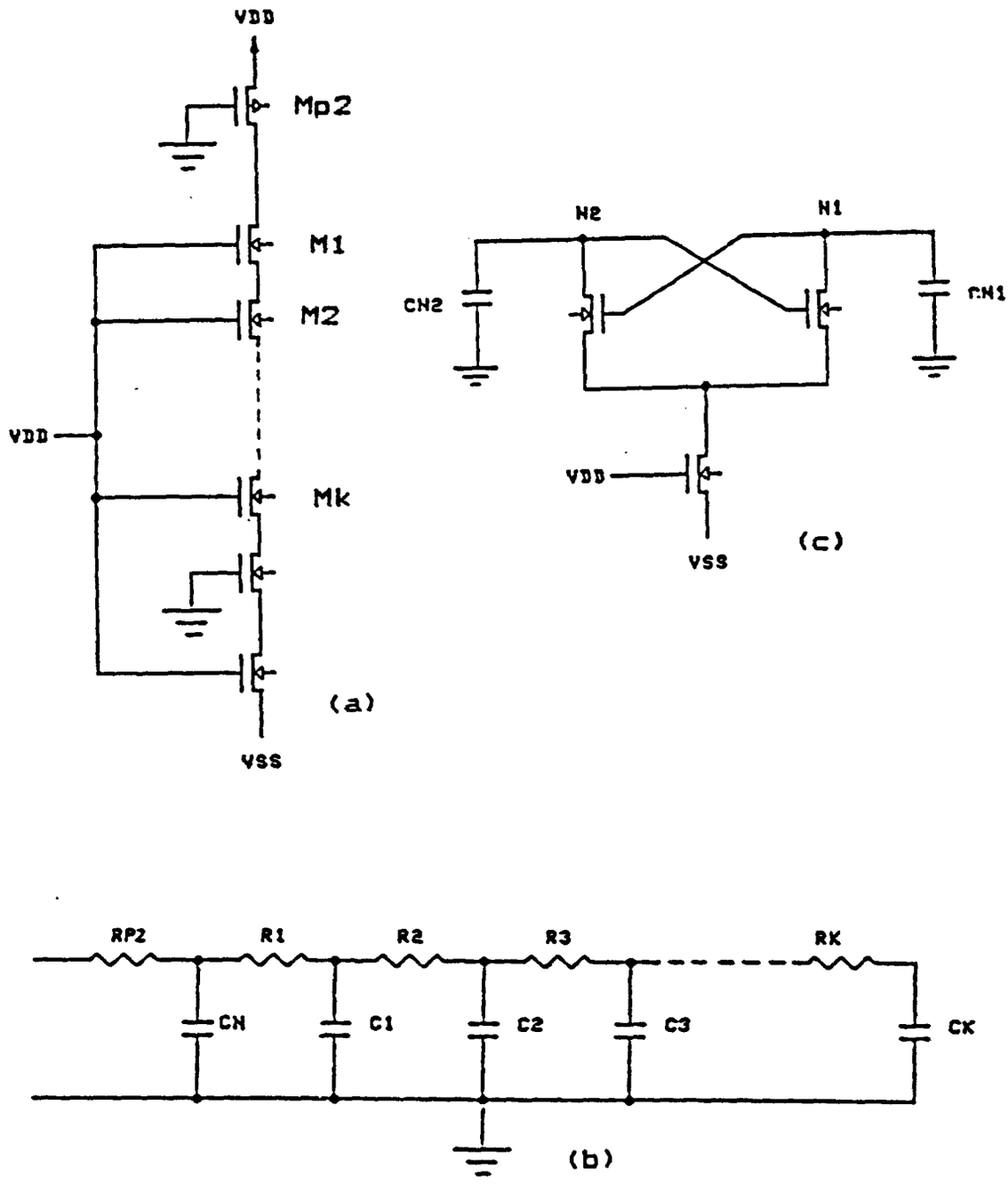


Fig. 4.27 (a) Equivalent Circuit of the Off-Side During Sample Phase
(b) RC Equivalent Circuit of the Off-Side During Sample Phase
(c) Equivalent Circuit of the Sense Amplifier During Set Phase

a result both nodes (N1 and N2) will experience almost the same amount of voltage increase.

The two components which are effective for this capacitive coupling are Cov(p), overlap capacitance of Mp and the effective gate-to-drain capacitance of a non saturated PMOS transistor. These are

$$Cov(p) = Wp\ LDp\ Cox$$

$$Cgd(p) = Wp\ (\ Lp - 2\ LDp)\ Cox$$

Where Wp and Lp are width and length of the gate of Mp and LDp is the lateral diffusion of PMOS transistors.

$$Vinc = (\ Vdd - V\ )\ \frac{Cov(p1) + Cgd(p1)}{CN + Cov(p1) + Cgd(p1)}$$

After a low to high transition of the clock, voltages on nodes N1 and N2 are:

$$VN1 = Vdd - V + Vinc = VO$$

$$VN2 = Vdd + Vinc = VO + v$$

This difference between the voltages of nodes N1 and N2 will be amplified by the latch sensor. The final unbalanced differential voltage will be comparable to the supply voltage . To maximize the final latched imbalance voltage, the off-side of the sense amplifier should not conduct any current . The optimum latching waveforms [28] are given by

$$Vs(t)=VO - Vtn- \frac{(B/2CN)v^2 f^2 t}{1-(B/2CN)vf^2t} \qquad \text{for } t < tsat.$$

and

$$Vs(t) = VO + v - Vtn - \frac{Vtn}{2f}\ [3 + Exp(\ \frac{t - tsat.}{T1}\ )$$

$$\text{for } t > Tsat.$$

where

$$f = \frac{CN}{CN + Cgn}$$

$$T1 = \frac{CN}{Bf^2Vtn}$$

$$tsat. = \frac{2CN(Vtn - fv)}{Bf^2Vtn\ v} \qquad 4\text{-}4$$

Where Cgn, B and Vtn are the gate capacitance, gain factors and threshold voltages of NMOS devices; and CN is the capacitance of the nodes N1 and N2. The latch time is given by

$$t1 = tsat. + \frac{CN}{f\ B\ Vtn} Ln \left[\frac{(V0 + v - Vtn)f - Vtn/2}{Vtn/2}\right] \qquad 4\text{-}5$$

4.7.2 DELAY OPTIMIZATION OF THE SSDL GATE

The analytical expressions of the delay times can be applied to the optimization of the SSDL gate. As was mentioned previously the slowest component of the delay determines the maximum clock frequency. Since SSDL circuits are useful for complex logic circuits, which require large fan-in, the sample-phase in these circuits are slower than the set phase. Therefore the expression of precharging time in the sample-phase is used as a measure of the SSDL gate delay time.

One technique that has been demonstrated to decrease the delay time of Domino gates is by scaling the NMOS transistors so that devices closest to the output are smallest, with transistors increasing in size from output to

ground [29]. The same procedure can be followed for the delay time optimization of the SSDL gate, where the channel width of the transistors are variables in the objective function. Assuming the transistor(s) closest to the output (level 0) has channel width W0, and all the transistors in the i-th level have channel width Wi such that:

$$Wi = W0\ (1 + i.k) \qquad 4\text{-}6$$

Where k is the gradient of the NMOS transistors. The parasitic capacitances and NMOS transistors' on-resistances are functions of the channel width of each transistor. The delay time of equation 4-3 can be written as:

$$Td = f(k) \qquad 4\text{-}7$$

By using any single variable optimization technique one can find the optimum value for k.

4.7.3 NOISE MARGIN OF THE SSDL CIRCUIT

The high noise margin of the SSDL gate has two components. One is due to the dynamic DCVS tree and the other one is due to the static inverter. The high noise margin of the static CMOS inverter are derived in [30]

$$NMH = Vdd - VIH$$

$$VIH = \frac{b - \sqrt{b^2 - 4ac}}{2a}$$

Where

$$Br = Bp/Bn$$

$$a = 3 - 2\ Br - Br^2$$

$$b = -8[Vtn0 + Br\ (Vdd + Vtp0)] - 2\ (1-Br)[\ Br\ (Vdd + Vtp0) - Vtn0\ ]$$

$c = 4\ Vtn0^2 - 4\ Br\ (Vdd + Vtp0)^2$

Bn and Bp are the NMOS and PMOS transistors' gain.

During the sample phase, one of the internal nodes (N1 or N2) is precharged to a value less than Vdd. The internal noise margin of that node is:

$INMH = NMH - D\ (VN)$

Where D (VN) is the voltage difference of the supply voltage and the voltage of the partially precharged node which is obtained in the next section.

4.7.3.1 CALCULATION OF D (VN)

A simplified circuit diagram of the On-side of the switch logic is shown in Fig. 4.28(a). All the inputs to the switch logic are high and the transistor Mn1 is also ON so that node N1 will be charged up to a voltage which is less than Vdd. All the transistors operate in the non saturation region except M1 which is in saturation[36]. The nonsaturated devices are replaced by an RC network (see Appendix E). Where

$Rp1 = 1/[Bp\ (Vdd - |Vtop|)]$

$Rni = 1/[Bn\ (Vdd - Vtn(av))] \qquad i = 1, K$

The equivalent model of this circuit is shown in Fig. 4.28(b). CN1, C1, C2 ,..... and CK are the parasitic capacitances (drain and source diffusion, wiring and gate capacitances) on nodes N1, 2 , 3, ... , and K respectively.

In steady-state there is no current flowing through the

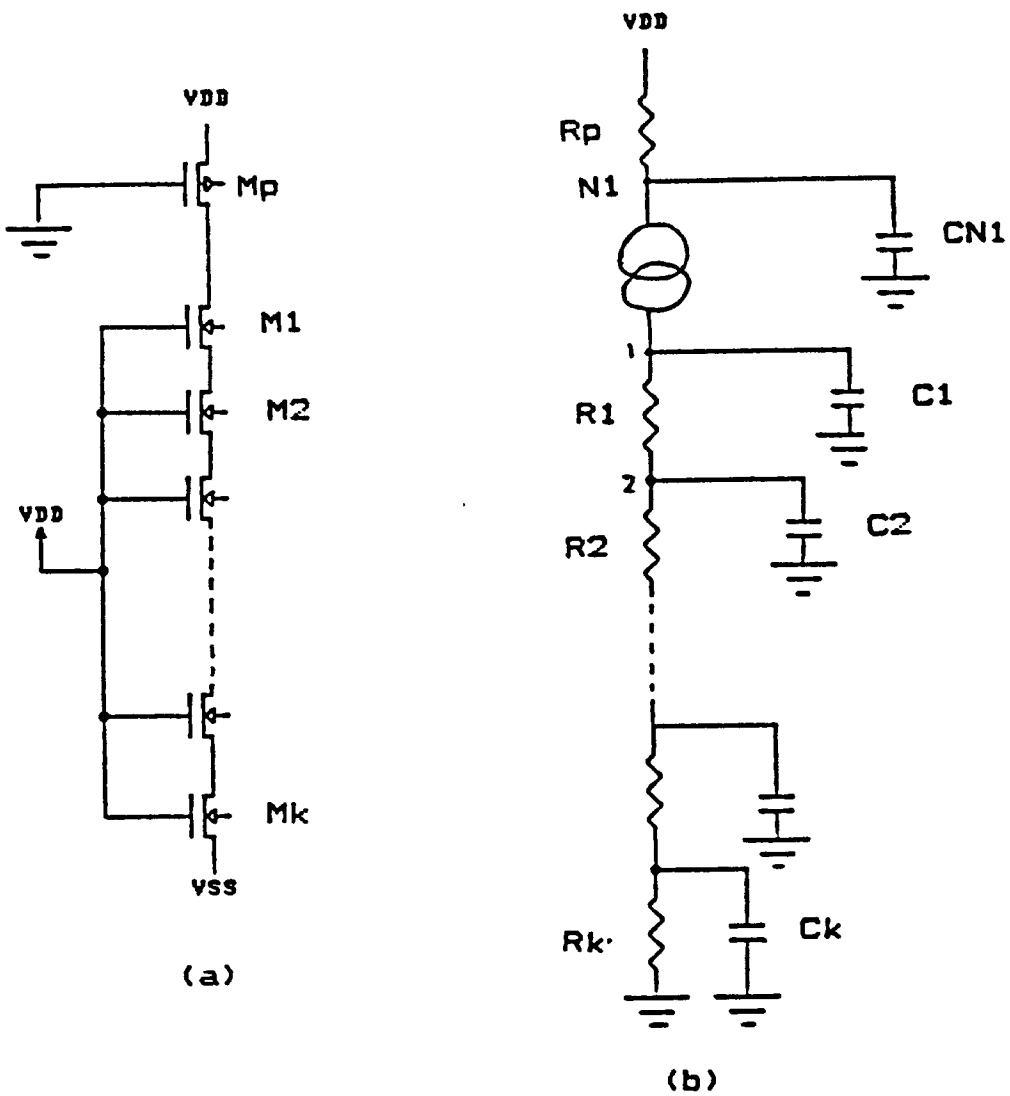


Fig. 4.28 (a) Equivalent Circuit of the On-Side During Sample Phase
(b) RC equivalent circuit of On-side During sample phase

parasitic capacitances and the circuit of Fig.4.28(a) can be further simplified as shown in Fig. 4.28(b). Req is the on-resistance of the series combination of nonsaturated transistors M2 to Mk.

Applying Kirchoff's current law (KCL) at node N1 and 1:

$$\frac{V1}{2} = \frac{Bn}{2}\left[Vdd - V1 - Vtn(av) \right]^2$$

$$\frac{\left[Vdd - Vtn(av) \right]}{K} V1 = \frac{Bn}{2}\left[Vdd - V1 - Vtn(av) \right]^2 \quad 4\text{-}8$$

$$Vm = Vdd - Vtn(av)$$

Solving the quadratic eqn. 4-8 for V1 we get :

$$V1 = Vm\left[(1 + 1/K) - \sqrt{(2/k + 1/k)} \right]$$

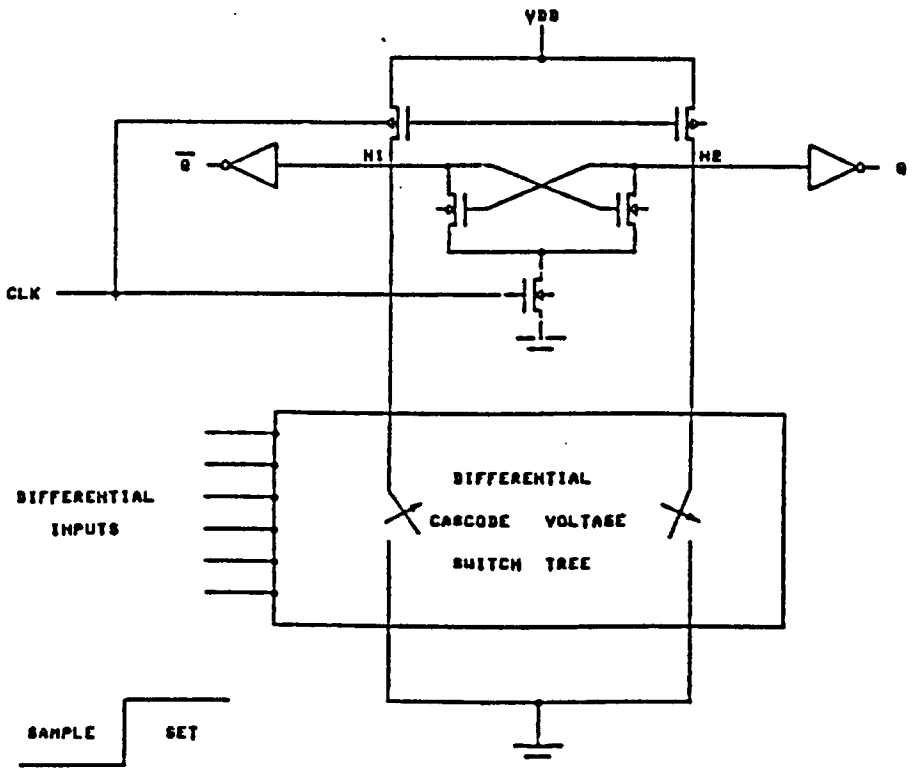
Applying KCL at node N1 we obtain

$$VN1 = Vdd - V1(Rp/Req)$$

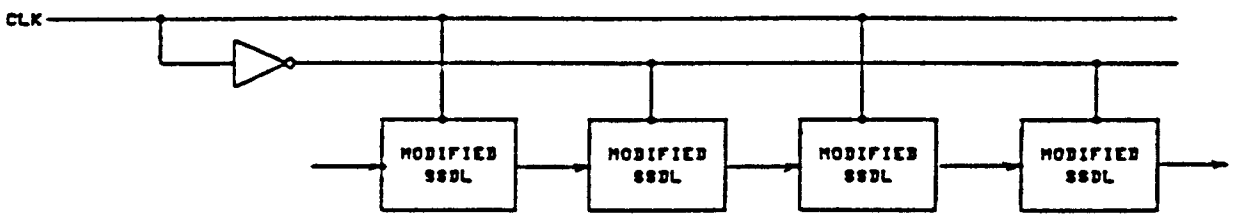
$$D(VN) = V1(Rp/Req)$$

4.7.4 APPLICATION TO PIPELINED ARCHITECTURES

Pipelining allows a significant increase in the throughput rate with only moderate increase in hardware. In a pipelined structure successive stages are interfaced with data latches. The SSDL circuit has a built-in flip-flop which makes it very suitable for the realization of pipelined systems. Alternate stages of the pipelined structure are clocked by clk and clkb respectively. Fig. 4.29(b) shows the SSDL pipelined system. In this structure, since the inputs are guaranteed to be zero during the set-phase and are



(a)



(b)

Fig. 4.29 (a) Modified SSDL Circuit
(b) Modified SSDL CMOS Pipelined Circuit

stable during the sample-phase, the ground switch (transistor Mn1) may be omitted, Fig. 4.29(a). In this way each stage is fed only by clk or clkb which results in a saving in area (less transistors and runners) and also an improvement in speed (less number of transistors in series).

4.8 TESTING

4.8.1 INTRODUCTION

The testing of SSI and MSI digital circuits with up to 100 logic functions is not a critical factor. A complete functional test (Exhaustive testing) of a combinational network with N inputs requires 2^N patterns. With SSI and MSI this test vector is obtainable and exhaustive testing is feasible. With LSI and VLSI circuits, however, complete testing is often neither possible nor cost effective. Consider for example a network with N=60 (e.g. a 30-bit adder). To test this circuit exhaustively a sequence of 2^{60} inputs must be applied. Assuming one had the patterns and applied them at a rate of 1us per pattern, the testing will take about 36 thousand years! Clearly design for testability is an important criterion for LSI/VLSI design [12].

At the present time, testing techniques are generally based on using an appropriate model that can adequately represent all physical failures in a particular technology. The most popular fault model, which is accepted throughout the industry, is the Stuck-At Fault model. This model assumes that a logic gate input or output is stuck-at-0 or stuck-at-1 for faulty gates. Fig. 4.30(a) shows a gate with input

A stuck-at-0. The short circuit, S1, is modeled by a stuck-at fault at input A. In the single stuck-at fault model, one assumes that the faulty machine will have either the stuck-at-0 or stuck-at-1 fault (not both faults at the same time). Experience has shown that the single stuck-at fault model, for prior technologies, has been adequate. The problem with CMOS is that there are a number of faults that cannot be modeled by stuck-at-1 or stuck-at-0 techniques [31]. Considering the S2 fault in Fig. 4.30(a), it cannot be modeled by single stuck-at fault model. Also there are some faults in CMOS (S3 in Fig. 4.30(b)) that changes a combinational circuit into a sequential circuit. Therefore, it is still unknown whether the single stuck-at fault model is adequate for CMOS LSI/VLSI circuits or not.

There are three main areas in LSI/VLSI testing procedures:

I - Test generation

II - Test verification

III - Design for test

Test generation for a chip involves generating a test pattern, as short as possible, to verify the behavior of the chip. The problem of test verification is concerned with proving that a test vector is effective. Design for testability is discussed in the next section.

4.8.2 DESIGN FOR TESTABILITY

When designing for testability, the key concepts are controllability and observability. Controllability is the ability to initialize the internal nodes into specific

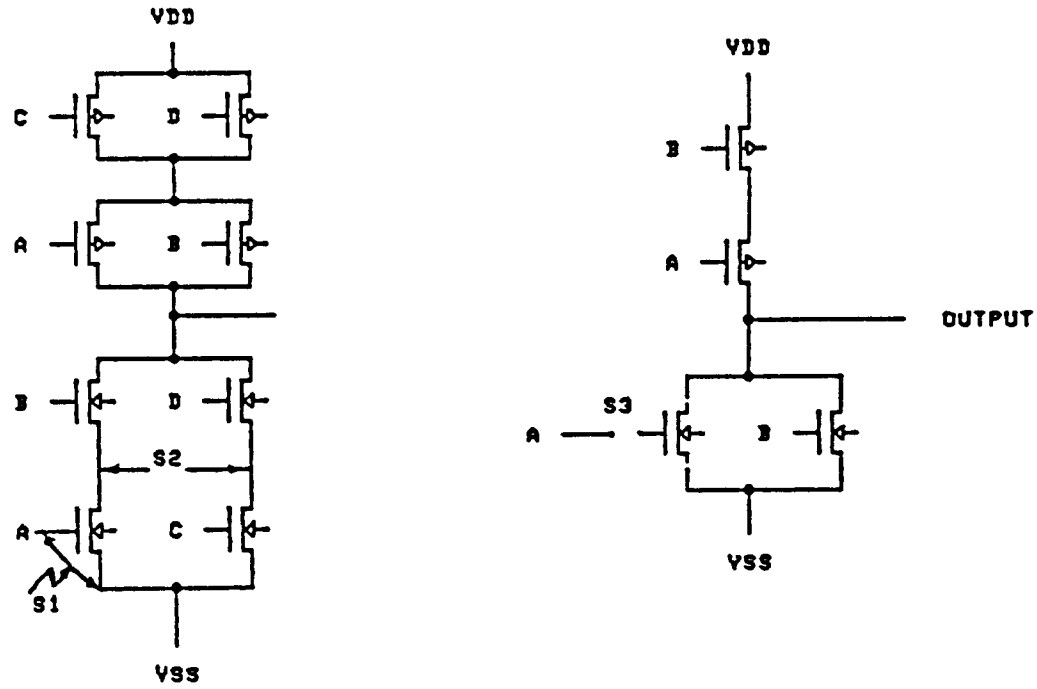


Fig. 4.30 (a) Fault Example in CMOS Gate
(b) Fault in CMOS Circuit

controlled states. Observability defines the ability to observe the states of the internal nodes of the circuit. The design for testability techniques are divided into three categories:

I - Ad hoc approach

II - Structured design approach

III - Self-testing and built-in tests

The first category, ad hoc technique, solves the problem for a specific design and is not general. A number of techniques that fall under this category are:

- Partitioning
- Test point
- Bus architecture
- Signature analysis

The second category is structured design for testability. "Most structured design practices are built upon the concept that if the values in all the latches can be controlled to any specific value, and if they can be observed with a very straightforward operation then the test generation, and possibly, the fault task, can be reduced to that of doing test generation and fault simulation for a combinational logic network. A control signal can switch the memory elements from their normal mode of operation to a mode that makes them controllable and observable" [12]. The techniques which fall under the structured design for testing are:

- Level sensitive scan design (LSSD)
- Scan path
- Scan/set logic

- Random access scan

Finally, self-test and built-in test has the attributes of both the LSSD network and the scan path network. The following techniques are under this category:

- Built-in logic block operation (BILBO)
- Syndrome testing
- Testing by verifying walsh coefficients
- Autonomous testing

A complete discussion on design for testability is given in [12].

CHAPTER 5

DESIGN AND OPTIMIZATION OF CMOS FULL ADDERS

5.1 INTRODUCTION

In chapter 2, the design of arithmetic operations at the gate logic level was presented and it was shown that a 1-bit full adder is the basic building block of many systems. A number of different techniques for CMOS logic design were also discussed in chapter 4. In this chapter various designs of full adders at the transistor level in CMOS, will be examined.

5.2 CMOS STATIC LOGIC FULL ADDERS

The design of complementary CMOS full adders is straightforward and can be found in any VLSI text book [30]. In this section a number of designs for full adders using transmission gate logic will be presented. Based on the novel transmission gate XOR gate of Fig. 4.4, different implementations of the full adder are possible. To illustrate the operation of the TG full adder let us consider the truth table for the full adder in Fig. 2.2. By examining this table it can be seen that :

I - If $A \oplus B = 1$ then SUM = $\overline{C}$ and CARRY = C

II- If $A \oplus B = 0$ then SUM = C and CARRY = A(or B)

Fig. 5.1 shows the concept of all transmission gate full adders.

By means of four transmission gates, three inverters, and two XOR gates an adder can be constructed, according to Fig.

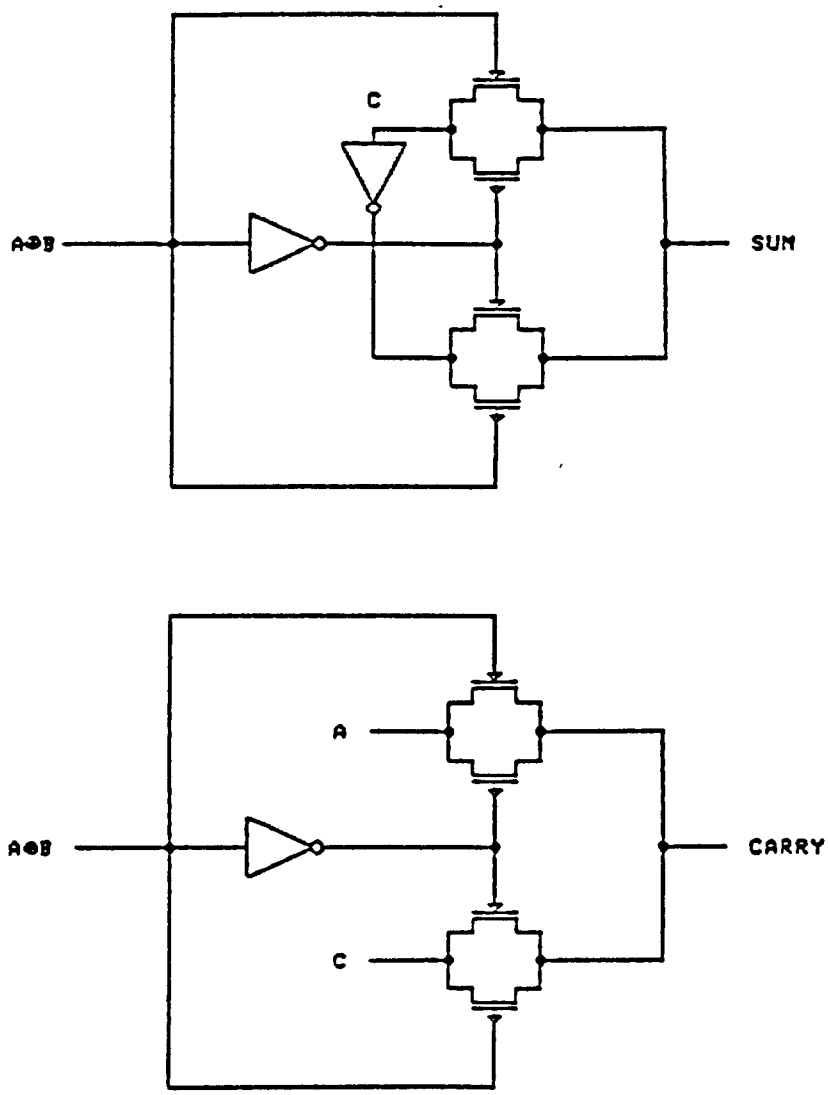


Fig. 5.1 Transmission Gate (TG) Full Adder

5.1. This adder has 24 transistors. The complete transistor schematic of the adder is shown in Fig. 5.2 (TGFA#1).

The two inverter buffers at the outputs of the TGFA#1 can be eliminated and with minor changes in the interconnections, the TG full adder of Fig. 5.3 will result (TGFA#2). This adder has 20 transistors and in the critical path has one inverter less than the previous design.

Fig. 5.4 shows another TG adder (TGFA#3), which has 18 transistors. The inverter for logic inversion of C is deleted and the pseudo inverter at the sum stage performs the inversion when required.

Finally, the TG adder of Fig. 5.5 (TGFA#4) has only 16 transistors. The XNOR section in TGFA#3 is replaced by a simple inverter. This way the number of transistors is reduced at the cost of a reduction in speed because the $A \oplus B$ and $\overline{A \oplus B}$ are not generated in parallel (simultaneously). TGFA#4 is the most compact static CMOS full adder.

All these TG full adders were simulated with SPICE for logic verification and measurement of delay time. See Appendix A for the results of SPICE simulations.

5.3 DYNAMIC CMOS LOGIC FULL ADDERS

In this section new dynamic CMOS full adders using recent techniques for CMOS logic design will be discussed.

A DOMINO CMOS full adder is shown in Fig. 5.6. This type of adder requires both the true and complement inputs for the generation of the true sum and carry output. Domino adders are useful for pipelined structures where the inversion of

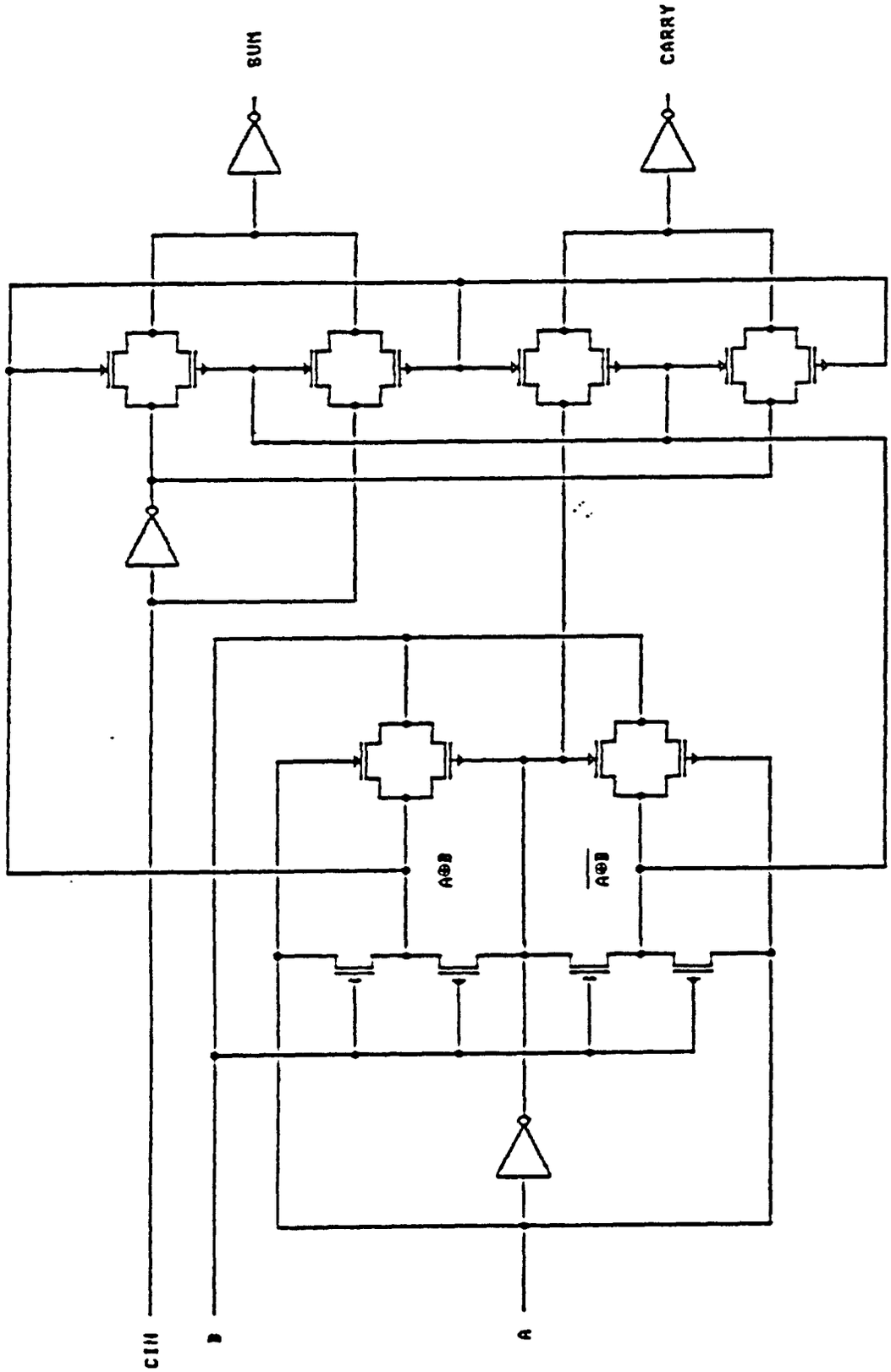


Fig. 5.2 24-Transistor TG Full Adder (TGFA#1)

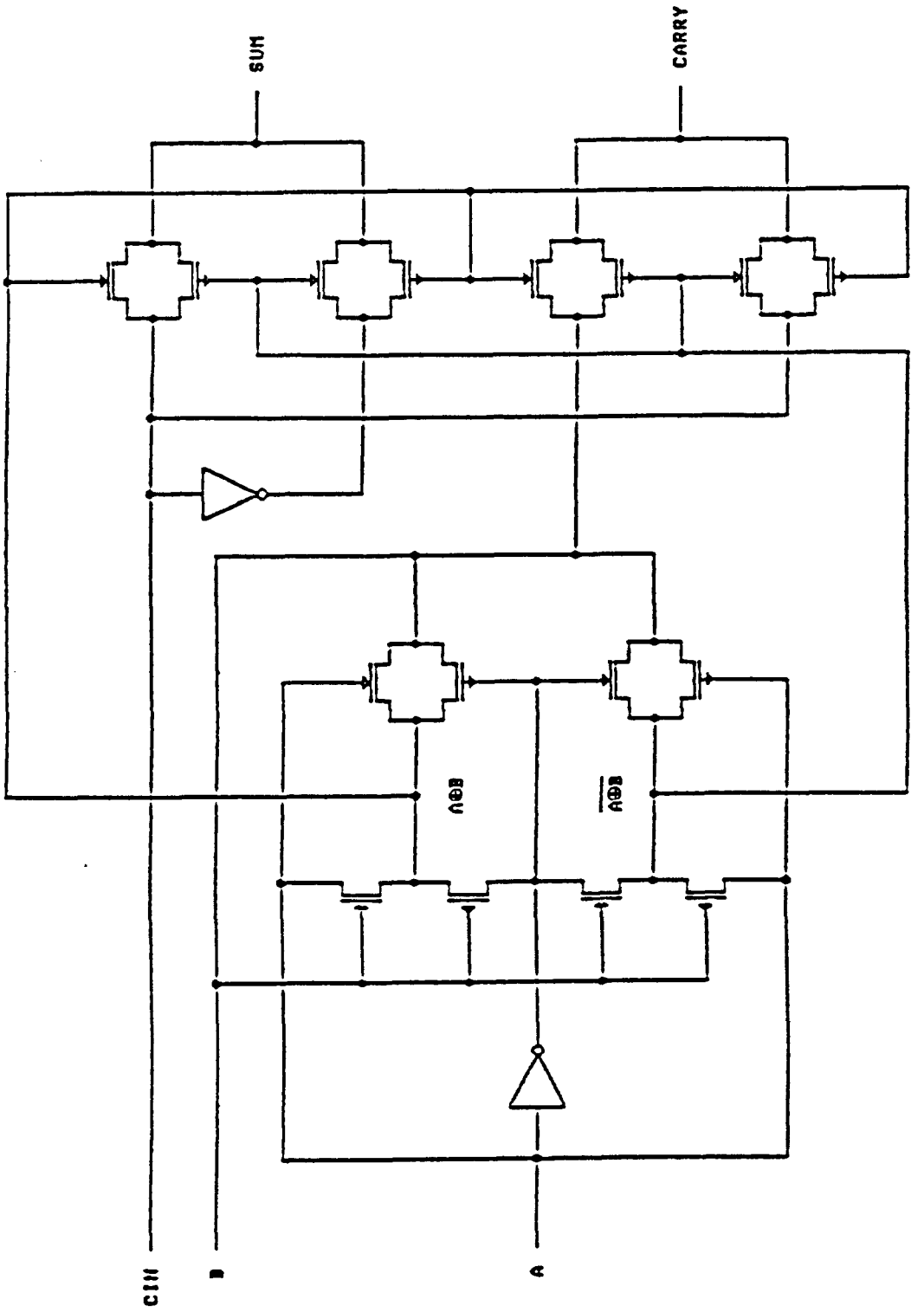


Fig. 5.3 20-Transistor TG Full Adder (TGFA#2)

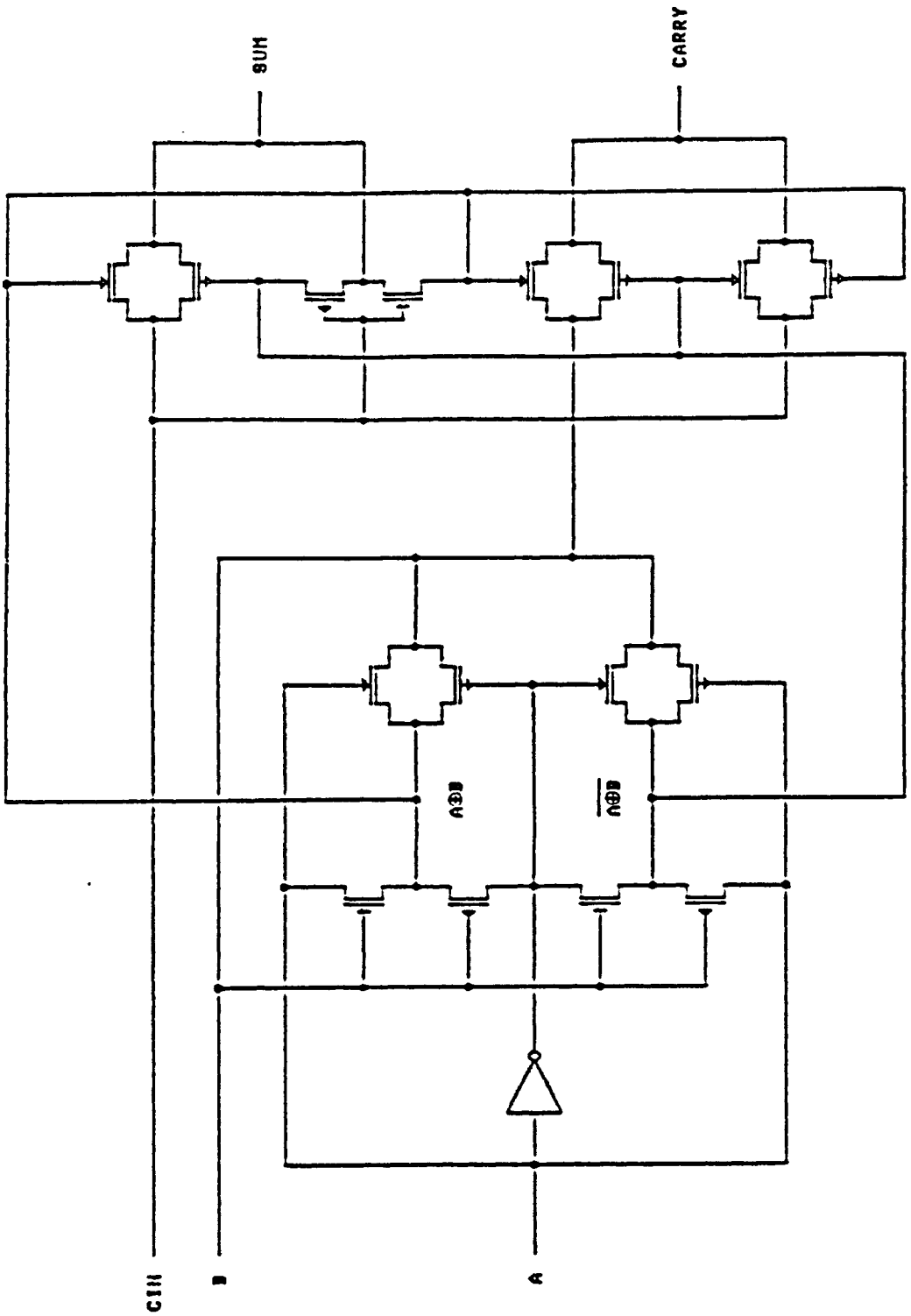


Fig. 5.4 18-transistor TG Full Adder (TGFA#3)

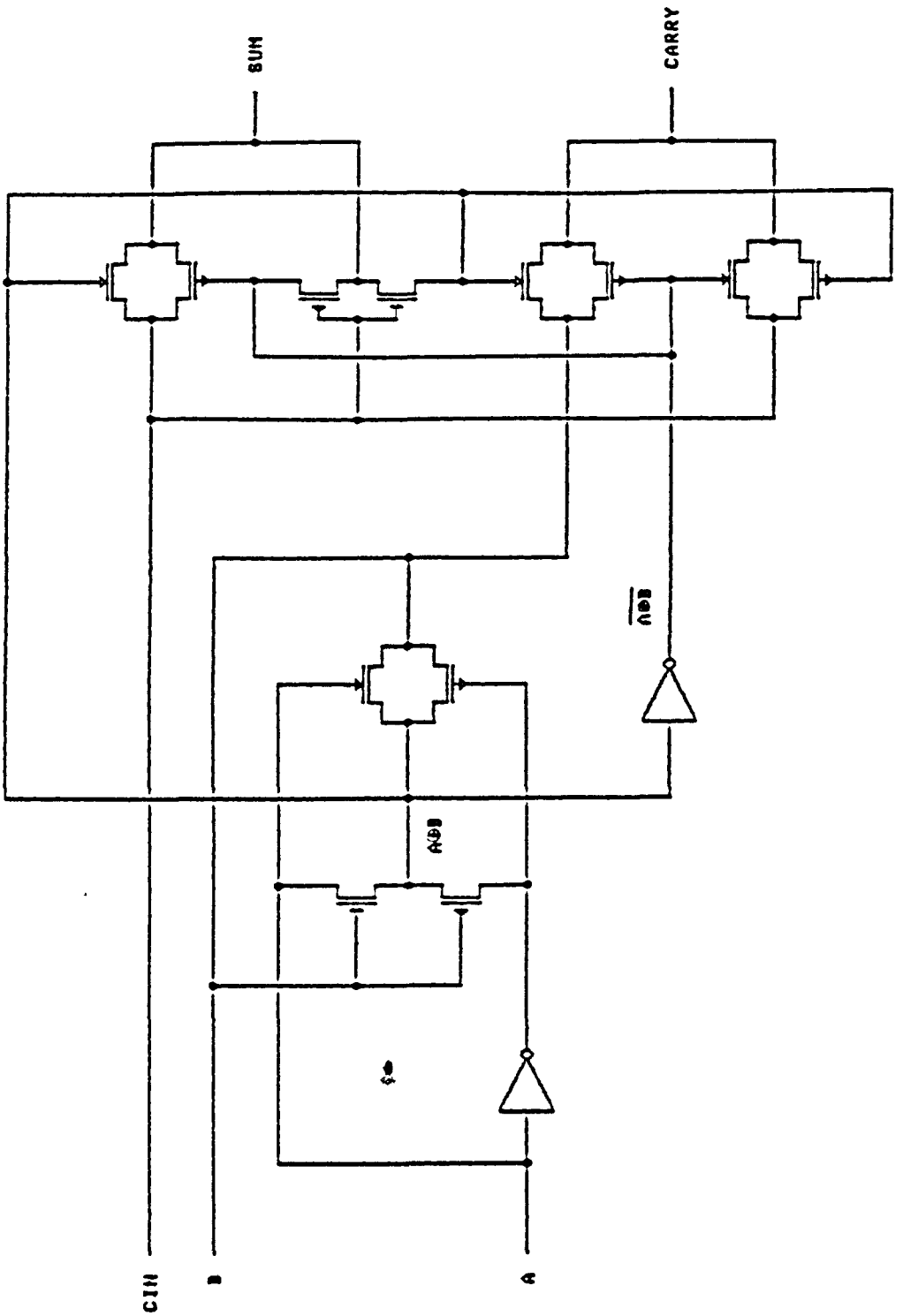


Fig. 5.5 16-Transistor TG Full Adder (TGFA#4)

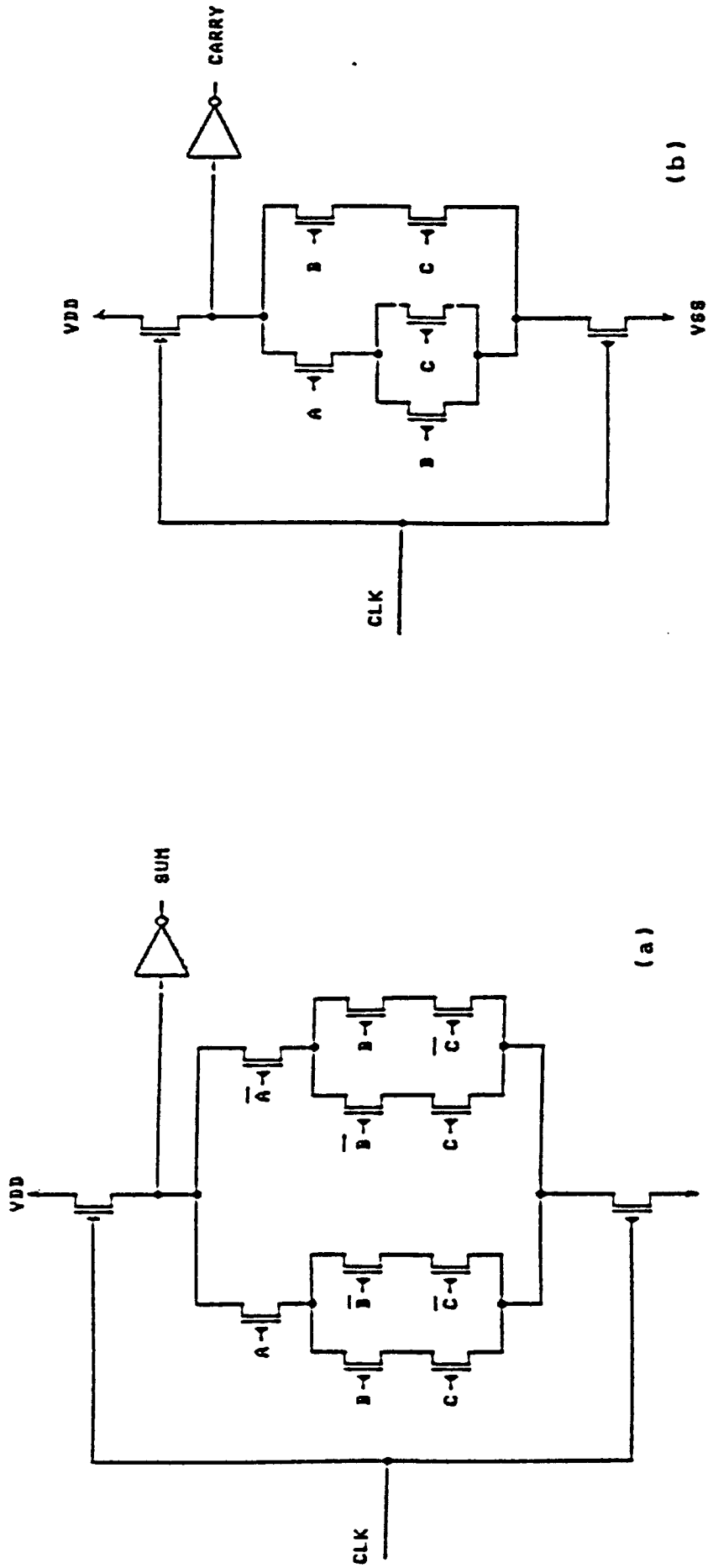


Fig. 5.6 Domino CMOS Full Adder

(a) Sum Section (b) Carry Section

inputs can be performed at the outputs of the latches. For construction of an n-bit ripple-carry adder, n DOMINO full adders are cascaded with a transmission gate switch between each stage. This way inversion is possible at the output of the transmission gates.

A NORA full adder, or N-P dynamic CMOS full adder is shown in Fig. 5.7. The carry stage is made by a P-network and the sum stage is made by an N-network.

Fig. 5.8 shows the transistor schematic of the dynamic DCVSL full adder. Differential inputs are needed to produce the differential outputs. The design procedure for a DCVS tree of sum and carry stages was discussed in chapter 4.

A CMOS SSDL full adder is illustrated in Fig. 5.9. SSDL full adders are very suitable for the design of pipelined arithmetic architectures because of the built-in flip-flop. This adder is faster than the other dynamic CMOS full adders discussed previously.

All the dynamic full adders discussed above have been simulated with the SPICE package (See Appendix A)for logic verification and measurement of circuit speed.
A test chip containing DCVSL and SSDL full adders has been implemented using 3um CMOS technology. This test chip will be used to compare the performance of DCVSL and SSDL full adder. Appendix B shows the layout plot of this test chip.

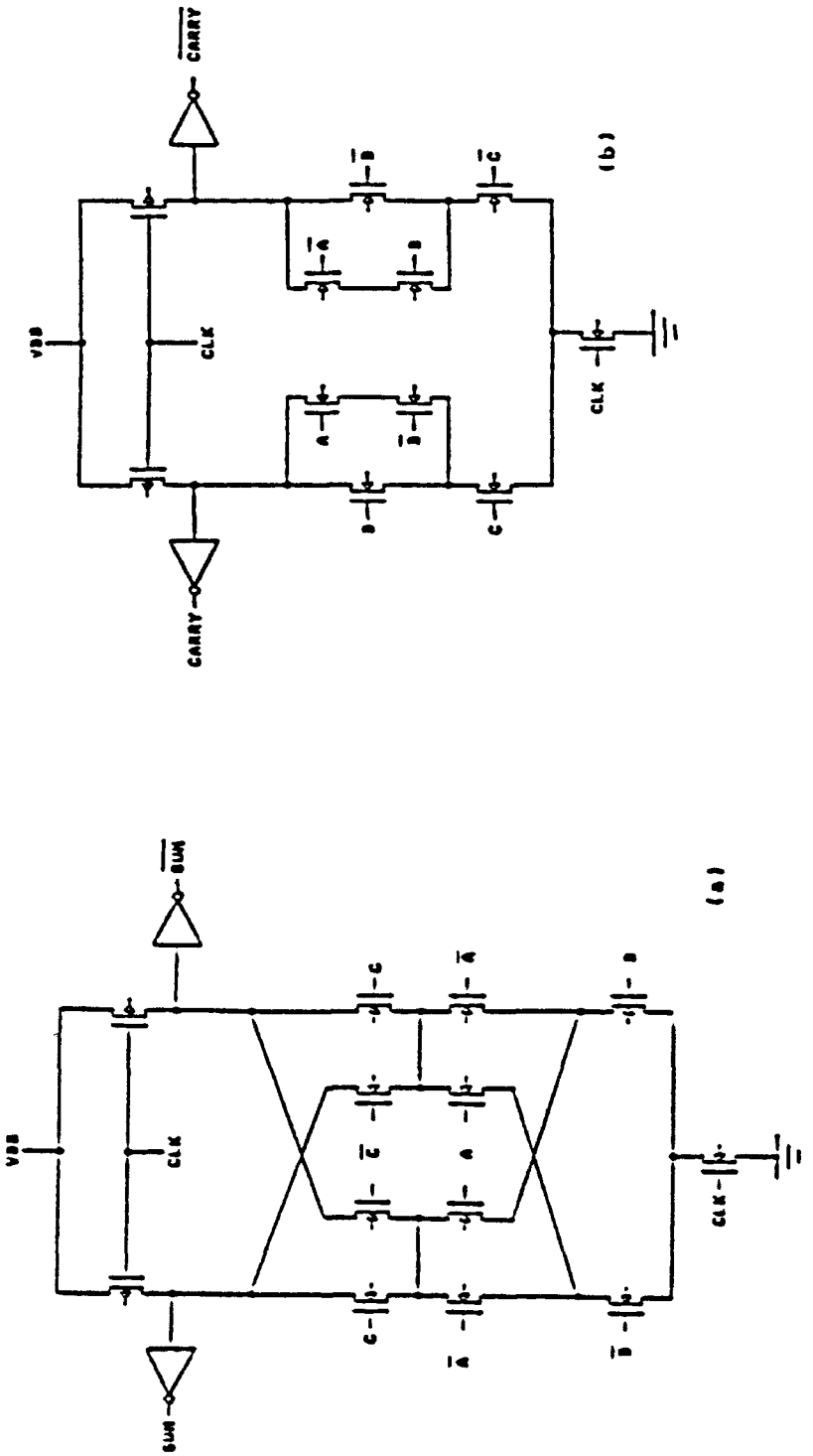


Fig. 5.7 DCVSL CMOS Full Adder

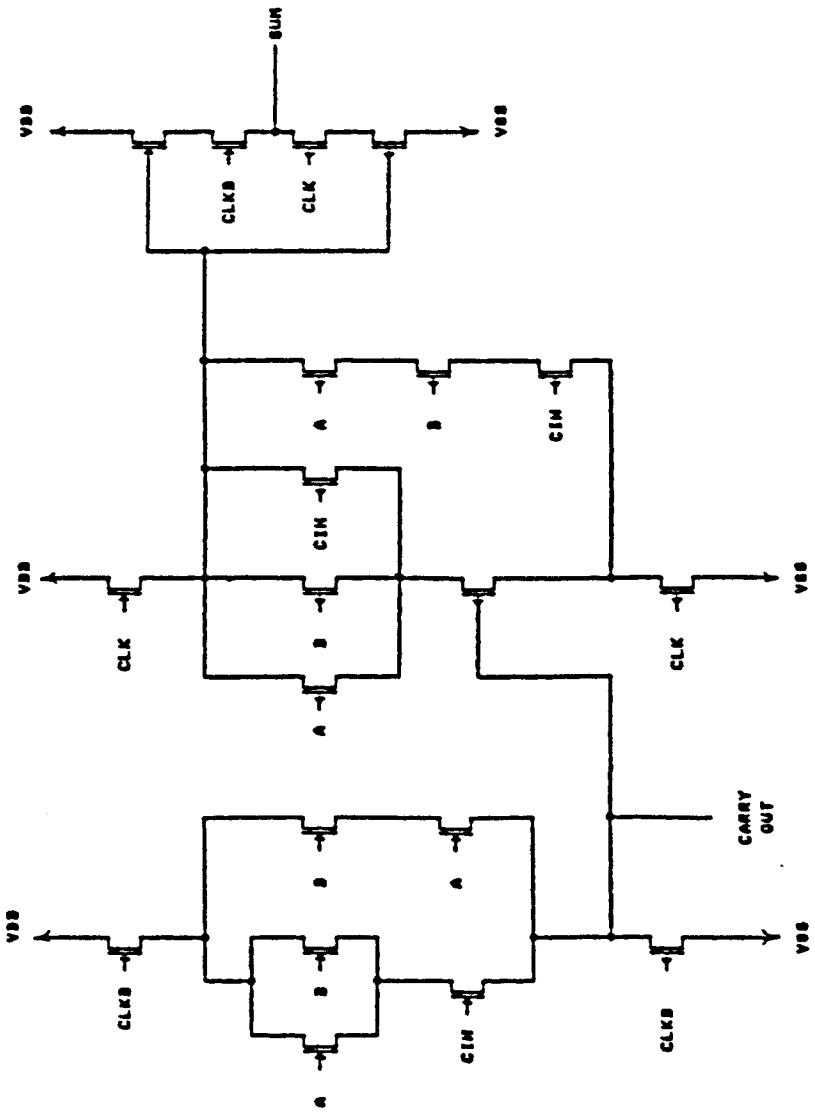


Fig. 5.8 NORA CMOS Full Adder

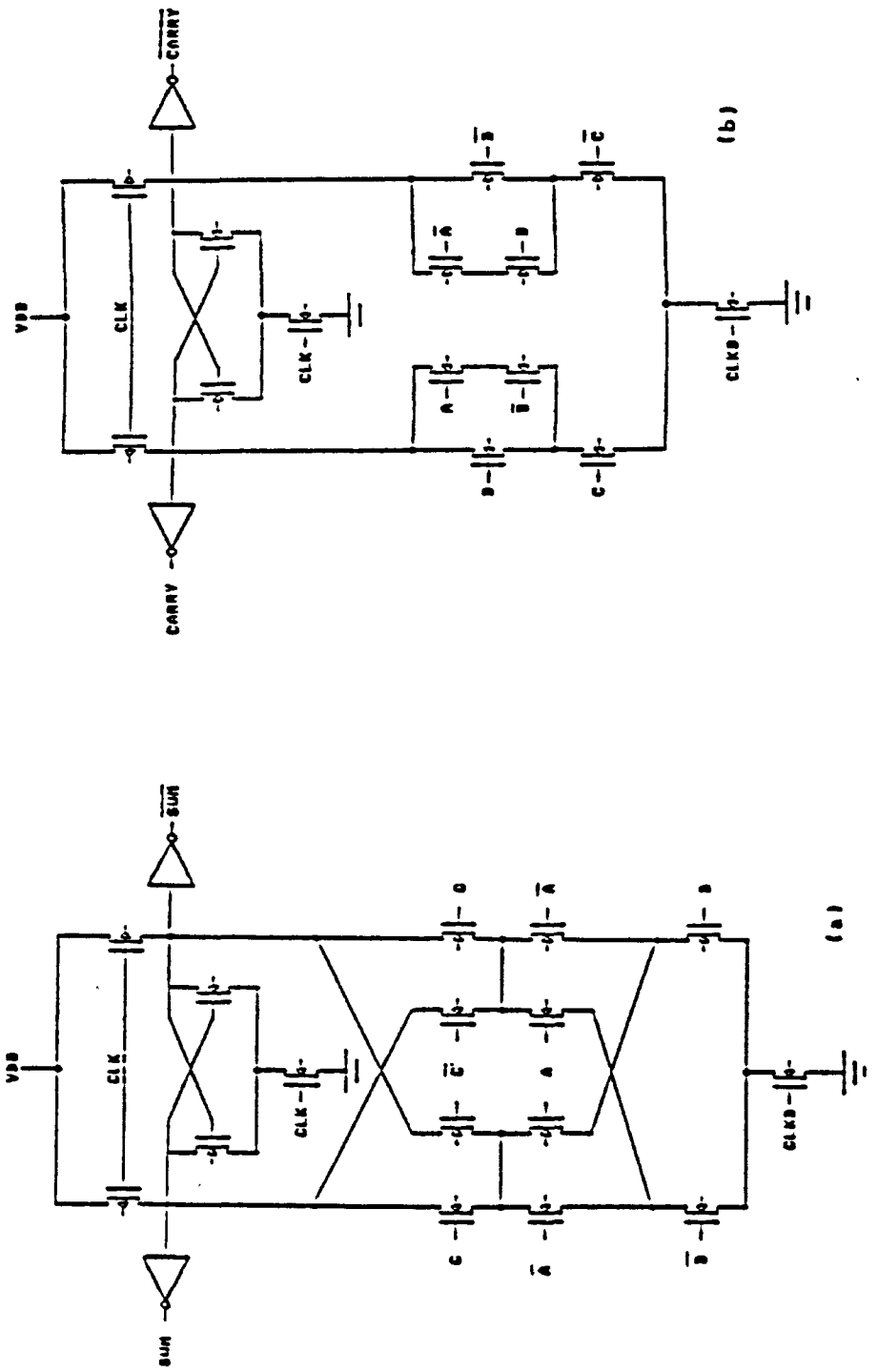


Fig. 5.9 SSDL CMOS Full Adder

5.4 OPTIMIZATION OF TRANSMISSION GATE FULL ADDE FOR STANDARD CELL LIBRARY

The optimization of any VLSI circuit uses some subset of the following categories in the cost function:

I - Logic domain optimization

II - Circuit domain optimization

III - Mask layout level optimization

5.4.1 LOGIC DOMAIN OPTIMIZATION

Logic minimization algorithms are used to minimize the number of gates required for a specific functional behavior. In the design of the transmission gate adder this step has been implicitly considered since the mapping is directly from the functional level to the circuit level.

5.4.2 CIRCUIT DOMAIN OPTIMIZATION

Standard cells are the building blocks of very large scale random logic integrated circuits. Geometric regularity must be considered when designing a standard cell or polycell. Usually a uniform height is used for all standard cells and width varies according to the function. The height of the standard cell depends on the width of the NMOS and PMOS transistors (Wn & Wp), and the separation between the P and N device wells (Dnp).

Height = Wn + Wp + Dnp

To minimize the parasitic capacitances, the horizontal distances between the polysilicon gates are kept the minimum determined by the design rules. Therefore the variable

design parameters for a standard cell are Wn and Wp [32].

For CMOS, the major performance function considerations are chip area and delay time because the power dissipation is small. In this section an analytical expression for the delay time of TG adder is derived using a simple first-order model. This expression is applied to the delay/area optimization of a TG standard cell full adder.

5.4.2.1 DERIVATION OF THE DELAY TIME

Logic circuits exhibit delay because the voltage across the parasitic capacitors cannot change instantaneously. The parasitic capacitances in CMOS circuits are due to gate, drain, source, and runners. The time taken to charge and discharge these capacitors, determines the switching speed of a CMOS gate.

Before proceeding further, let us recall some related key words and their definitions:

Propagation delay : Time required for a switching transition to occur at the output in response to a transition in the input.

Delay time : Time interval between input transition (50 percent level) and the 50 percent output level.

Since the TG adders consist of a few inverters and transmission gates, the delay times of an inverter and a transmission gates are first discussed.

Following the derivation of the the delay time for CMOS inverter in [30], the delay is approximated by:

$$tf = 2 \frac{CL}{Bn\ (Vdd - Vtn)} \times \left[\frac{Vtn - 0.1\ Vdd}{Vdd - Vtn} + \frac{1}{2} Ln \left(\frac{19\ Vdd - 20\ Vtn}{Vdd} \right) \right]$$

$$tr = 2 \frac{CL}{Bp\ (Vdd - Vtp)} \times \left[\frac{Vtp - 0.1\ Vdd}{Vdd - Vtp} + \frac{1}{2} Ln \left(\frac{19\ Vdd - 20\ Vtp}{Vdd} \right) \right]$$

$$Td = \frac{tf + tr}{4}$$

Where Bn and Bp are the gain factors for NMOS and PMOS devices respectively. CL is the cumulative load capacitance which consists of drain capacitances in the driver (Co), routing capacitances (Cw), and total input capacitance of the fan-out gates (Ci).

$CL = Co + Cw + Ci$

Drain diffusion area has two capacitance components. One is the junction capacitance per unit area (C_{ja}), and the other is the sidewalls periphery capacitance per unit perimeter (C_{jsw}). The total drain diffusion capacitance of an NMOS transistor is

$Cdn = C_{jan}\ (Wn.Ln) + 2\ C_{jswn}\ (an + bn)$

Where an and bn are the length and width of the drain diffusion area. In the same manner, the drain diffusion capacitance of a PMOS transistor is

$Cdp = C_{jap}\ (Wp.Lp) + 2\ C_{jswp}\ (ap + bP)$

Routing capacitances, which are due to parasitic capacitances of interconnect wiring, can be approximated by

a parallel plate model,

$Cw = Cxy \cdot A$

Where Cxy is the capacitance per unit area of parallel x and y plates (e.g. metal over diffusion) and A is the area of the parallel plates.

The gate capacitances also can be modeled as parallel plate capacitors. The thin silicon oxide layer over the gate area acts as an insulator between polysilicon and substrate. The gate capacitor of an NMOS transistor is approximated by

$Cgn = Cox\ (Wn.Ln)$

Let us define the CMOS gate capacitance as follows:

$Cg = Cgn + Cgp$

Where Cgp is the PMOS gate capacitance.

When a signal is steered through a transmission gate a delay is encountered. The circuit acts like an RC network. Let us define

$$Rpass = \frac{Rp.Rn}{Rp + Rn}$$

The time constant of this circuit "Rpass.CL" gives a pretty good estimate of rise time and fall time delays. Horwitz has estimated the best rise and fall time delay through a transmission gate to be [14]

$Tf = 0.79\ Rpass.CL$

$Tr = 0.63\ Rpass.CL$

Therefore, in general the delay time of the TG full adder can be represented as :

$Td = f\ (Wn, Wp)$ 5-1

The delay time also depends on the shape of the input waveform. Input with shorter rise and fall time cause less delay. Equation 5-1 is valid for step inputs.

5.4.2.2 CALCULATION OF THE AREA

As it was pointed out earlier in this chapter, the height of the standard cell depends on the channel widths of the NMOS and PMOS transistor and the separation of P and N device wells (Dnp).

$H = Wn + Wp + Dnp$

The length of the standard cell adder is constant, because to minimize the parasitic capacitances, horizontal distances are kept at the minimum determined by the design rules. Therefore there is no real variable governing the length of the standard cells. Thus the area of the chip can be represented by

$A = Wn + Wp + Dnp$

5.4.2.3 OPTIMIZATION OF THE ADDER WITH DIFFERENT CRITERIA

The objective function for the product of area and delay time is found to be

$Fobj\ (Wn, Wp) = A.Td$

By using any multivariable optimization technique (e.g., Hook-jeeves) the optimum values of Wn and Wp can be obtained.

Optimization for a specific delay involves minimization of the following objective function:

Minimizing $Wn + Wp + Dnp$

Subject to Td = a

From the method of lagrangian multipliers the above problem can be reduced to the unconstrained minimization problem of the following cost function:

$Fobj\ (Wn, Wp) = (Wn + Wp + dnp) - K\ (Td-a)$

Optimization for a specific height means :

Minimizing Td = f(Wn, Wp)

Subject to Wn + Wp + Dnp = h

Replacing Wp = h - Dnp - Wn in the objective function, results in a single variable minimization problem

$Fobj = F\ (Wn, h-Dnp-Wn)$

An optimization example using this algorithm is given in Appendix C.

5.4.4 LAYOUT DOMAIN OPTIMIZATION

In CMOS a logic function may be implemented by means of a library of standard cells such as NAND or NOR gates or using NMOS AND PMOS transistors. The latter is called custom integrated circuit design. Custom ICs have the advantage of smaller size and better delay characteristics. But a standard cell based design allows a significant reduction in design effort. Also, when logic design involves the repetition of a cell, (such as pipelined adder or multiplier) the standard cell approach is better. The custom integrated circuit approach is selected for the design of the TG full adder to be used in the library of standard cells.

The most common scheme used in the layout design of CMOS gates is "line of diffusion". It consist of a single row of PMOS transistors and a row of NMOS transistors. The N and P devices are aligned vertically. Since usually in static CMOS the NMOS and PMOS blocks are duals of each other, therefore, the number of transistors in both rows are equal.

The optimal layout is obtained by following the algorithm given in [33]. The CMOS circuit is converted to a graph as follows:

I - Every transistor is represented by an edge.

II - Every source/drain connection is represented by a vertex.

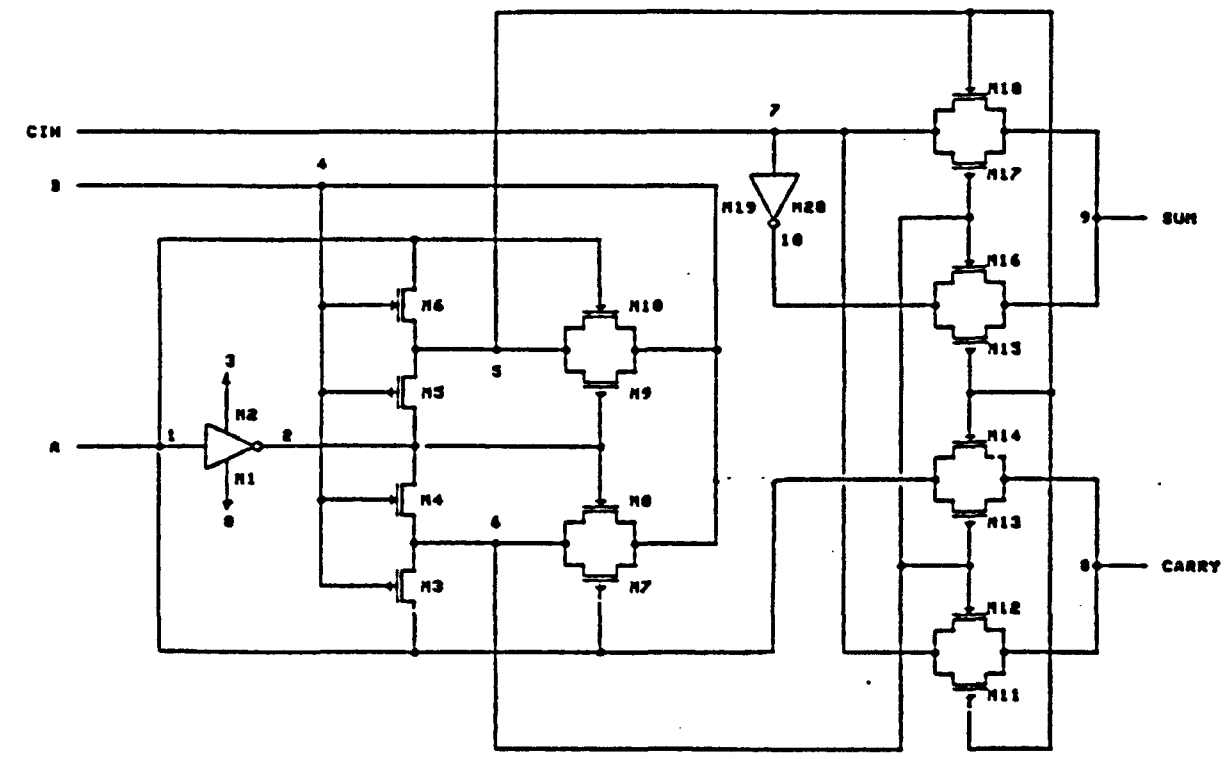
The adjacent edges in the graph model may share a common source-drain connection. If it is possible to find an EULER PATH (a sequence of the edges containing all the edges in the n-graph and p-graph that have identical labeling), then the gate may be designed with no breaks in the device wells. If such a path does not exist then the graph is broken into a minimum number of subgraph each one with an Euler path.

A single Euler path could not be found for either of the TG full adders, however drawing the graph model of the circuit helped to improve the layout design. For example the n-graph model of the TGFA#2 is shown in Fig. 5.10. Even though a single Euler path does not exist for this circuit, the optimal layout can be obtained by the following sequence of edges(single pseudo Euler path):

n-graph M1-M5-M9-M7-M3-M13-M11-M17-M15-M19

p-graph M2-M4-M8-M10-M6-M14-M12-M18-M16-M20

This way no break has to be made in the diffusion areas. Appendix B contains the layout plot of this adder.



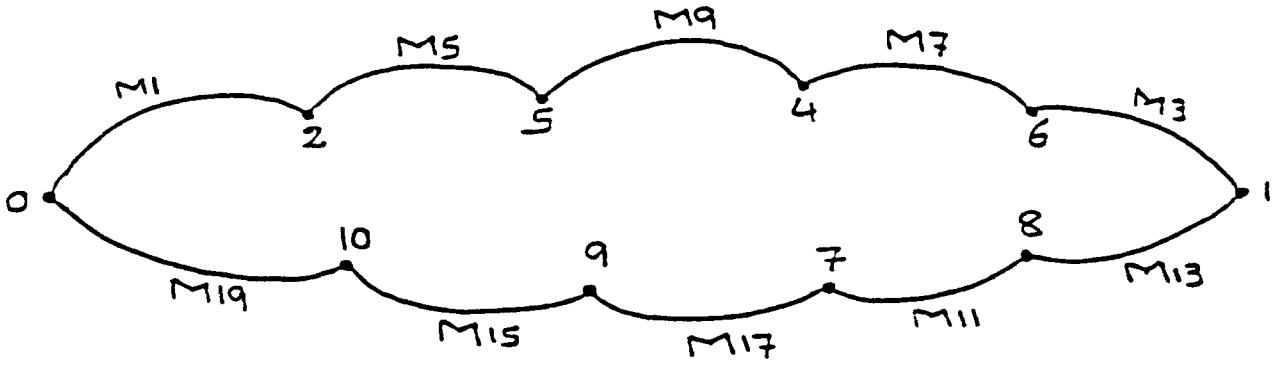


Fig. 5.10 Graph Model of TGFA#2

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 SUMMARY

In this work, the development of very high-speed VLSI adders for digital signal processing applications has been investigated.

Design methodologies associated with various techniques for fast two-operand binary adders, and parallel multipliers have been studied. Based on the speed evaluation of different algorithms available for binary addition, pipelined ripple-carry adders have been selected as the best addition technique. A 10-bit pipelined ripple-carry binary adder has been implemented using 3um CMOS technology.

Residue Number System (RNS) arithmetic has been described and four different approaches for VLSI implementation of RNS adders have been discussed. A comparative analysis shows that the highest throughput rate for RNS addition is obtained by using pipelined ripple-carry-based RNS adders. A 5-bit pipelined RNS adder has been implemented using 3um CMOS technology.

Various CMOS logic families have been presented with a major emphasis placed on new dynamic circuit techniques. Sample-set differential logic, which is an improved dynamic CMOS logic style, was found to be the fastest CMOS gate for high fan-in circuits. Through a first-order model, an expression for the delay time of SSDL gates has been derived. Other

issues in CMOS VLSI design such as charge sharing, noise margin, graph models, and testability have also been addressed. A dynamic CMOS full adder test chip containing DCVSL and SSDL full adders has been implemented using 3um CMOS technology. This test chip will be used to compare the performance of DCVSL and SSDL full adders.

A one-bit full adder, which can be used as the basic building block for realizing high-speed arithmetic operations, has been optimized for standard cell applications.

6.2 CONCLUSIONS

The main conclusions based on this work are explained in the following:

I - Superiority of RNS adder over Binary adder

Pipelined adders have been found to offer the highest throughput rate. For operations requiring a wide-dynamic range (more than 10-bit), pipelined ripple-carry RNS adders have advantages over their binary counterparts. These advantages show up in throughput rate, latency time, hardware complexity, and testability.

II - Suitability of SSDL in pipelined structure

Sample-set differential logic has been found to be a very suitable circuit technique for pipelined architectures because of its built-in flip-flop. This dynamic logic circuit was simplified by eliminating the ground switch to be used in the pipelined structures.

6.3 FUTURE WORK

The results of this thesis have shown that pipelined RNS arithmetic is very useful for designing high-speed special-purpose digital signal processors. There are some suggestions in this area for further research and study.

I - Multi-operand operations in pipelined RNS arithmetic

In binary pipelined structures, multioperand operations are possible on skewed data; this is not possible for existing RNS algorithms. In RNS, multi-operand operations are decomposed into several two-operand operations with nonskewed output data. This way the hardware investment is not efficient due to the requirement for deskewing latches.

II- Optimum set of moduli for a specific dynamic range

For a specific dynamic range of operation, usually there are several sets of moduli which can be used for the Residue system. There should be an optimum set of moduli for any dynamic range which results in the most efficient Residue Number System implementation. This would be a useful area for further study.

Appendix A

The SPICE 2G analog simulation program has been utilized for logic verification and delay time measurement of the circuits discussed in this thesis. The Northern Telecom CMOS3 process parameters were used for the SPICE models. In this appendix two samples of SPICE decks corresponding to a static and a dynamic full adder are presented.

(1) Static full adder simulation

Fig. A-1 shows the transmission gate full adder TGFA#2. The SPICE deck associated with this circuit is shown on the next page and the resulting waveforms are illustrated in Fig.A-2.

(2) Dynamic full adder simulation

The SUM section of the SSDL CMOS full adder is shown in Fig. A-3. Fig. A-4 shows the associated Waveforms obtained from SPICE simulations. The SPICE deck used to generate these waveforms is also listed.

```
TG 1-BIT BINARY FULL ADDER
*-------------------------------------------
*              DESIGN NO. TGFA#2
*  Transmission gate 1 bit binary full adder
*               20 transistors
*-------------------------------------------
VDD 3 0 DC 5
.OPTIONS ITL5=10000 LIMPTS=500 NOMOD
M1 2 1 0 0 MOD1 L=3U W=3U AD=20P PD=16U
M2 2 1 3 3 MOD2 L=3U W=6U AD=27P PD=15U
M3 6 4 1 0 MOD1 L=3U W=3U AD=23P PD=18U AS=27P PS=31U
M4 6 4 2 3 MOD2 L=3U W=6U AD=27P PD=15U AS=27P PS=15U
M5 5 4 2 0 MOD1 L=3U W=3U AD=20P PD=16U AS=20P PS=16U
M6 5 4 1 3 MOD2 L=3U W=6U AD=27P PD=15U AS=40P PS=20U
M7 4 1 6 0 MOD1 L=3U W=3U AD=23P PD=18U AS=23P PS=18U
M8 4 2 6 3 MOD2 L=3U W=6U AD=35P PD=18U AS=27P PS=15U
M9 4 2 5 0 MOD1 L=3U W=3U AD=23P PD=18U AS=20P PS=16U
M10 4 1 5 3 MOD2 L=3U W=6U AD=35P PD=18U AS=27P PS=15U
M11 8 5 7 0 MOD1 L=3U W=3U AD=33P PD=23U AS=33P PS=23U
M12 8 6 7 3 MOD2 L=3U W=6U AD=53P PD=24U AS=53P PS=24U
M13 8 6 1 0 MOD1 L=3U W=3U AD=33P PD=23U AS=27P PS=31U
M14 8 5 1 3 MOD2 L=3U W=6U AD=53P PD=24U AS=40P PS=20U
M15 9 5 10 0 MOD1 L=3U W=3U AD=33P PD=23U AS=20P PS=16U
M16 9 6 10 3 MOD2 L=3U W=6U AD=53P PD=24U AS=27P PS=15U
M17 9 6 7 0 MOD1 L=3U W=3U AD=33P PD=23U AS=33P PS=23U
M18 9 5 7 3 MOD2 L=3U W=6U AD=53P PD=24U AS=53P PS=24U
M19 10 7 0 0 MOD1 L=3U W=3U AD=20P PD=16U
M20 10 7 3 3 MOD2 L=3U W=6U AD=27P PD=15U
VA 1 0 PULSE(0 5 0 .5NS .5NS 100NS 200NS)
VB 4 0 PULSE(5 0 0 .5NS .5NS 100NS 200NS)
VC 7 0 PULSE(0 5 0 .5NS .5NS 50NS 100NS)
*------ C1 & C2 load capacitance---------
C1 1 0 38F
C2 2 0 37F
C4 4 0 98F
C5 5 0 129F
C6 6 0 75F
C7 7 0 57F
C8 8 0 .1P
C9 9 0 .1P
C10 10 0 12F
*-----------N & Pmos models-----------------------
.MODEL MOD1 NMOS LEVEL=3 KP=50.0E-6 VTO=0.7 TOX=5E-8 GAMMA=1.1
+PHI=0.6 LAMBDA=0.01 RD=40 RS=40 PB=0.7 CGSO=3E-10 CGBO=5E-10
+CGDO=3E-10 RSH=25 CJ=44E-5 MJ=0.5 CJSW=4E-10 MJSW=0.3 JS=1E-5
+NSUB=1.7E16 XJ=6E-7 LD=3.5E-7 UO=775 VMAX=1E5 THETA=0.11
+ETA=0.05 KAPPA=1
.MODEL MOD2 PMOS LEVEL=3 KP=16E-6 VTO=-0.8 TOX=5E-8 GAMMA=0.6
+PHI=0.6 LAMBDA=0.03 RD=100 RS=100 PB=0.6 CGSO=2.5E-10 CGBO=5E-10
+CGDO=2.5E-10 RSH=80 CJ=15E-5 MJ=0.6 CJSW=4E-10 MJSW=0.6 JS=1E-5
+NSUB=5E15 XJ=5E-7 LD=2.5E-7 UO=250 VMAX=0.7E5 THETA=0.13
+ETA=0.3 KAPPA=1
*-------------------------------------------
```

```
*    nodes :  (1)addend , (4)augend , (7)carry in
*             (9) sum ,   (8) carry out
*-----------------------------------------------
*
.TRAN .25NS 60NS
 PRINT TRAN V(7) V(9) V(8)
 PLOT TRAN  V(7) V(9) V(8)
 END
```

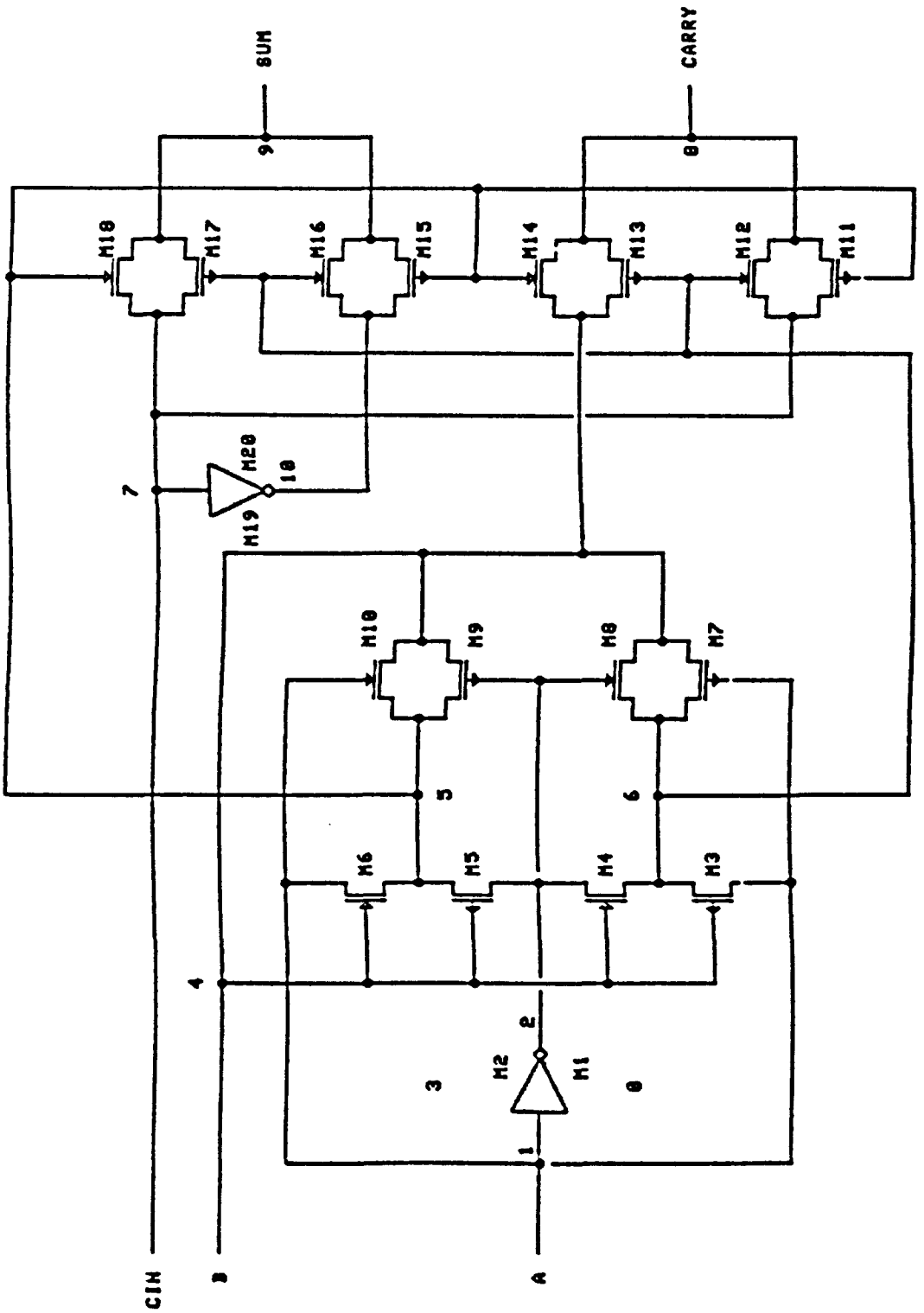


Fig. A-1 Schematic circuit of TGFA#2

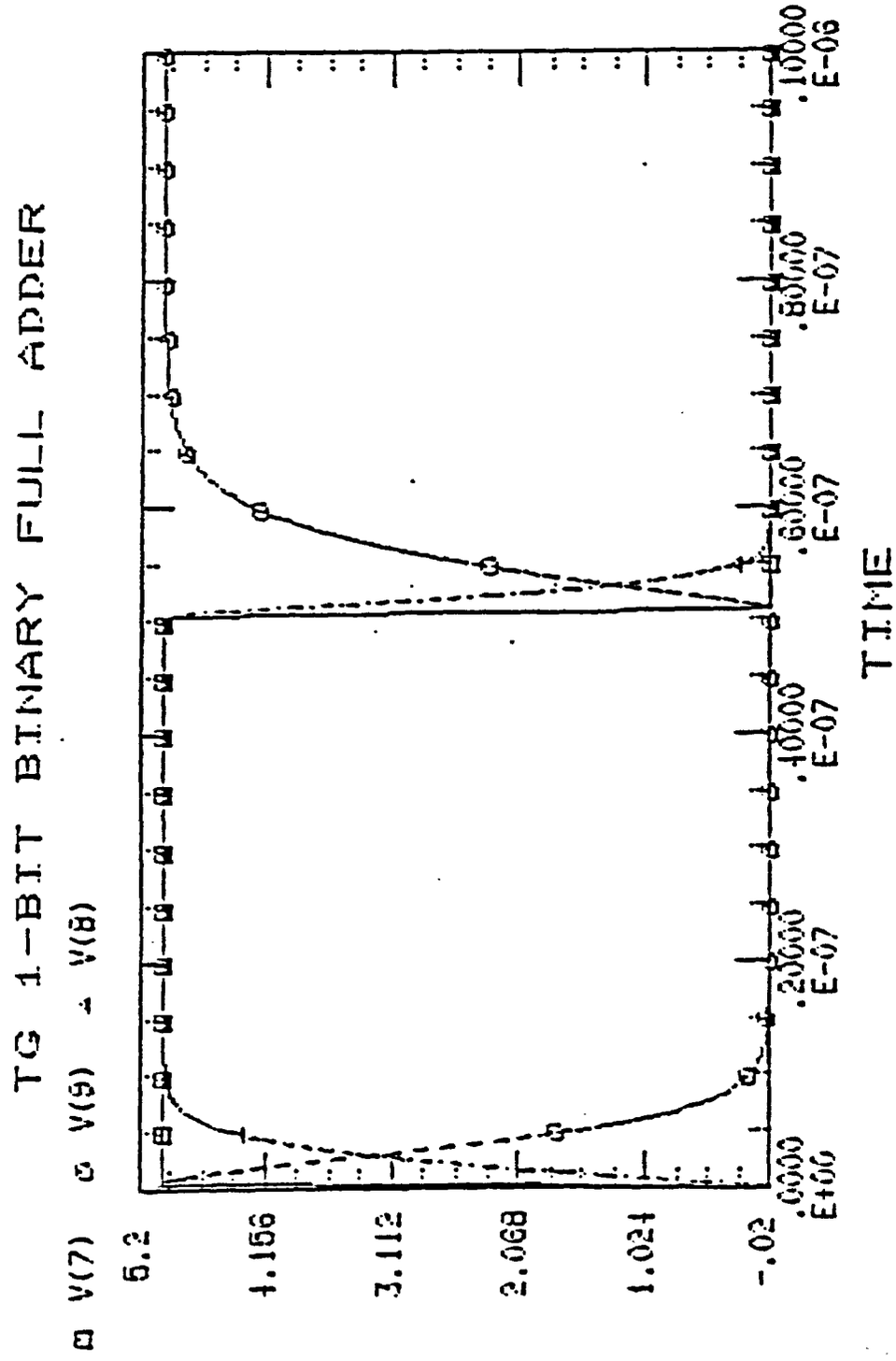


Fig. A-2 Waveforms Associated with the Fig. A-1, obtained from SPICE

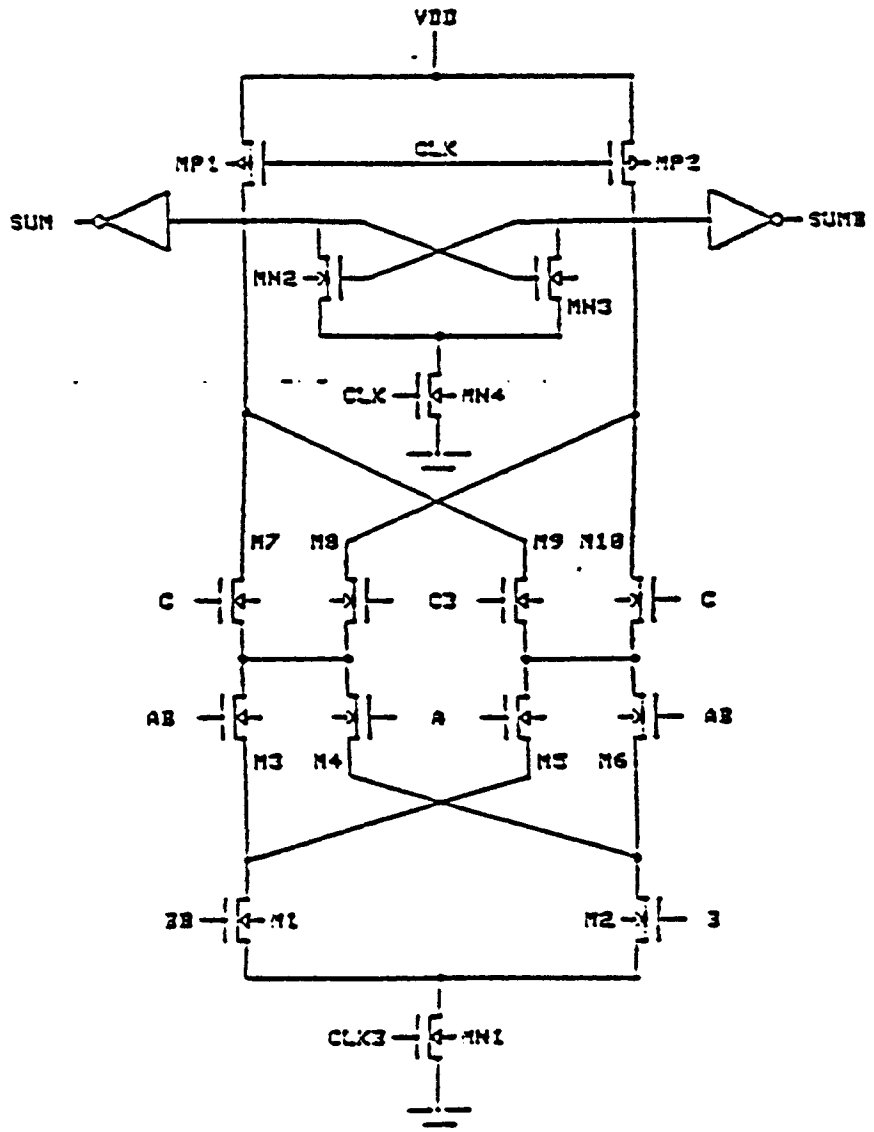


Fig. A-3 The Sum section of the SSDL CMOS Full Adder

```
SSDL 1 BIT BINARY ADDER
*-------------------------------------------------------
*                  DESIGN NO. SSDL
*  Sample and set differential logic gate
*  We consider the worst case for swiching,i.e. when
*  nodes 8 and 9 be at logic 1 & 0 .
*  TG is used to connect inputs at proper times.
*-------------------------------------------------------
VDD 12 0 DC 5
.OPTIONS ITL5=10000 LIMPTS=500 NOMOD DEFL=3U DEFW=3
.IC V(8)=5 V(9)=0
VCLK 10 0 PULSE(5 0 0 0 0 5NS 10NS)
VCLKB 11 0 PULSE(0 5 0 0 0 5NS 10NS)
M1 1 11 0 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M2 4 2 1 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M3 5 3 1 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M4 6 2 4 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M5 6 3 5 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M6 7 3 4 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M7 7 2 5 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M8 8 3 6 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M9 9 2 6 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M10 8 2 7 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M11 9 3 7 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M12 8 10 12 12 MOD2 W=9U AD=23P AS=23P PD=22U PS=22U
M13 9 10 12 12 MOD2 W=9U AD=23P AS=23P PD=22U PS=22U
M14 8 9 15 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M15 9 8 15 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M16 15 10 0 0 MOD1 AD=22P AS=22P PD=16U PS=16U
.SUBCKT INVERTER 1 2 3
*       nodes:(1)input ,(2)output ,(3)vdd
M21 2 1 0 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M22 2 1 3 3 MOD2 W=9U AD=23P AS=23P PD=22U PS=22U
.ENDS INVERTER
.SUBCKT TG 1 2 3 4 5
*    nodes    1:ctrl  2:input  3:ctrlb  4:output   5:vdd
M23 4 1 2 0 MOD1 AD=22P AS=22P PD=16U PS=16U
M24 2 3 4 5 MOD2 W=9U AD=23P AS=23P PD=22U PS=22U
.ENDS TG
X1 8 14 12 INVERTER
X2 9 13 12 INVERTER
X3 11 0 10 2 12 TG
X4 11 12 10 3 12 TG
*------ C1 & C2 load capacitance---------
C1 13 0 .05P
C2 14 0 .05P
*-----------N & Pmos models----------------------
.MODEL MOD1 NMOS LEVEL=3 KP=50.0E-6 VTO=0.7 TOX=5E-8 GAMMA=1.1
+PHI=0.6 LAMBDA=0.01 RD=40 RS=40 PB=0.7 CGSO=3E-10 CGBO=5E-10
+CGDO=3E-10 RSH=25 CJ=44E-5 MJ=0.5 CJSW=4E-10 MJSW=0.3 JS=1E-5
+NSUB=1.7E16 XJ=6E-7 LD=3.5E-7 UO=775 VMAX=1E5 THETA=0.11
+ETA=0.05 KAPPA=1
```

```
.MODEL MOD2 PMOS LEVEL=3 KP=16E-6 VTO=-0.8 TOX=5E-8 GAMMA=0.6
+PHI=0.6 LAMBDA=0.03 RD=100 RS=100 PB=0.6 CGSO=2.5E-10 CGBO=5E-10
+CGDO=2.5E-10 RSH=80 CJ=15E-5 MJ=0.6 CJSW=4E-10 MJSW=0.6 JS=1E-5
+NSUB=5E15 XJ=5E-7 LD=2.5E-7 UO=250 VMAX=0.7E5 THETA=0.13
+ETA=0.3 KAPPA=1
*-------------------------------------------------
*   nodes :      (14) sum ,  (13) sum Complement
*-------------------------------------------------
.TRAN .04NS 12NS
 PRINT TRAN V(10) V(13) V(14)
 PLOT TRAN V(10) V(13) V(14)
 END
```

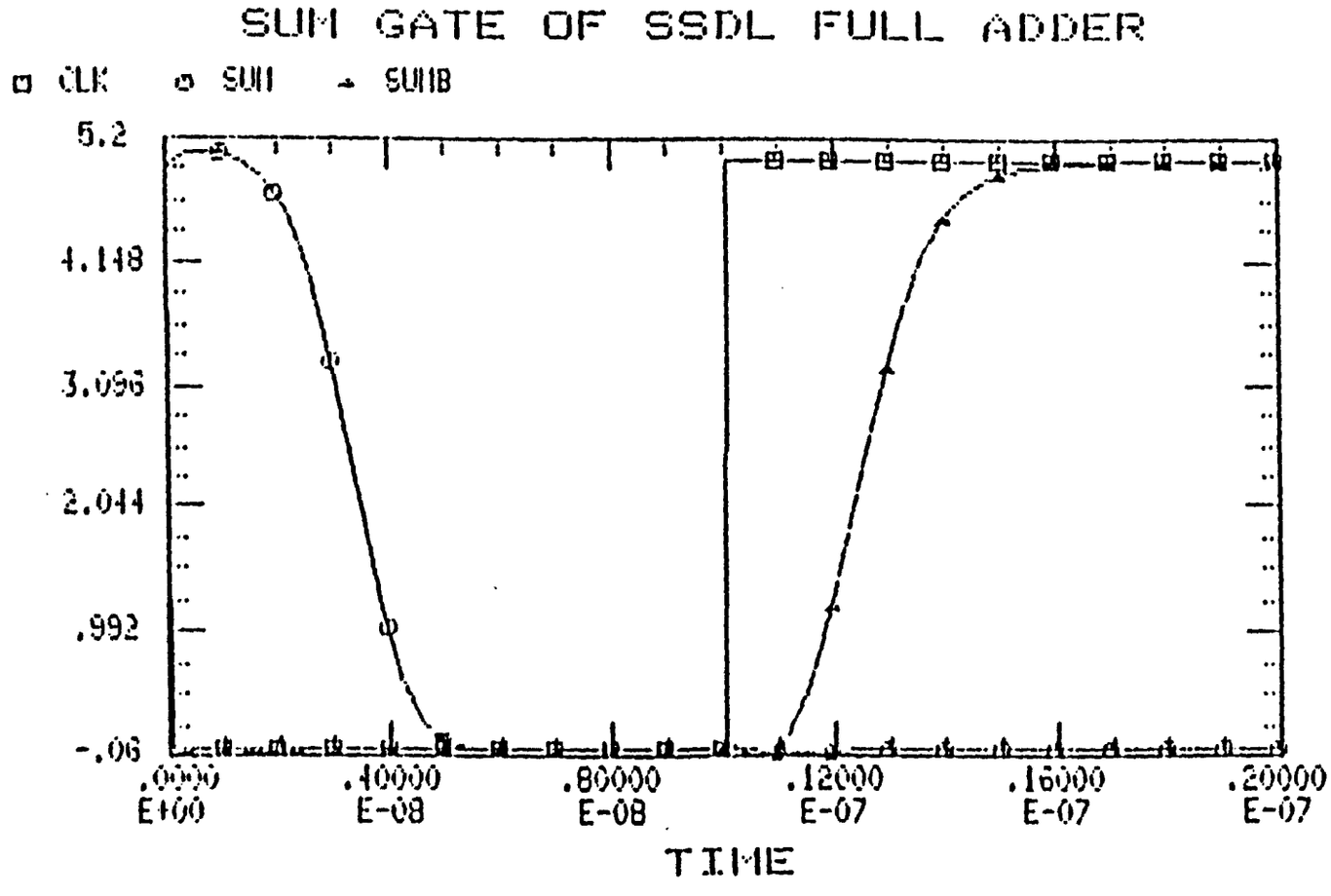


Fig. A-4 Waveforms associated with the Fig. A-3, obtained from SPICE

APPENDIX B

The DAISY Chipmaster workstation was utilized to lay out the CMOS circuits. In this section some of the layouts created by the DAISY system will be presented.

The layout in Fig. B-1 is a realization of the half adder of Fig. 4.5.

The full adder layout in Fig. B-2 is a realization based on the schematic of Fig. 5.3.

Fig. B-3 shows the layout of a dynamic register in CMOS using transmission gate. The schematic layout of this register is shown in Fig. 4.6.

The following three chips have been implemented in 3um CMOS technology and have been fabricated by Northern Telecom.

(1) ICWRDFA

This test chip contains two dynamic CMOS full adders. Differential Cascode Voltage Switch Logic (DCVSL) and Sample-Set Differential Logic (SSDL) circuit techniques are used to implement these full adders. The schematic circuits of the adders are shown in Fig. B-4 and B-5; both designs have the same input and output buffers. This test chip will be used to compare the performance of the DCVSL and SSDL full adders.

(2) IC3WRPRA

This chip performs modulo 29 addition on an input steam of data. High-Speed addition techniques (Pipelined ripple-carry

adder of Fig. 3.5) are utilized to achieve high throughput rate performance. The transmission gate full adder of Fig. B-2 and the pseudo two-phase dynamic TG register of Fig. B-3 are used in the realization of this adder. The throughput rate is limited by the output pad buffers which are designed to drive a 5PF load.

This design can easily be modified to perform modulo M addition, where M is any 5-bit number.

(3) IC3WRBPA

This chip contains a 10-bit binary adder. The design is based on the pipelined ripple-carry adder of Fig. 2.15. The full adder cells are implemented with a TG full adder (Fig. B-2) and the registers are pseudo two-phase TG flip-flops (Fig. B-3). The throughput rate of this adder is determined by the delay time of a full adder plus a latch (about 100MHZ), but such a high-speed performance cannot be achieved because of the limited I/O pad capabilities.

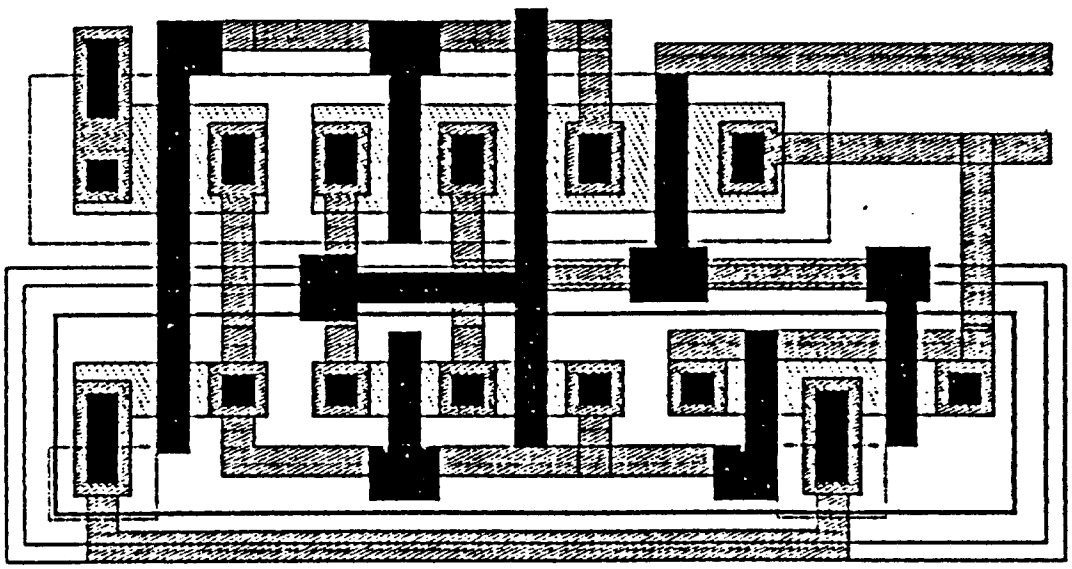


Fig. B-1 A TG Half Adder Layout

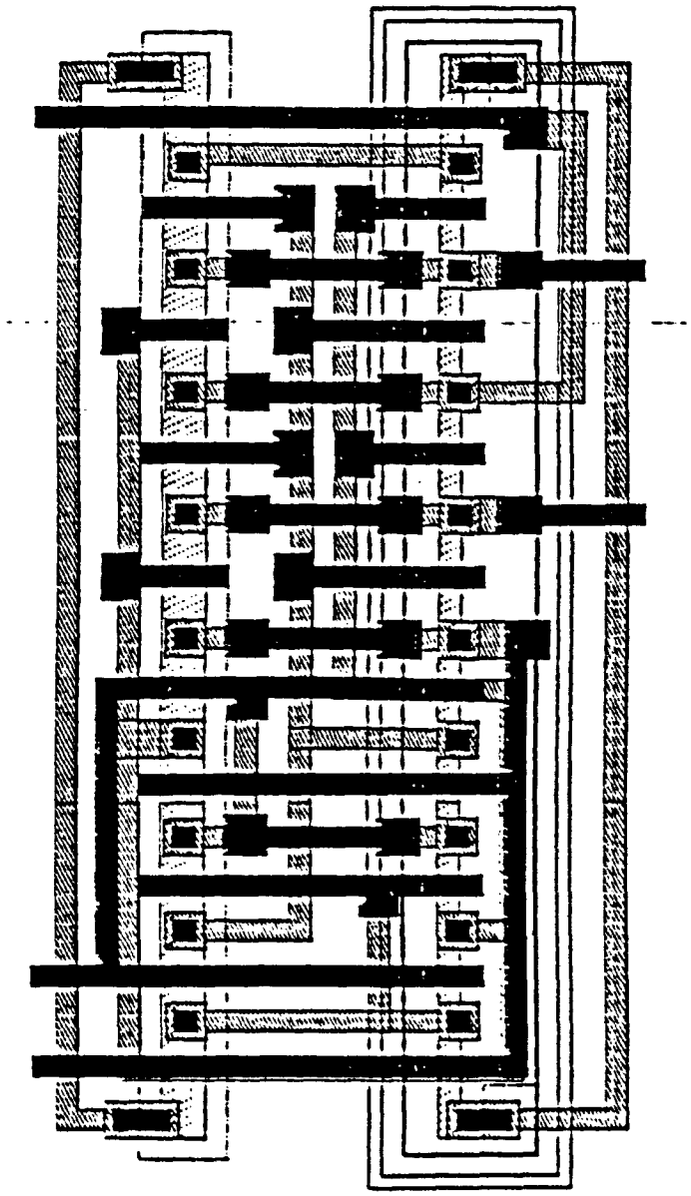


Fig. B-2 A TG Full Adder Layout

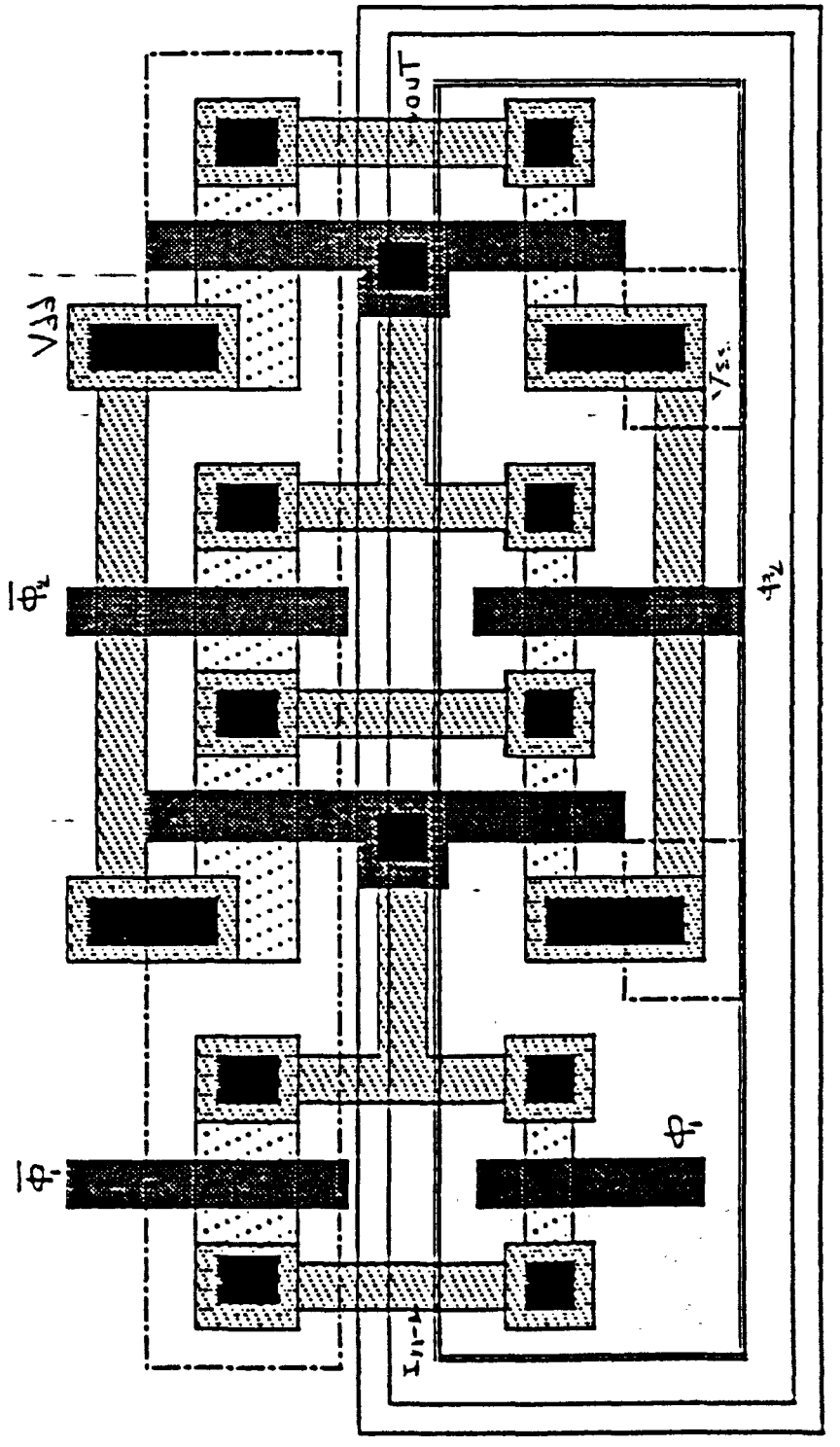


Fig. B-3 A Pseudo Two-phase Dynamic Register Layout

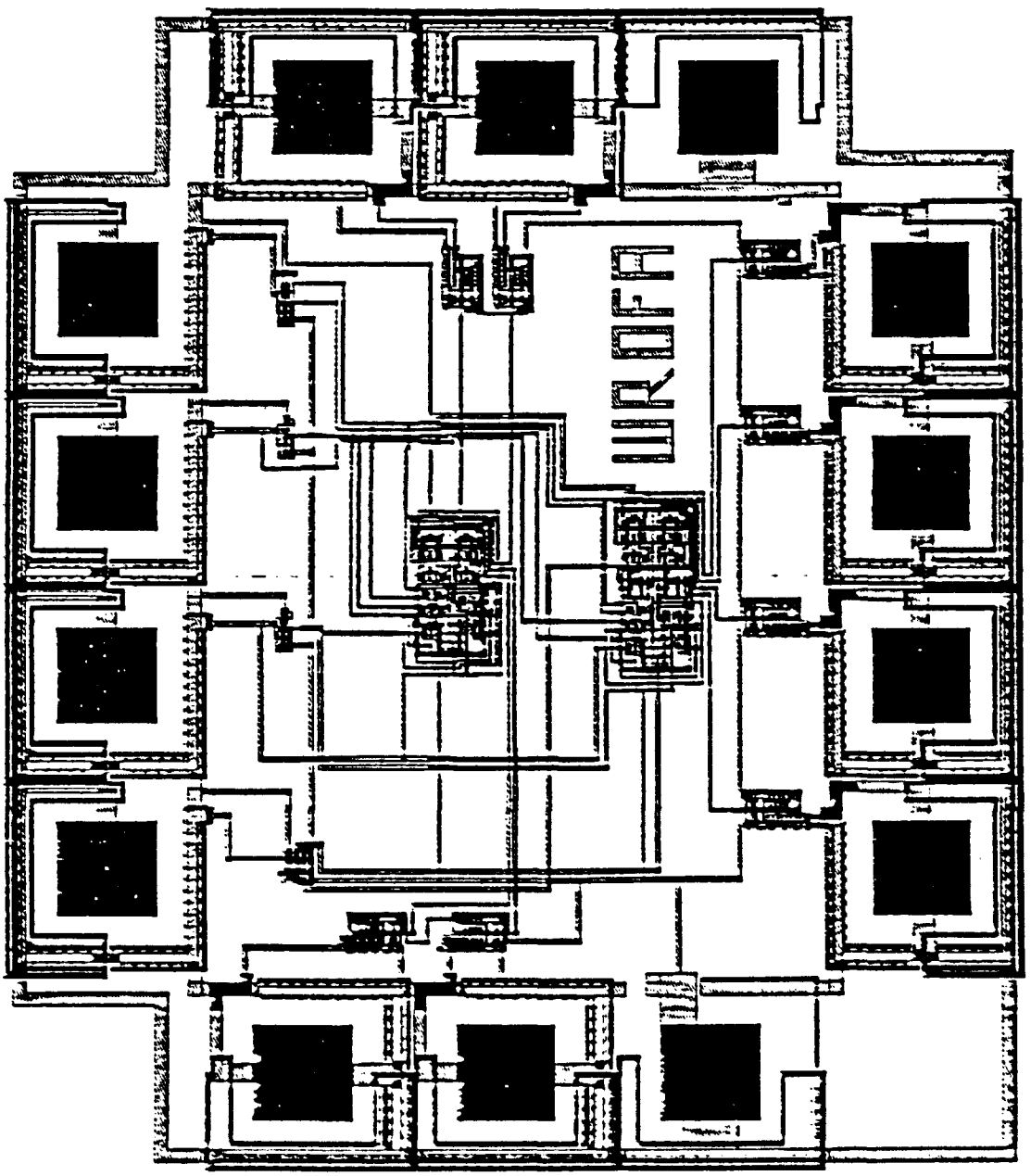


Fig. B-4 IC3WRDFA Chip Layout

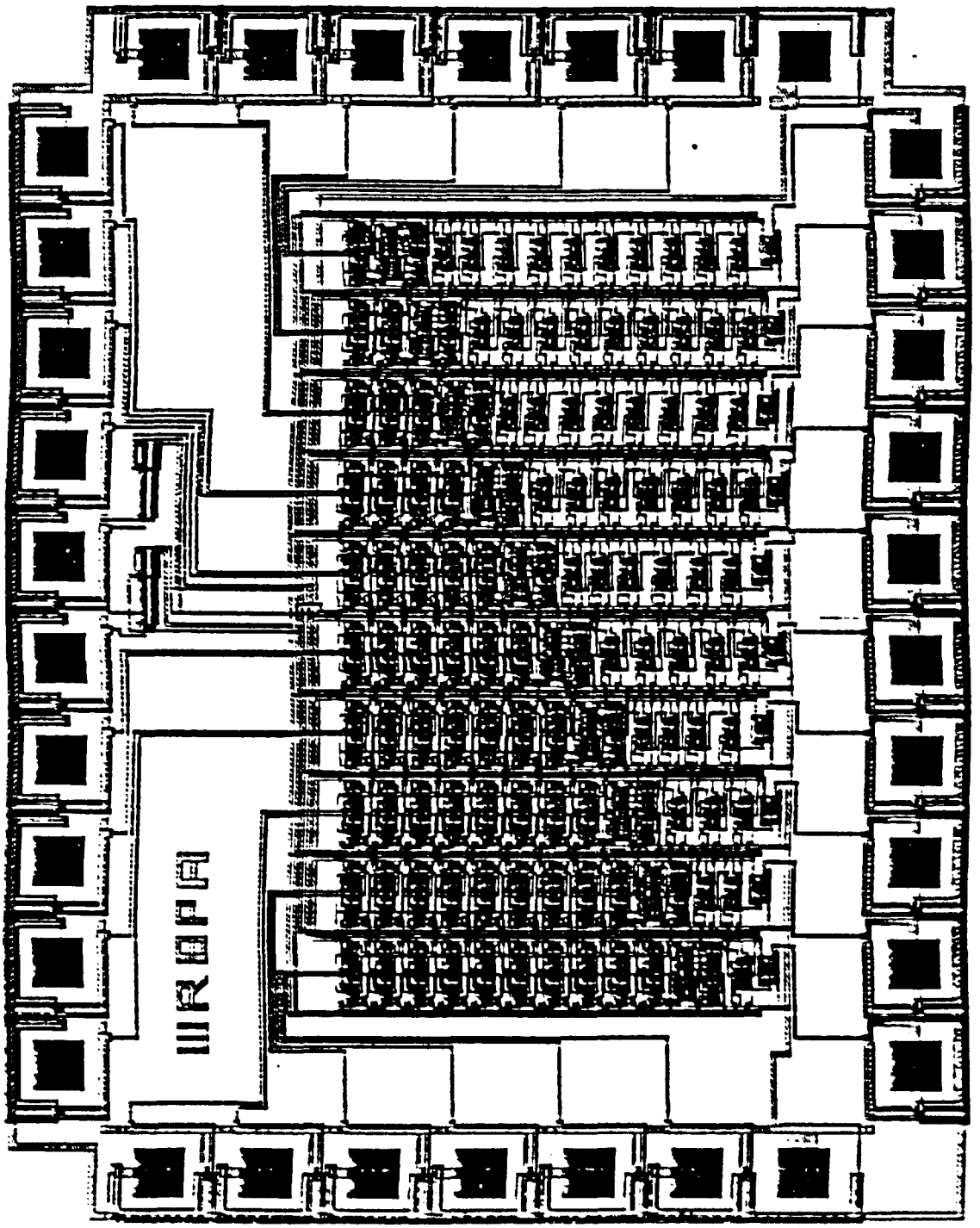


Fig. B-5 IC3WRBPA Chip Layout

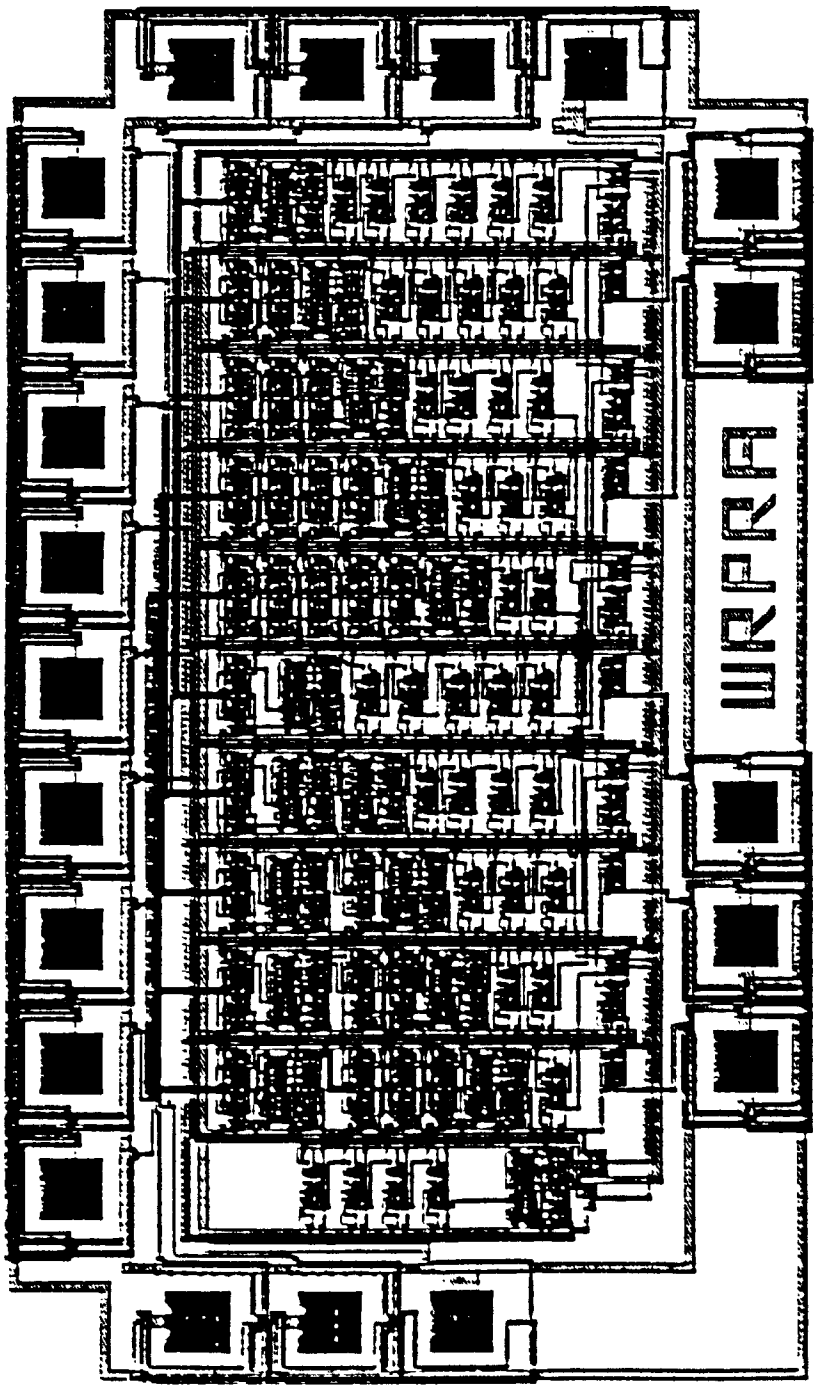
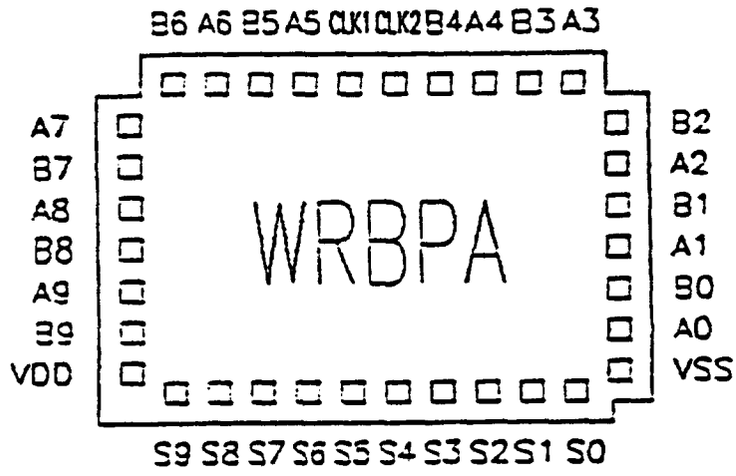
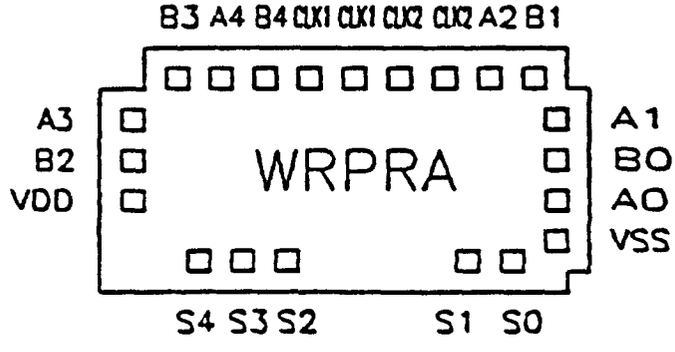


Fig. B-6 IC3WRPRA Chip Layout





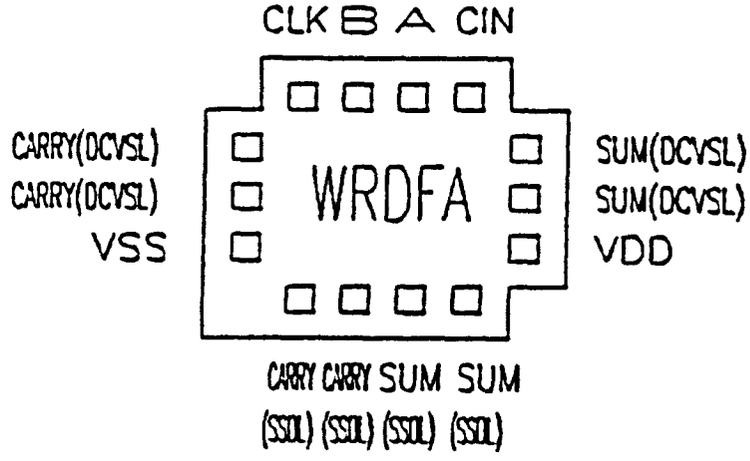


Fig. B-7 Pinouts of ICJWRBPA, ICJWRPRA and ICJWRDFA

APPENDIX C

In this appendix the analytical model of the TG full adder, derived in section 5.4, is applied to the optimization of TGFA#2 (Fig. 5.2) with three different criteria.

(1) Minimization of delay-area product:

The Hook-Jeeves multivariable optimization technique was implemented for the minimization of the performance function, Eq. 5-1. A FORTRAN program was developed for this optimization problem using the Hook-Jeeves algorithm (PROGRAM1). The optimum values of Wn and Wp were found to be Wn = 6 um and Wp = 12 um. The corresponding delay is Td = 5.5 ns.

(2) Optimization for a specific delay

One easy way of solving this problem is to keep Wn constant and to change Wp until the specified gate delay is achieved. This process is repeated for several values of Wn in the proper search region.

The optimum values of Wn and Wp for the gate delay of 5ns are Wn = 8 um and Wp = 18 um.

(3) Optimization for a specific height

This is a single variable optimization. The results for H=50 and Dnp = 30um are Wn = 6 um and Wp = 14um.

```
C*************************************************************
C*       OPTIMIZATION OF AN 1-BIT FULL ADDER                 *
C*                                                           *
C*      IN THIS PROGRAM THE METHOD OF "HOOK-JEEVES" IS       *
C*    RESORTED IN ORDER TO OPTIMIZE THE MULTI-VARIABLES      *
C*    FUNCTION, (DELAY-AREA PRODUCT)                         *
C*              ------------------------                     *
C*      PARAMETERS USED:                                     *
C*                                                           *
C*    VTON   : ZERO BIAS THRSHOLD VOLTAGE FOR N-CHANNEL      *
C*    VTOP   : ZWRO BIAS THRESHOLD VLOTAGE FOR P-CHANNEL     *
C*    UN     : MOBILITY OF ELECTRON                          *
C*    UP     : MOBILITY OF HOLES                             *
C*    COX    : GATE OXIDE CAPACITANCE PER UNIT AREA          *
C*    WN     : NMOS TRANSISTOR CHANNEL WIDTH; XB(1)          *
C*    WP     : PMOS TRANSISTOR CHANNEL WIDTH; XB(2)          *
C*                                                           *
C*************************************************************
C
      REAL XB(2),XR(2),XBP(2)
      INTEGER FLAG(2)
      N=2
      READ,(XB(I),I=1,N)
      PRINT 99,XB(1),XB(2)
  99  FORMAT(/,5X,'INITIAL VALUES',/,5X,
     1  'X1= ',F9.4,5X,'X2=',F9.4,/)
      DO 3 I=1,N
      XBP(I)=0.
  3   XR(I)=XB(I)
      READ,DEL,ALPHA,EPS
C
C*************************************************************
C*    EXPLORATORY SEARCH DO LOOP.                            *
C*************************************************************
C
 100  DO 10 I=1,N
      FB=F(XB)
      XP=XB(I)+DEL
      XB(I)=XP
      FP=F(XB)
      IF(FP.LT.FB)THEN
          XBP(I)=XR(I)
          XR(I)=XB(I)
          FLAG(I)=1
        ELSE DO
          XM=XR(I)-DEL
          XB(I)=XM
          FM=F(XB)
         IF(FM.LT.FB)THEN
            XBP(I)=XR(I)
            XR(I)=XB(I)
            FLAG(I)=1
          ELSE DO
            XB(I)=XR(I)
            FLAG(I)=0
```

```
            END IF
         END IF
   10    CONTINUE
         DO 20 I=1,N
C
C*************************************************************
C*  CHECK TC SEE IF THE EXPLORATCRY SEARCH WAS                *
C*  SUCCESSFUL(FLAG(I)=1) OR NOT.                             *
C*************************************************************
C
         IF(FLAG(I).EQ.1)GC TO 30
   20    CCNTINUE
C
C*************************************************************
C*  CHECK TC SEE IF DELTA IS LESS THAN EPSILON.              *
C*************************************************************
C
   25    IF(DEL.LE.EPS)GO TO 50
         DEL=DEL/ALPHA
         GC TO 100
C
C*************************************************************
C*          PATTEERN MOVE DO LCOP.                           *
C*************************************************************
C
   30    DC 40 I=1,N
         XB(I)=XB(I)+(XB(I)-XBP(I))
   40    CCNTINUE
         IF(F(XB).LT.F(XR))THEN
            DC 45 I=1,N
            XBP(I)=XR(I)
            XB(I)=XB(I)
   45    CCNTINUE
          ELSE DO
            DO 47 I=1,N
            XB(I)=XR(I)
   47    CCNTINUE
         END IF
         GC TO 25
   50    PRINT,'OPTIMUM VALUES ARE:'
         PRINT 200,(XB(I),I=1,N)
  200    FORMAT(/,5X,'WN=',F9.5,/,5X,'WP=',F9.5,/)
         PRINT 300,F(XB)
  300    FORMAT(5X,'DELAY*AREA=',F9.4,/)
         STOP
         END
C*************************************************************
C*    THIS IS THE MULTI-VARIABLE FUNCTION WHICH HAS TO       *
C*    BE MINIMIZED.                                          *
C*************************************************************
```

```
      FUNCTION F(X)
C-----------------------------------------
C             Wn NMOS channel width in microne
C             WP PMOS channel width inmicrone
C             CL load capacitance in PF
C             delay time will be in NS
C-----------------------------------------
C
       REAL X(2),UP,UN,DP,DN,WN,WP
       REAL LN,LP,BRKT1,BRKT2,KPP,KPN
         WN=X(1)
         WP=X(2)
           COX=6.9E-4
           KPN=50.E-6
           KPP=16.E-6
           UN=775.E8
           UP=250.E8
           VDD=5
           VTN=0.7
           VTP=0.8
           DN=8
           DP=8
           LN=3.
           LP=3.
           CJAN=4.4E-4
           CJSWN=4.0E-4
           CJAP=1.5E-4
           CJSWP=4.0E-4
        WDN=WN
        WDP=WP
       BN=(KPN*WN)/LN
       BP=(KPP*WP)/LP
       BRKT1=(VTN-.1*VDD)/(VDD-VTN)+ALOG((19*VDD-20*VTN)/VDD)/2.
       RN=2*BRKT1/(BN*(VDD-VTN))
       BRKT2=(VTP-.1*VDD)/(VDD-VTP)+ALOG((19*VDD-20*VTP)/VDD)/2
       RP=2*BRKT2/(BP*(VDD-VTP))
        IF(WN .LT. 6)WDN=6
        IF(WP .LT. 6)WDP=6
       CDN=CJAN*WDN*DN+2*(WDN+DN)*CJSWN
       CDP=CJAP*WDP*DP+2*(WDP+DP)*CJSWP
         CD=(CDP+CDN)
       CGN=WN*LN*COX
       CGP=WP*LP*COX
         CG=CGN+CGP
       CL1=3*CD+CG
       CL2=2*CD+2*CG
       CL3=2*CD+CG
       CLOAD=0.05
       RPASS=(RP*RN)/(RP+RN)
       TF1=RN*CL1
       TR2=RP*CL2
       TF3=RPASS*CL3
       TR4=RP*(CLOAD+CD)
       TD=TF1+TR2+TF3+TR4
       F=(WN+WP+30)*(TD*1E-3)
      RETURN
      END
```

APPENDIX (D)

Consider the circuit of Fig. 4.28(a) which shows a series connection of a single PMOS and K NMOS transistors. All the transistors are on except the one which is nearest to ground (MK). At steady-state, node N1 will be charged up to Vdd, since the PMOS transistor passes logic 1 without any degradation . Then:

$V_{N1} = V_{dd}$

The NMOS transistors pass logic 1 with some degradation. The voltage at the source of any NMOS transistor will

$$\text{Min } [\, V_g - V_{tn} \,,\, V_d \,]$$

Where Vd is the NMOS drain voltage. Therefore

$$V_1 = V_{dd} - V_{tn1}$$

The value of Vtn1 is obtained from the following simultaneous equations :

$$V_1 = V_{dd} - V_{tn1}$$

$$V_{tn1} = V_{tOn} + \gamma * [\sqrt{V_{N1} - 2\,\phi_{Fn}} - \sqrt{2\,\phi_{Fn}}\,]$$

In the same way it can be shown that all the other nodes will be charged up to approximately (Vdd - Vtn1).

APPENDIX E

In the critical path of the SSDL circuit, Fig. 4.28(a)) there are a number of NMOS transistor in series discharging a node. This will cause an increase in source-to-substrate voltage of the transistors which are further apart from the ground. This in turn will affect the threshold voltage of the devices [36]. The maximum threshold voltage along the channel (at pinch-off point) can be expressed as follows:

$$V_{tn(max)} = V_{tn0} + \gamma_n\left(\sqrt{\gamma_n^2/4 - v_{tn0} + V_g + \gamma_n\sqrt{2\,\phi_{Fn}} + 2\,\phi_{Fn}} - \sqrt{2\,\phi_{Fn}} - \gamma_n/2\right)$$

where

V_{TOn} : Threshold voltage at zero source to substrate voltage

γ_n : Bulk threshold parameter

ϕ_{Fn} : Fermi potential

V_g : Gate voltage

An average value for Threshold voltage is defined as :

$$V_{tn(av)} = [\,V_{tn(max)} + V_{tOn}\,]/2$$

REFERENCES

[1] K. Marrin,"DSP: A Technology in Search of Application," Computer Design, Nov. 15, 1986, pp. 59-77

[2] M.A. Bayoumi, "VLSI Implementation of Residue Number System Architecture," Ph.D. dissertation, University of Windsor, Windsor, Ontario, 1985

[3] M. Hatamian and G. L. Cash, "A 70 MHz 8-bitx8-bit Parallel Pipelined Multiplier in 2.5um CMOS," IEEE J. Solid-State Circuits, Vol. SC-21, No. 4, August 1986, pp. 505-513

[4] K. Hwang, Computer Arithmetic Principles, Architecture, and Design. New York, John Wiley & Sons, 1979

[5] F.J. Mowle, A Systematic Approach to Digital Logic Design, Reading MA: Addison-Wesley, 1976

[6] J.Y. Lee, H.L. Garvin, and C.W. Slayman, "A High-Speed High-Density Silicon 8x8-bit Parallel Multiplier," IEEE J. Solid-State Circuits, Vol. SC-22, No. 1, February 1987, pp.35-40

[7] N.S. Szabo and R.I. Tanaka, Residue Arithmetic and its Application to Computer Technology, New York, McGraw-Hill, 1967

[8] G.A. Jullien , "Residue Number Scaling and other Operations using ROM arrays," IEEE trans. Comput., Vol. C-27, April 1978, pp. 325-336

[9] M.A. Bayoumi, G.A. Jullien and W.C. Miller, " A VLSI Implementation of Residue Adders," IEEE Trans. Circuits and System, Vol. CAS-34, No. 3, March 1987, pp. 284-288

[10] N. Tomabechi, M. Kameyama, and T. Higuchi, "Design of LSI-oriented Digital Signal Processing Systems based on Pulse-Train Residue Arithmetic Circuits, IECE Japan Trans., Vol. J68-D, August 1985, pp. 1457-1464

[11] A. Fisher and H.T. Kung, "Synchronizing Large VLSI Processor Arrays," IEEE Trans. Computers, Vol. C-34, No. 8, August 1985, pp. 734-740

[12] T.W. Williams and K.P. Parker, "Design for Testability - A Survey," Proc. IEEE, Vol. 71, No. 1, Jan. 1983, pp.98-112

[13] J. R. Pfiester, J. D. Shott, and J.D. Meindl, "Performance limits of CMOS ULSI," IEEE J. Solid-State Circuits, Vol. SC-20, No. 1, Feb. 1985, pp.253-263

[14] A. Mukherjee , Introduction to NMOS and CMOS VLSI System Design, Printice-hall, Englewood Cliff, N.J., 1986

[15] J.Y. Chen, " CMOS - The Emerging VLSI Technology," IEEE Circuits and Devices Magazine, Vol. 2, No. 2, March 1986, pp.16-29

[16] S. Whitaker, "Pass-Transistor Networks Optimize NMOS Logic,"Electronics, Sep. 22, 1983, pp. 144-148

[17] Y. Suzuki, K. Odagawa, and T. Abe, "Clocked CMOS Calculator Circuitry," IEEE J. Solid-State Circuits, Vol. SC-8, No. 6, Dec. 1973, pp. 462-469

[18] V. Friedman and S. Liu, "Dynamic Logic CMOS Circuits," IEEE J. Solid-State Circuits, Vol. SC-19, No. 2, April 1984, pp. 263-266
[19] R.H. Kramback, C.M. Lee, and H.S. Law, "High-Speed Compact Circuits with CMOS," IEEE J. Solid-State Circuits, Vol. Sc-17, No. 3, June 1982, pp. 614-619
[20] L.G. Heller, W.R. Griffin, J.W. Davis and N.G. Thomas, "Cascode Voltage Switch Logic: A Differential CMOS Logic Family," IEEE Int. Solid-State Circuits Conf., Feb. 1984, pp.16-17
[21] J.A. Pretorius, A.S. Shubat and C.A.T. Salama, "Latched Domino CMOS Logic," IEEE J. Solid-State Circuits, Vol. SC-21, No. 4, August 1986, pp.514-522
[22] T.A. Grotjohn and B. Hoeflinger, " Sample-Set Differential Logic (SSDL) for Complex High-Speed VLSI," IEEE J. Solid-State Circuits, Vol. SC-21, No. 2, April 1986, pp. 367-369
[23] V.G. Oklobdzija and R.K. Montoye, "Design-Performance Trade-offs in CMOS-Domino Logic," IEEE J. Solid-State Circuits, Vol. SC-21, No. 2, April 1986, pp. 304-306
[24] J.A. Pretorius, A.S. Shubat and C.A.T. Salama, "Charge Redistribution and Noise Margin in Domino CMOS Logic," IEEE Trans. Circuits and Systems, Vol. CAS-33, No. 8, August 1986, pp. 786-793
[25] K.M. Chu and D.I. Pulfrey, " Design Procedures for Differential Cascode Voltage Switch Circuits," IEEE J. Solid-State Circuits, Vol. SC-21, No. 6, December 1986, pp. 1082-1087
[26] W.C. Elmore, " The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," J. Appl. Phys., Vol. 19, No. 1, Jan. 1948, pp. 55-63
[27] P.K. Chan, " An Extension of Elmore's Delay and its Application for Timing Analysis of MOS Pass Transistor Networks," IEEE Trans. Circuits and Systems, Vol. CAS-33, No. 11, November 1986, pp. 1149-1152
[28] W. T. Lynch and H.T. Boll, "Optimization of the Latching Pulse for Dynamic Flip-Flop Sensor," IEEE J. Solid-State Circuits, Vol. SC-9, No. 2, April 1974, pp. 49-55
[29] M. Shoji, "FET Scaling in Domino CMOS Gates," IEEE J. Solid-State Circuits, Vol. SC-20, No. 5, October 1985, pp. 1067-1071
[30] N. Weste and Kamran Eshraghian, "Principle of CMOS VLSI Design: A System Perspective. Reading MA: Addison-Wesley, 1985
[31] S.A. Al-arian and D.P. Agrawal, "Physical Failures and Fault Models of CMOS circuits," IEEE Trans. Circuits and System, Vol. CAS-34, No. 3, March 1987, pp. 269-279
[32] S. M. Kang, "A Design of CMOS Polycells for LSI Circuits," IEEE Trans. Circuits and Systems, Vol. CAS-28, No. 8, August 1981
[33] T. Uehara and W.M. Vancleemput, "Optimal Layout of CMOS Functional Arrays," IEEE Trans. Computers, Vol. C-30, No. 5, May 1981, pp. 305-312

[34] C. Mead and L. Conway, Introduction to VLSI Systems. Reading MA: Addison-Wesley, 1980
[35] N.F. Goncalves and H.J. De Man, "NORA: A Dynamic CMOS Technique for Pipelined Structure," IEEE J. Solid-State Circuits, Vol. SC-18, No. 3, June 1983, pp. 261-266
[36] J.A. Pretorius, A.S. Shubat, and C.A.T. Salama, " Analysis and Optimization of Domino CMOS Logic with Application to Standard cells," IEEE J. Solid-State Circuits, Vol. SC-20, NO. 2, April 1985, pp. 523-530

BIBLIOGRAPHY

[1] N. Weste and K. Eshraghian, Principle of CMOS VLSI Design: A System Perspective. Reading MA: Addison-Wesley, 1985

[2] L. A. Glasser and D.W. Dobberpuhl, The Design and Analysis of VLSI Circuits. Reading MA: Addison-Wesley, 1985

[3] C. Mead and L. Conway, Introduction to VLSI System. Reading MA: Addison-Wesley, 1980