2005

Investigation into arithmetic sub-cells for digital multiplication.

G. Michael Howard
University of Windsor

Investigation into Arithmetic Sub-Cells for Digital Multiplication

by

G. Michael Howard

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

Windsor, Ontario, Canada

2005

Your file Votre référence
ISBN: 0-494-09742-6
Our file Notre référence
ISBN: 0-494-09742-6

Abstract

With the continually growing use of portable computing devices and increasingly complex software applications, there is a constant push for lower power high speed circuitry to support this technology. Because of the high usage and large complex circuitry required to carry out arithmetic operations used in applications such as digital signal processing, there has been a great focus on increasing the efficiency of computer arithmetic circuitry. A key player in the realm of computer arithmetic is the digital multiplier and because of it's size and power consumption, it has moved to the forefront of today's research.

As the main determining factor with regards to the performance characteristics of a multiplier, the most studied aspect of digital multiplication is the partial product reduction circuitry. Traditionally partial product reduction has been carried out through the use of carry-save adders consisting of rows of (3,2) counters otherwise known as full-adders. More recently, a focus has been put on higher-order reduction mainly through the use of 4:2 compressors.

A study of several low-power and high-speed (3,2) counter designs is initially presented, then based on these findings, a new 4:2 compressor design is introduced and proven against other existing and newly devised 4:2 compressors using various logic styles. The results obtained with regards to speed, power and size were used to categorize the circuits in terms of individual and cumulative performance characteristics. A complete 16-bit multiplier design which uses a highly efficient layout scheme along with the top performing 4:2 compressor form the above study is presented. A second multiplier using industry standard (3,2) counters in a 4:2 compressor configuration following the same optimized layout scheme is constructed and simulated as a benchmark for comparison to the new design.

In order to carry out this investigation, the proper methodology for power measurement in pass-logic circuits was developed and is presented within. This survey offers an unpartisan approach to power measurement, and an accurate reflection of the vantage points of each logic style. Moreover, with a growing interest in the applications of asynchronous circuitry, average delay, in addition to worst case delay, has been considered.

To my parents.

Acknowledgments

There are several people who deserve my sincere thanks for their generous contributions to this thesis.

I would first like to express my sincere gratitude to my supervisor Dr. Majid Ahmadi for all of his generous support and guidance. His guidance and advice both academically and personally have had, and will continue to have, a tremendous impact on my life. He has been an excellent mentor to me. To him I am deeply indebted.

I would also like to thank the faculty at the University of Windsor, including Dr. W.C. Miller and Dr. M. Sid-Ahmed for the many conversations and advice offered, and my committee members, Dr. Arunita Jaekel and Dr. Huapeng Wu for their patience and support.

Ms. Shelby Marchand also deserves much thanks and recognition. Her dedication to the students is second to none. For all of her assistance and cheerful conversation I am extremely appreciative.

To Mr. Till Kuendiger, Mr. Pedram Mokrian and Mr. Kris Perta I also offer great gratitude. Their sound technical advice has not only contributed to this thesis but their friendship made the late nights in the lab all the more bearable.

Finally, to my parents and my girlfriend I must extend my most sincere love and gratitude. For every path I have chosen, they have always shown unending support and enthusiasm.

Table of Contents

List of Figures

List of Tables

Table 4.6 16-bit Partial Product Reduction Delay Using PASS2 4:2 Compressors . 59
Table 4.7 16-bit Multiplier Delay Using CMOS1 4:2 Compressors 60
Table 4.8 16-bit Multiplier Delay Using PASS2 4:2 Compressors 60
Table 4.9 Summarized Simulation Results 61
Table 5.1 Compressor Recommendations in Terms of System Requirements 65

List of Abbreviations

ALU	*Arithmetic Logic Unit*
ASIC	*Application Specific Integrated Circuit*
CLA	*Carry Look-Ahead Adder*
CMOS	*Complementary Metal-Oxide Semiconductor*
CPU	*Central Processing Unit*
CSA	*Carry Save Adder*
DRC	*Design Rule Check*
FA	*Full-Adder*
GND	*Ground*
HA	*Half-Adder*
HDL	*Hardware Descriptor Language*
IC	*Integrated Circuit*
IEEE	*Institute of Electrical and Electronics Engineers*
I/O	*Input / Output*
MOS	*Metal-Oxide Semiconductor*
MOSFET	*Metal-Oxide Semiconductor Field Effect Transistor*
MUX	*Multiplexer*
NFET	*N-type Field Effect Transistor*
PDP	*Power Delay Product*
PFET	*P-type Field Effect Transistor*
PP	*Partial Product*
TSMC	*Taiwan Semiconductor Manufacturing Company*
VDD	*Supply Voltage*

VHDL	*Verilog Hardware Descriptor Language*
VLSI	*Very Large Scale Integration*
VSS	*Lowest Chip Voltage (usually ground)*

Chapter 1

Introduction

1.1 Introduction

Prior to 1935, a computer was known as a person who performed arithmetic calculations or "one who computes". Computer was actually a job title during this period of time. The modern machine definition is based on von Neumann's concepts [1]: "a device that accepts input, processes data, stores data, and produces output". While technology has come a long way in the many years since von Neumann's work, the basic formula for the components of a computing system have remained the same.

Von Neumann and his associates state that "a general purpose computing machine should contain certain main organs relating to arithmetic, memory-storage, control and connection with the human operator"[1]. The arithmetic organ is known today as the ***arithmetic logic unit (ALU)***; it is required to be capable of adding, subtracting, multiplying, and dividing. This thesis deals specifically with the multiplication function of this arithmetic organ.

1.2 Thesis Highlights

This thesis will present a general investigation into digital multiplication circuitry and simulation and will highlight a new low-power 4:2 compressor circuit. The proposed 4:2 compressor utilizes the more promising aspects of existing 4:2 compressors and (3,2) counter designs. The principle advantage of this design is it's ability to provide clear, correct outputs while maintaining a low power-delay-product (PDP) and minimal circuitry. The performance of this circuit will be proven through test using a newly devised standard power measurement procedure for pass-logic circuits.

In addition, a complete multiplier design at the layout level will be presented utilizing the new 4:2 compressor circuit along with a new partial product reduction layout technique [2]. Results showing extreme savings in terms of power dissipation over a more traditional design will be presented. A final fabricated version of this multiplier using TSMC 0.18 μm technology will also be presented.

1.3 Thesis Organization

The thesis will begin with a general overview of the concept of digital multiplication, standard notation, and some various multiplication algorithms in Chapter 2. Most importantly, this chapter will present the fundamentals of partial product reduction.

Chapter 3 will initially introduce both the (3,2) counter and the 4:2 compressor in terms of basic functionality. This will be followed by an in-depth analysis of the circuitries and logic styles used in the construction of these cells. This analysis will include the presentation of existing 4:2 compressor designs as well as some new proposed designs. This chapter will conclude with the simulation and comparison of these circuitries in terms of power and delay using a new proposed standard power measurement procedure for pass-logic circuits.

The design and test of a 16-bit multiplier will be dealt with in Chapter 4 beginning with the partial product reduction layout scheme followed by schematic generation of the multiplier. This will include the presentation of Hardware Descriptor Language (HDL) code for both the architecture of the multiplier and it's functional validation. Chapter 4 continues with an introduction to custom layout focusing on it's importance in the design of the multiplier. The layout for this 16-bit multiplier will then be presented along with the rational behind the considerations taken in order to make this design suitable for fabrication. This chapter will conclude with the presentation and discussion of results obtained through computer simulation of the laid out multiplier and physical test of the fabricated multiplier.

This thesis will conclude with a summary of contributions and conclusions in Chapter 5.

Chapter 2

Digital Multiplication

2.1 Basic Digital Multiplication

Basic digital multiplication is performed very similarly to our traditional pen and paper method. An *nxm*-bit multiplication of a ***multiplier X*** (2.1) and a ***multiplicand A*** (2.2) yield a final ***product P*** (2.3).

$$X = [x_{n-1}, x_{n-2}, x_{n-3}, \ldots x_2, x_1, x_0] \tag{2.1}$$

$$A = [a_{m-1}, a_{m-2}, a_{m-3}, \ldots a_2, a_1, a_0] \tag{2.2}$$

$$\begin{aligned} P = [p_{n+m-1}, p_{n+m-2}, p_{n+m-3}, \cdots p_2, p_1, p_0] \\ = x_{n-1}(a_{m-1}, a_{m-2}, a_{m-3}, \ldots a_2, a_1, a_0) \\ + x_{n-2}(a_{m-1}, a_{m-2}, a_{m-3}, \ldots a_2, a_1, a_0) \\ \ldots \\ + x_1(a_{m-1}, a_{m-2}, a_{m-3}, \ldots a_2, a_1, a_0) \\ + x_0(a_{m-1}, a_{m-2}, a_{m-3}, \ldots a_2, a_1, a_0) \end{aligned} \tag{2.3}$$

Multiplication is carried out through the *bitwise logical-ANDing* of multiplier X and multiplicand A to produce a matrix of ***partial products***. These products are then reduced through various methods to obtain the final product *P*. Figure 2.1 provides an example of a 4x4-bit multiplication.

				a_3	a_2	a_1	a_0
				x_3	x_2	x_1	x_0
				x_0a_3	x_0a_2	x_0a_1	x_0a_0
			x_0a_3	x_0a_2	x_0a_1	x_0a_0	
		x_0a_3	x_0a_2	x_0a_1	x_0a_0		
	x_0a_3	x_0a_2	x_0a_1	x_0a_0			
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Figure 2.1 Multiplication of 2 4-bit numbers

2.1.1 Dot Diagrams

In order to make digital multiplication algorithms easier to visualize, the concept of ***dot diagrams*** is introduced. Referring to Figure 2.2, each dot represents one binary bit. Figure 2.2 illustrates a dot diagram for a *4x4-bit* multiplication.

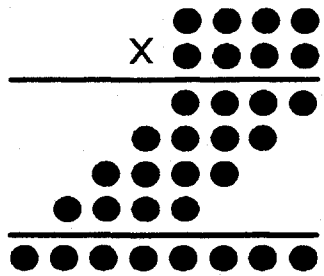


Figure 2.2 Dot Diagram Representation of 4x4-bit Multiplication

2.2 Sequential Multiplication

2.2.1 Basic Sequential Multiplication

The sequential multiplier has the most simple hardware realization of all multiplier designs. It realizes the basic shift/add multiplication algorithm through the use of

registers, multiplexors, and adders. Figure 2.3 shows an example of this hardware realization. Three registers are needed for the multiplication of two *kxk*-bit operators. The partial product register is initialized to *0*. *k* multiplexors *(MUX)* receive all *k-bits* of the multiplicand as an input and consecutive bits of the multiplier beginning with the least significant as control signals. With each new input bit from the multiplier, the outputs of the multiplexors are added with the *k* most significant bits of the intermediate partial product. The result of this addition is again stored in the partial product register and shifted right *1*-bit.

In the case of *left-shift* multiplication, a similar structure is used. The difference being that the most significant bits of the multiplier are first shifted in and a *2k-bit* adder is instead required for proper partial and final product generation. Because of the slightly larger circuit size, right-shift multipliers are preferred for sequential multiplication.

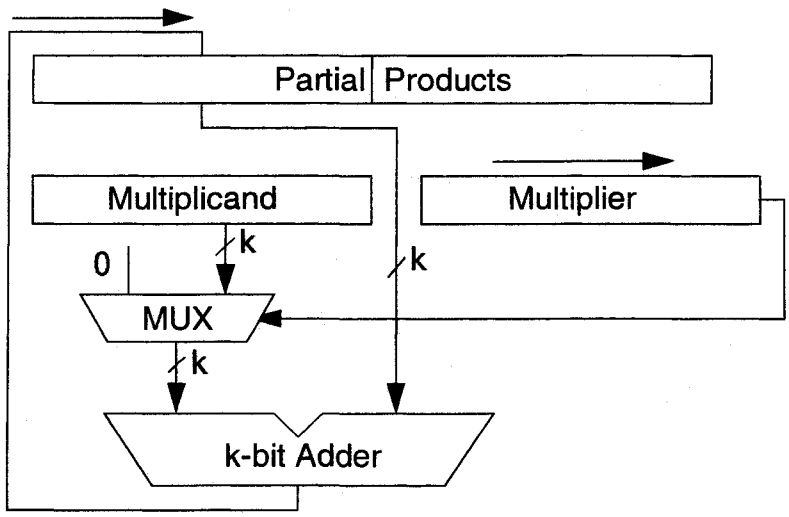


Figure 2.3 Sequential Right-Shift Multiplier

2.2.2 High-Radix Multiplication

In order to speed up the multiplication process, the idea of ***high-radix*** multiplication was introduced. Essentially this is performed by splitting the *k-bit* multiplier into a (*k/n*)-digit,

radix-2^n number. This reduces the number of partial products to *k/n* and in the case of sequential multiplication, reduces the number of additions to *k/n*. Figure 2.4 illustrates this in the form of a *4x4-bit* multiplication performed in *radix-4*.

This form of multiplication relies on the fact that the multiples (*0A*, *1A*, *2A*, and *3A*) of the multiplicand are readily available. *0A*, *1A*, and *2A* (left-shift) are very straight forward, however, *3A* requires two operations (*1A+2A*). This requires additional pre computational circuitry and registers that make this method of multiplication costly in terms of hardware.

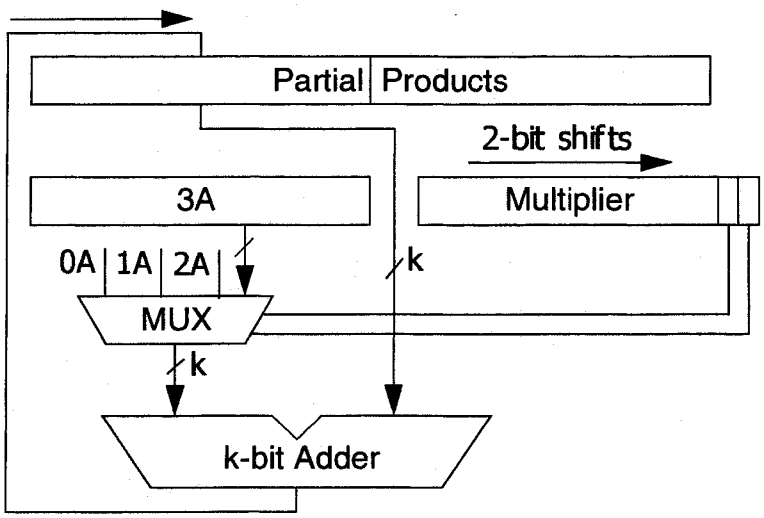


Figure 2.4 Multiplication Performed Using Radix-4

2.3 Parallel Multipliers

Parallel multipliers, otherwise known as ***full-tree multipliers*** can be viewed as a special case of high-radix multiplication. In this case the highest possible radix is used (*radix-2^k*). In other words, all partial products are generated at the same time. These partial products are then reduced through the use of a ***carry-save-adder (CSA)*** summation network producing two partial products which are then summed using a final ***carry-propagate*** or ***fast-adder.*** The block diagram Figure 2.5 depicts this process. Figure 2.6 also illustrates

this process but does so in dot notation form. These types of multipliers may have a high cost in terms of realestate but are well worth it in applications where speed is critical.

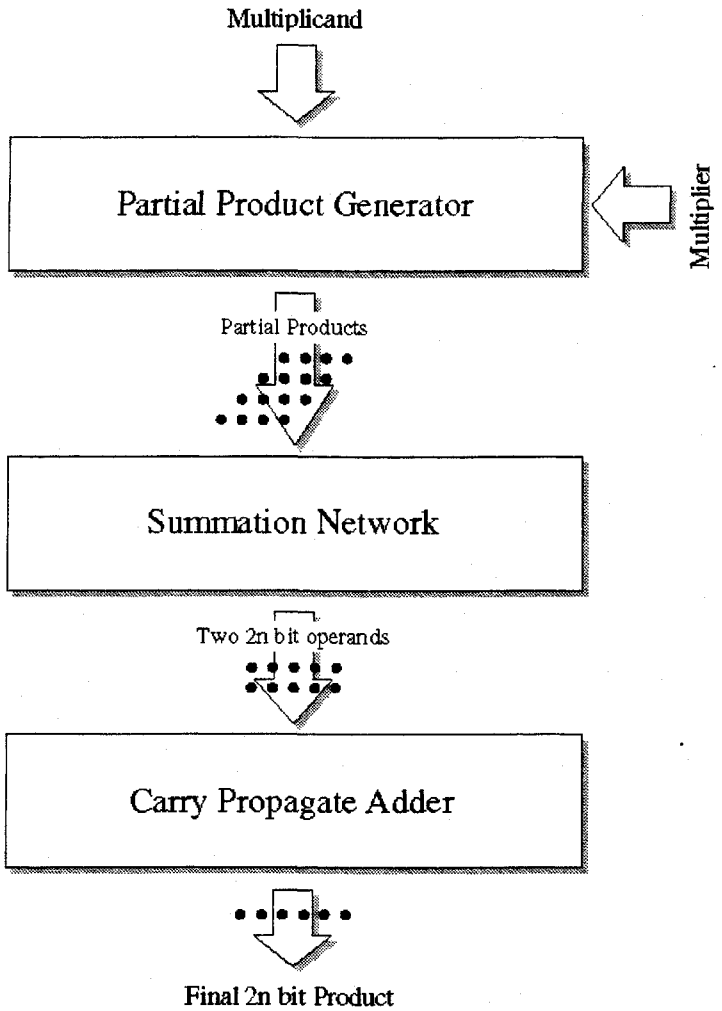


Figure 2.5 Parallel or Full-Tree Multiplier

2.3.1 Partial Product Reduction

The method in which partial products are reduced is the main determining factor with regards to the performance of parallel multipliers. Hence partial product reduction is the most studied aspect of parallel multiplication. As mentioned, traditional partial product reduction is carried out through the use of a CSA tree in column compression format. Figure 2.7 depicts the layout of a conventional CSA. Column compression involves

reducing bits in a each column partial products by a factor of 3:2. Figure 2.8 [3] illustrates the addition of seven 6-bit numbers using a CSA column compression format. Partial product reduction using CSA column compression is usually carried out using either a ***Wallace*** or ***Dadda*** tree format.

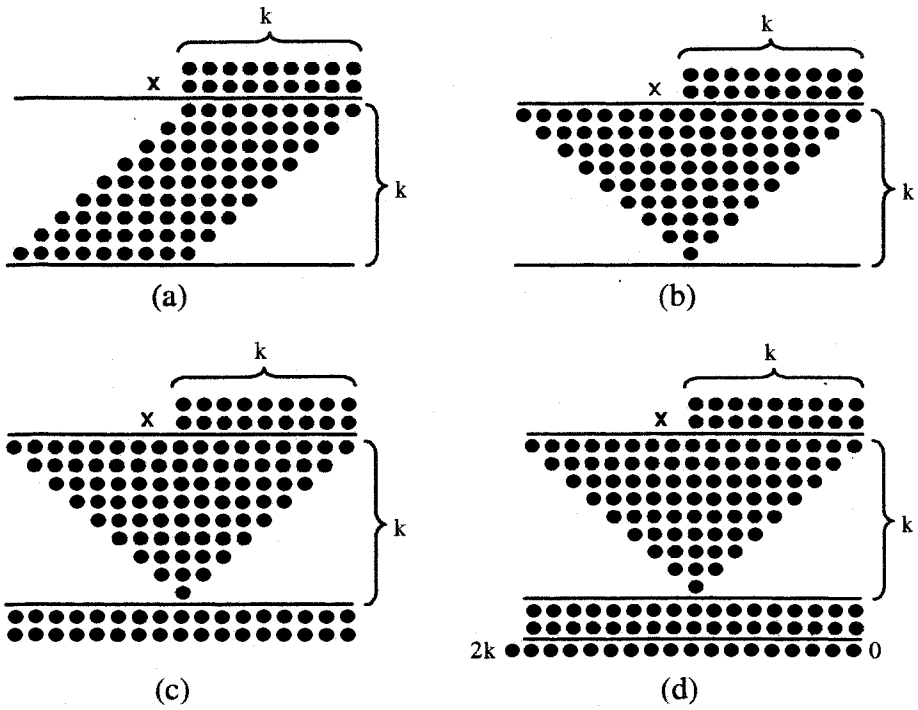


Figure 2.6 Parallel Multiplier Dot Representation

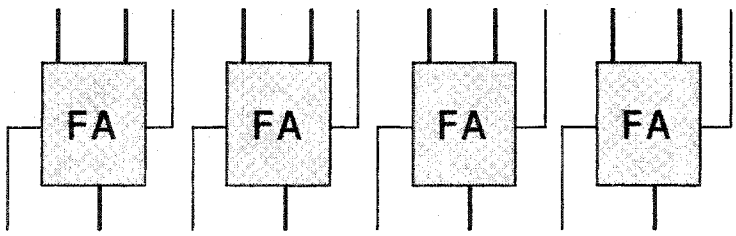


Figure 2.7 Carry Save Adder (CSA)

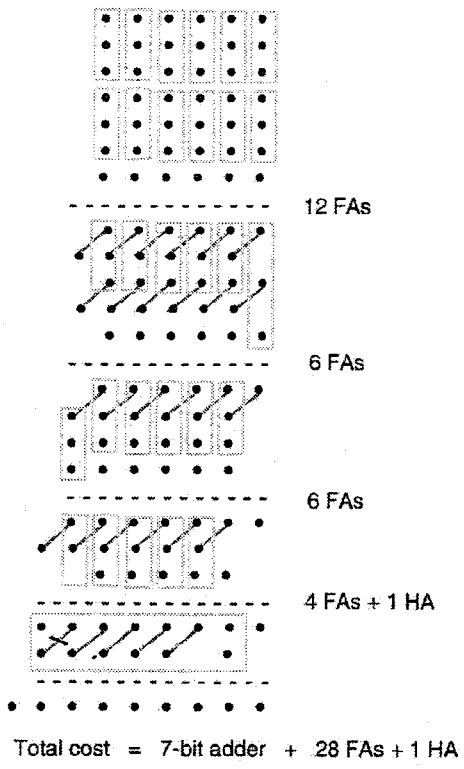


Figure 2.8 Column Compression of 7 6-bit Numbers

Wallace Trees

In 1964 Wallace [4] introduced a new column compression architecture for fast multiplication as an alternative to array multiplication. His scheme involves three basic steps:

1. Generate all partial products at the same time using AND gate array.
2. Reduce all partial products to two numbers using (3,2) and (2,2) counters.
3. Sum the two final numbers using some form of fast addition such as a carry-lookahead-adder (CLA).

In contrast to the linear growth of delay as word length increases in array multipliers, when using this column compression architecture, delays proportional to the logarithm of the operand word size may be achieved. Therefore, column compression parallel multipliers are faster than array multipliers.

Wallace's method involves grouping all rows in each stage of partial product reduction into groups of three during each reduction stage. All columns in each group containing 3 bits are reduced using ***(3,2) counters***, also known as ***full-adders***, and all columns containing 2 bits are reduced using ***(2,2) counters***, also known as ***half-adders***. All rows that are not part of a three row set are then transferred to the next stage without modification. It is apparent that the Wallace method for column compression reduces the most digits at the earliest possible time. Figure 2.9(a) shows the reduction process for a 12x12-bit multiplication.

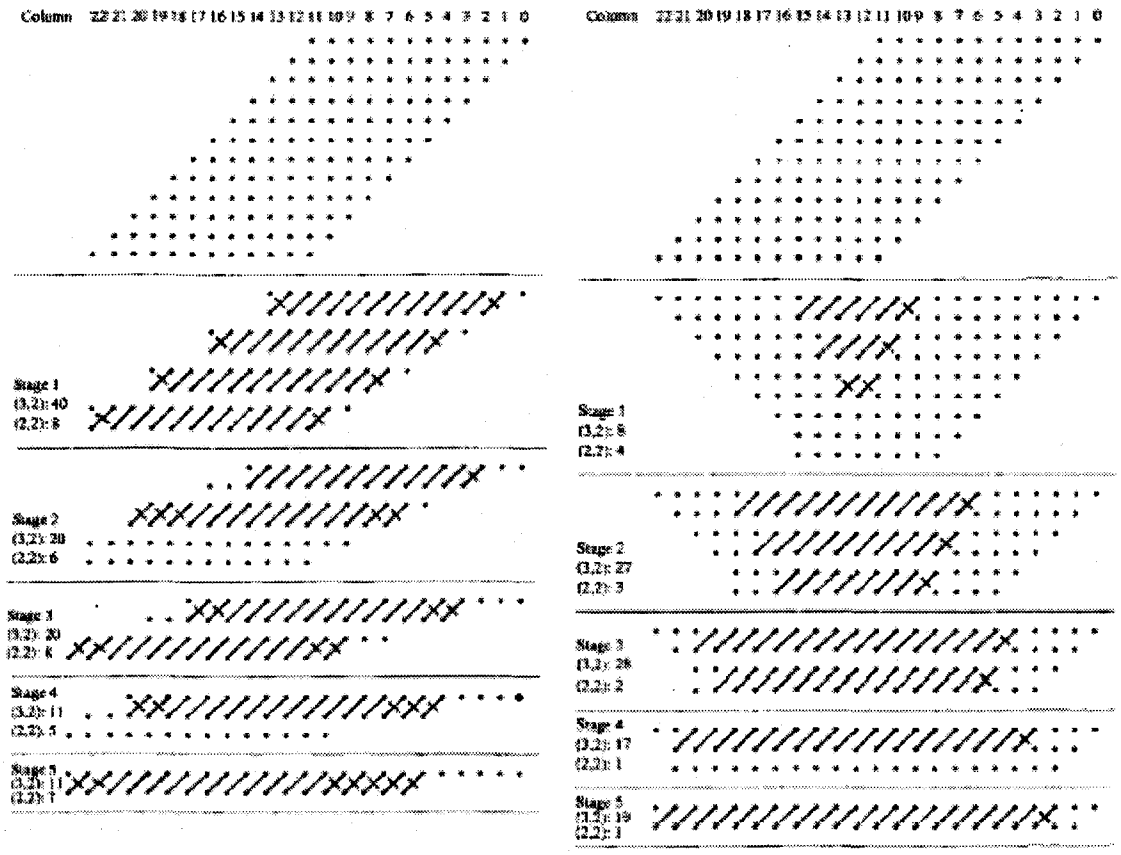


Figure 2.9 (a) Wallace Compression (b) Dadda Reduction [5]

Dadda Trees

Shortly after Wallace presented his method for partial product reduction using CSA column compression, Dadda [6] was able to improve on this method by utilizing a unique placement strategy for the reduction stage counters. Like Wallace's method, Dadda uses the same three step process described earlier but unlike Wallace's method of reducing as many bits as possible at the earliest possible time, Dadda's method involves strategically reducing only some columns of each stage. This is done in order to reduce the overall number of (3,2) counters required for the entire reduction process. The process for reduction in a Dadda multiplier is developed using the following method:

1. Find the smallest j such that at least one column of the original partial product matrix has more than d_j bits where dj is the height of the j^{th} stage from the end

$$d_{j+1} = \lfloor 1.5 \cdot d_j \rfloor$$
$$d_1 = 2$$

2. In the j^{th} stage from the end, employ (3,2) and (2,2) counters to obtain a reduced matrix with no more than d_j bits in any column.
3. Let $j = j\text{-}1$ and repeat step 2 until a matrix with only two rows is generated.

A dot diagram illustrating this process is presented in Figure 2.9(b). While this system reduces the number of counters required for the reduction operation, the final adder is usually required to be larger which counter balances some of the circuitry gains.

2.3.2 4:2 Compressor Reduction

Since their inception by Weinberger [7], 4:2 compressors have become the topic of considerable research in the arithmetic community. The 4:2 compressor has transformed the standard frame of mind of counter based partial product reduction schemes by introducing the notion of ***horizontal data paths*** within stages of reduction. The use of such compressors, and those of higher magnitude, in partial product reduction trees has been well documented and a variety of these circuits have been suggested [7-22]. Figure 2.10 illustrates 4:2 reduction in dot notation form.

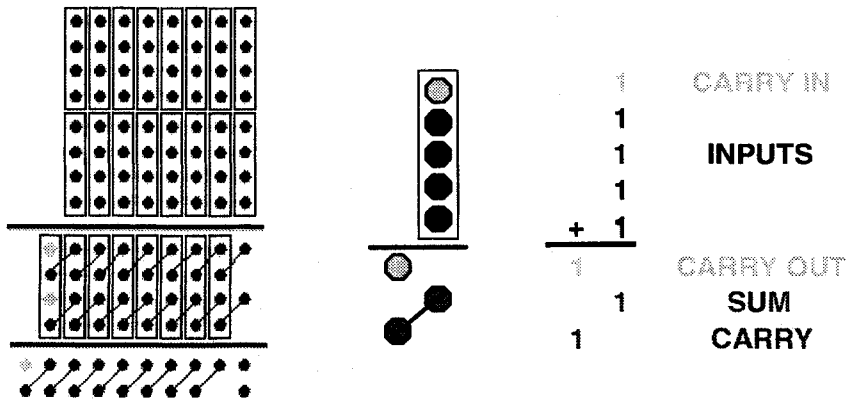


Figure 2.10 Dot Diagram of 4:2 Reduction

Until recently there have been no proposals for reduction scheme for the use of 4:2 compressors in partial product reduction. Mokrian et al. [2] introduced a layout scheme which follows the same ideas as the Dadda (3,2) counter scheme in that it minimizes the number of cells used in a complete reduction.

This layout scheme defines a compressor row as depicted in Figure 2.11. These rows begin with a half-adder or in the rightmost least significant position followed by a chain of 4:2 compressors and ending with a full-adder in the leftmost or most significant position.

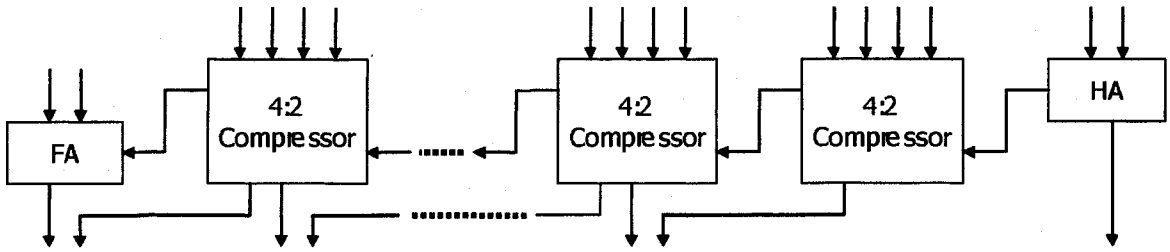


Figure 2.11 Definition of a 4:2 Compressor Row

The iterative design process defined for implementation of this scheme is as follows:

1. Determine the number of compressor rows (N_R) required for the given stage according to the equation:

$$N_R = \left\lceil \frac{n(h) - 2^{\lceil \log_2 n(h) - 1 \rceil}}{2} \right\rceil$$

2. N_R rows of 4:2 compressors are placed in the partial product reduction tree. The first row will begin at column j_{1F} and end at column j_{1L}:

$$j_{1F} = 2^{\lceil \log_2 n(h) - 1 \rceil}$$

$$j_{1L} = 2k - 1 - 2^{\lceil \log_2 n(h) - 1 \rceil}$$

Every subsequent row will begin at column:

$$j_{iF} = 2^{\lceil \log_2 n(h) - 1 \rceil} + 2i = j_{(i-1)F} + 2i$$

and end at column:

$$j_{iL} = (2k - 1 - 2^{\lceil \log_2 n(h) - 1 \rceil}) - 2i$$

where i is the row number within each stage up to N_R.

3. Repeat steps 1 and 2 until only two rows remain for fast addition.

This scheme will be graphically depicted in Chapter 4 for the 16-bit multiplier which is studied.

Chapter 3

Counters and Compressors

3.1 Counters and Compressors

An *(N,M)* counter is a device which takes N equally weighted inputs and sums them to provide an M-bit binary output. Figure 3.1 illustrates the standard counter operation. The two types of counters studied here are the (3,2) counter or full-adder and the (2,2) counter or half-adder.

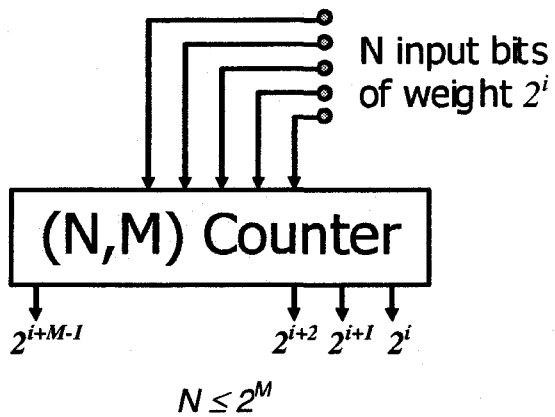


Figure 3.1 General Counter Representation

In general, *(N:M)* compressors reduce *N-input* bits to a single sum bit of equal weight to that of the inputs but unlike counters, the remaining output bits are all of equal weight: one bit-position

greater than that of the inputs. The standard representation for a (N:M) compressor is pictured in Figure 3.2. The compressor under study here is the 4:2 compressor.

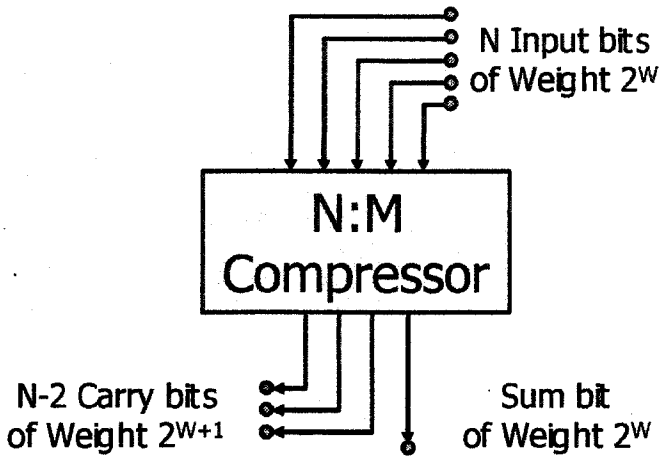


Figure 3.2 General Compressor Representation

3.2 Half-Adders

The ***half-adder (HA)*** is a two-input/two-output device which takes two equally weighted bits *(x,y)* and produces a sum-bit *(s)* as well as a carry-bit *(c)*. The sum-bit is produced through the logical ***exclusive-oring (XOR)*** of the two inputs while the carry-bit is the product of the logical ***anding (AND)*** of the input bits. The truth table for a half-adder is presented in Table 3.1 and Figure 3.3 illustrates the boolean expression for the operation and a standard symbol for this circuit

Table 3.1 HA Truth Table

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

$$s = x \oplus y$$

$$c = x + y$$

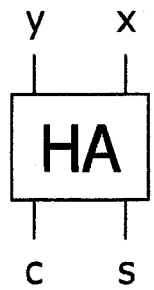


Figure 3.3 The Half-Adder

3.3 Half-Adder Circuitry

The half-adder is a simple circuit. It is so simple in fact that there is little discussion on the optimization of such cells. Standard gate level representations use two or three gates as depicted in Figure 3.4. As with most logical circuits, minimization at a transistor level will yield a smaller circuit size than that of gate level minimization. Figure 3.5 shows a transistor-level half-adder representation using 14 transistors as opposed to the 16 transistors required for standard CMOS realization of either of the gate level descriptions.

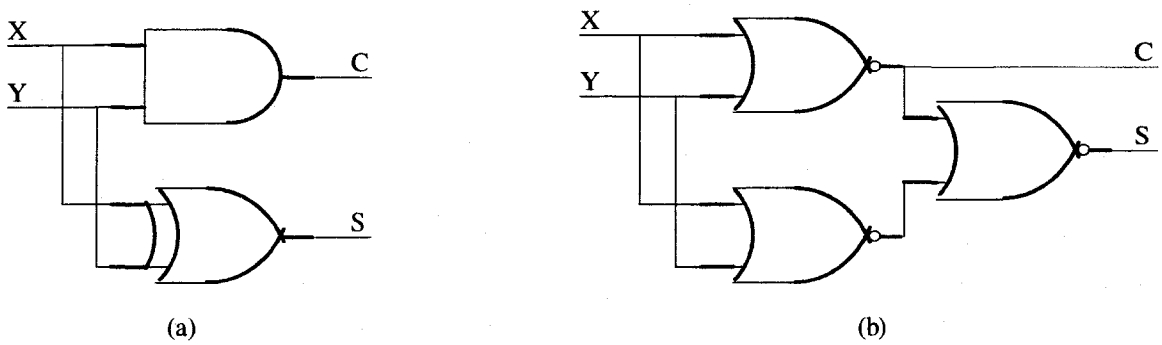


Figure 3.4 (a) AND/XOR Configuration (b) NOR Configuration

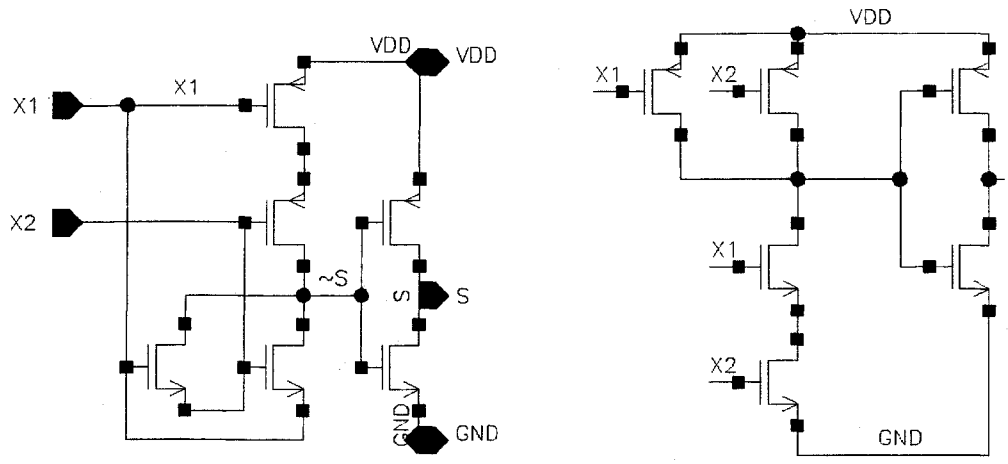


Figure 3.5 CMOS Realization of the Half-Adder

3.4 Full-Adders

The full-adder is one of the most basic building blocks in computer arithmetic. It is the main component in CSA trees and is therefore of great importance in digital multiplier design. Much of the circuitry involved in full-adder design is paralleled in that of the 4:2 compressor thus making this design of equal significance.

The full-adder is a three-input/two-output device which takes three equally weighted bits (x,y,c_{in}) and produces a sum-bit (s) as well as a carry-bit (c_{out}). The sum-bit is produced through the logical "exclusive-oring" (XOR) of all three inputs while the carry-bit is the product of the logical "anding" (AND) of all two input combinations followed by the "oring" (OR) of the results produced. The truth table for the full-adder is presented in Table 3.2 and Figure 3.6 illustrates the boolean expression for the operation and a standard symbol for this circuit.

Numerous full-adder designs have been presented over the years using both traditional and non-traditional logic styles. Several of these designs are examined with a focus on power consumption, speed, and area.

Table 3.2 FA Truth Table

C_{in}	X	Y	S	C
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

$$s = x \oplus y \oplus c_{in}$$

$$c = xy + xc_{in} + yc_{in}$$

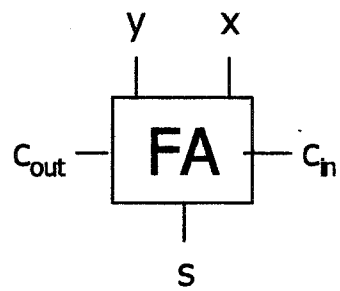


Figure 3.6 The Full-Adder

3.5 Full-Adder Circuitry

At a gate level, the full-adder appears to be a very simple design. It can very easily be represented as a pair of cascaded half-adders as shown in Figure 3.7. Again, building this circuit strictly from the gate level representations using ***standard CMOS*** gates will produce relatively large circuits. These circuits have a high cost both in terms of area and power consumption. Several circuits have been presented which reduce the circuitry through transistor level minimization. The most promising of these designs are studied here.

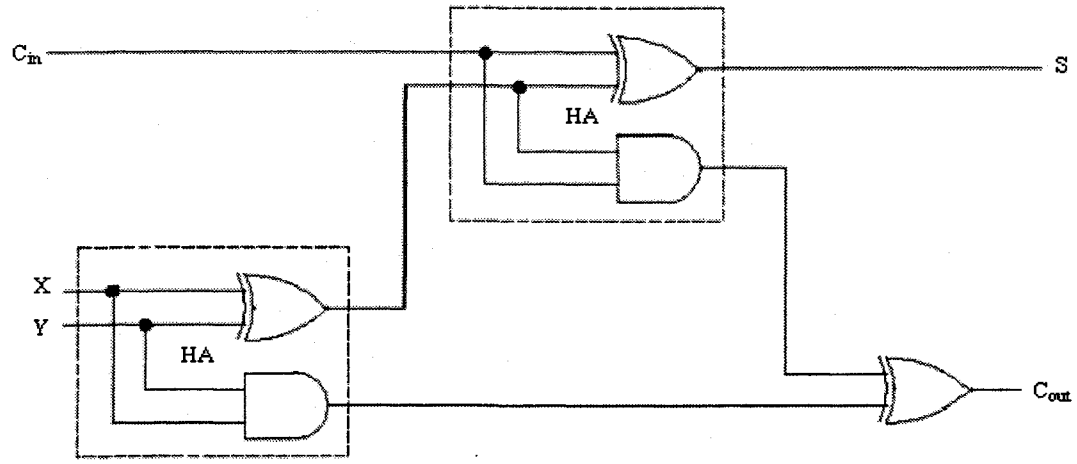


Figure 3.7 Half-Adder Representation of the Full-Adder

3.5.1 Standard CMOS Full-Adder Designs

As discussed, using standard CMOS gates to construct the full-adder comes at a high cost. To build a full-adder using the gate level representation in Figure 3.7 and standard CMOS cells, the transistor count for the entire circuit balloons to 40. Furthermore, in a ***single-rail*** circuit, these cells require a number of inverters in order to provide complemented inputs. Due to the high power requirements and additional delays associated with these inverters, this configuration is only truly feasible in a ***dual-rail*** system where these inverters might not be required.

A much more efficient standard CMOS realization of the full-adder [23] is devised by inspection of the truth table as a whole rather than building the circuit based on individual gates. This leads to a great savings in transistor count (28-transistors) resulting in dramatic decreases in delay and power. Shown in Figure 3.8, this adder design is also known as the mirror configuration FA. This adder is devised by inspection of the truth table and arranging the ***nFET*** and ***pFET*** networks accordingly.

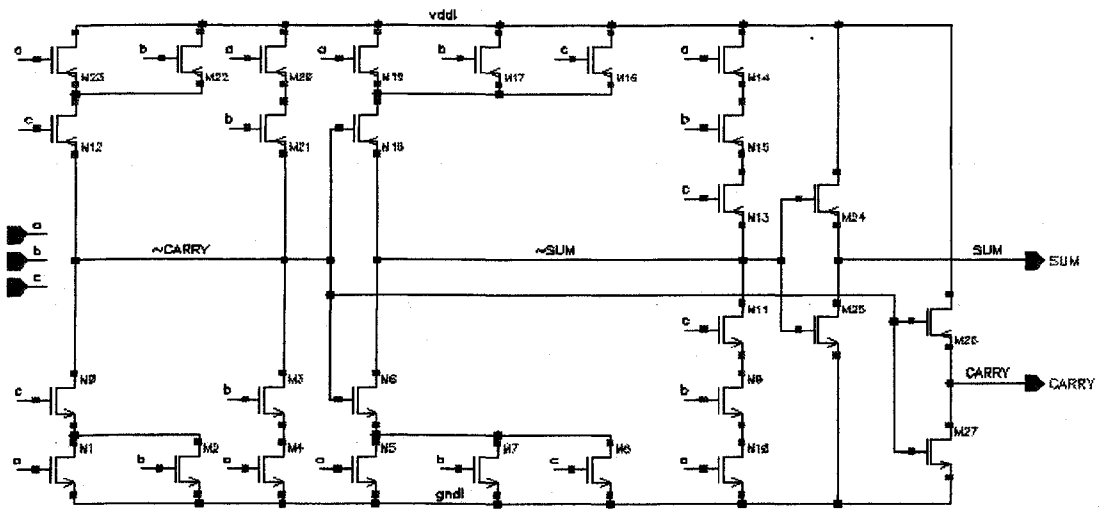


Figure 3.8 28-transistor Standard CMOS FA

3.5.2 Transmission Gate Full-Adder Design

The full-adder depicted in Figure 3.9 [24] is built using what has been coined "***transmission function theory***". It uses transmission gates and pass-logic style gates to perform the full-adder operation. This design reduces the number of required transistors dramatically from 28 for a standard CMOS adder to 16 for this design. The cost comes when considering driving capabilities. This is due to the fact that there exists conditions where input signal X_1 may be conducted through the entire circuit from input to output. This is capable of causing problems from signal degradation, added delay and even false outputs.

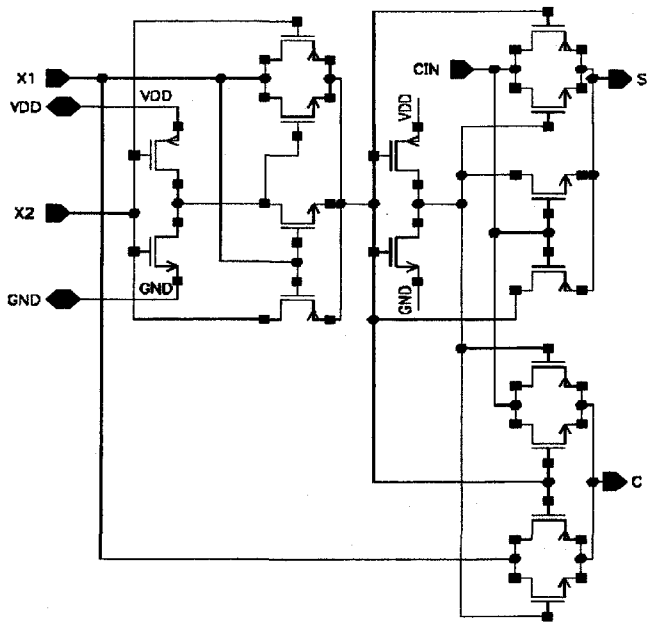


Figure 3.9 Transmission Gate FA

3.5.3 Pass-Logic Full-Adder Designs

The majority of recently proposed full-adder designs are of a ***pass-logic*** nature. Some of these designs show much promise but many of them, especially 10-transistor configurations, experience either multiple ***glitch*** conditions or suffer from signal degradation due to multiple occurrences of ***threshold voltage loss***.

The first pass-logic design studied utilizes innovative *XOR/XNOR* circuitry along with transmission style output circuitry Figure 3.10 [25]. This *XOR/XNOR* configuration is derived by combining a more traditional pass-logic *XOR* circuitry with it's paralleled *XNOR* configuration. Both of these traditional pass-logic designs suffer from threshold-voltage-loss (or gain), while this recently devised circuit provides ***full-voltage-swing*** at it's outputs. Because the *XOR* and *XNOR* are wired in a complementing manner, if there is a state where one of the functions would normally experience a loss, the other half of the circuit will drive a transistor pulling the other output to a full *"high"* or *"low"* state.

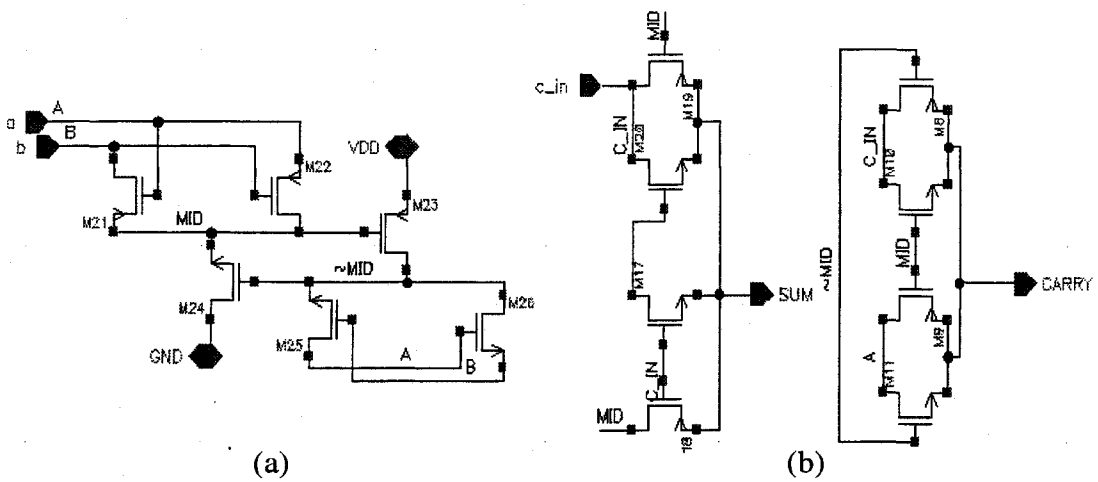


Figure 3.10 14-transistor Pass-Logic FA #1
(a) XOR/XNOR Circuitry (b) Transmission Style Output

A very similar design to that in Figure 3.10 is shown in Figure 3.11 [26]. It utilizes the same output circuitry but applies different XOR/XNOR configurations on the inputs. This XOR/XNOR circuitry does not always provide full-swing signals but because of the output circuitry, all signals leaving the cell are of a full-swing nature. This is an important power saving aspect of this design. By the definition of power, the lower the internal operating voltages, the less power is consumed within the cell. This is in effect a type of ***voltage scaling*** which is a low-power technique normally thought of in the dynamic sense, where the operating voltage of a circuitry is purposely lowered and raised as needed in accordance with clock frequency. For this circuit we can view the voltage scaling as ***input-dependant internal voltage scaling,*** meaning the internal operating voltage for the circuit changes with respect to the input combinations rather than the clock frequency.

Finally, the last full-adder studied is presented in Figure 3.12 [30]. This 10-transistor model uses similar circuitry to that of the full-adder depicted in Figure 3.11 but essentially removes all circuitry which provides guaranteed full-voltage swing at the outputs in order to minimize transistor count. This method may obtain much lower power consumption but at the cost of a usable output signal.

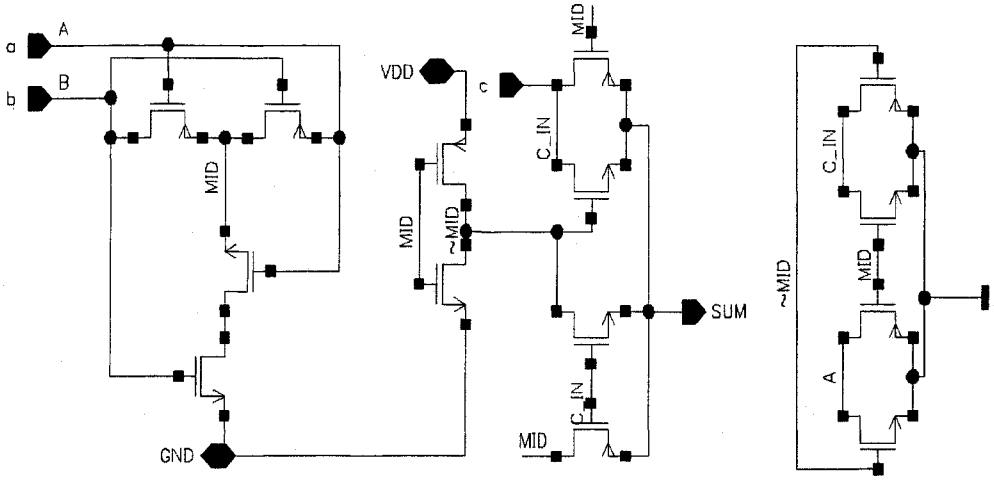


Figure 3.11 14-transistor Pass-Logic FA #2

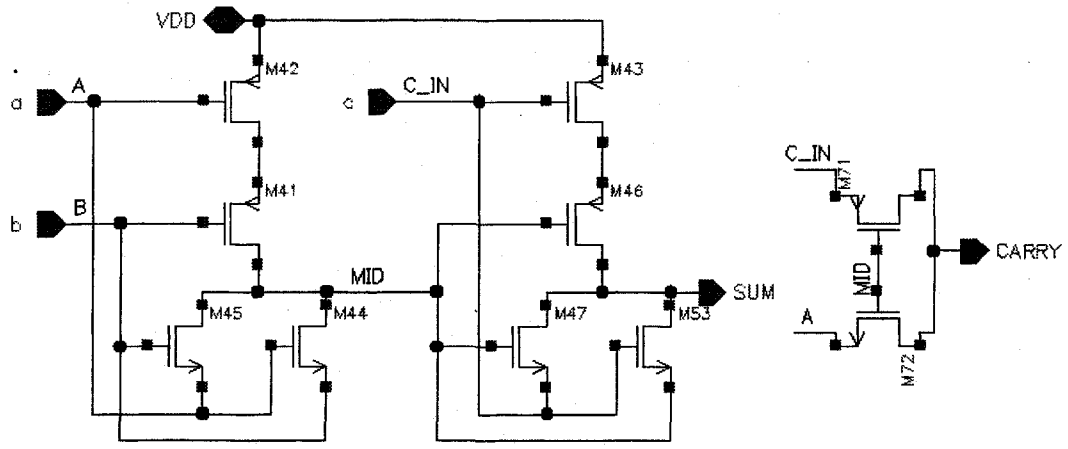


Figure 3.12 10-transistor Pass-logic FA

3.6 4:2 Compressors

The 4:2 compressor is a five-input/three-output device which takes five equally weighted inputs (C_{in}, X_0, X_1, X_2, X_3) and produces a sum-bit (S), a carry-bit (C) and a carry-propagate-bit (C_{out}) as shown in Figure 3.13. The most primitive representation of the 4:2 compressor, Figure 3.13(a), is that of two cascaded full-adders [3]. By increasing regularity, this configuration lends itself to gains at the architectural level of the multiplier

[2]. It does not however present any gains in terms of circuitry. The optimized 4:2 compressor configuration arises when the entire structure is regarded as one entity, as opposed to a composition of two full adders. This allows for further enhancement at the transistor level.

Another representation of the 4:2 compressor shown in Figure 3.14 consists of two ***multiplexers (MUX)*** and three ***exclusive-or (XOR) gates*** [12]. When building the compressor, using this representation and standard CMOS pull-up/down style logic, the transistor count for the overall circuit actually increases to a much greater number than if standard full-adders were used in the original configuration. This leads to the search for alternate design methods to reduce the size, power consumption, and the speed of the circuit. It will be shown however, that when the size of the circuit is reduced, performance in terms of power and delay do not always follow suit.

$$S = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in}$$

$$C = (X_1 \oplus X_2 \oplus X_3 \oplus X_4)C_{in} + \overline{(X_1 \oplus X_2 \oplus X_3 \oplus X_4)}X_4$$

$$C_{out} = (X_1 \oplus X_2)X_3 + \overline{(X_1 \oplus X_2)}X_1$$

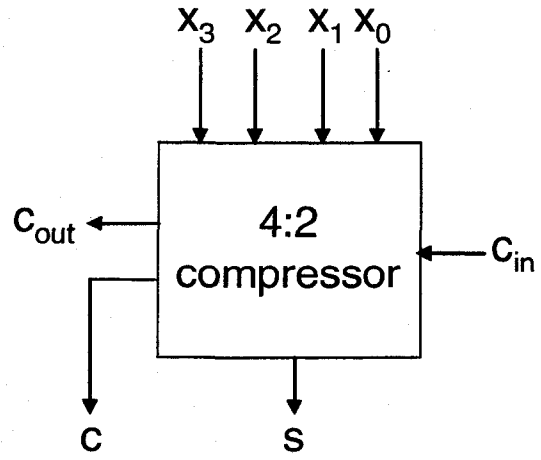


Figure 3.13 Boolean Expressions and Standard Symbol for 4:2

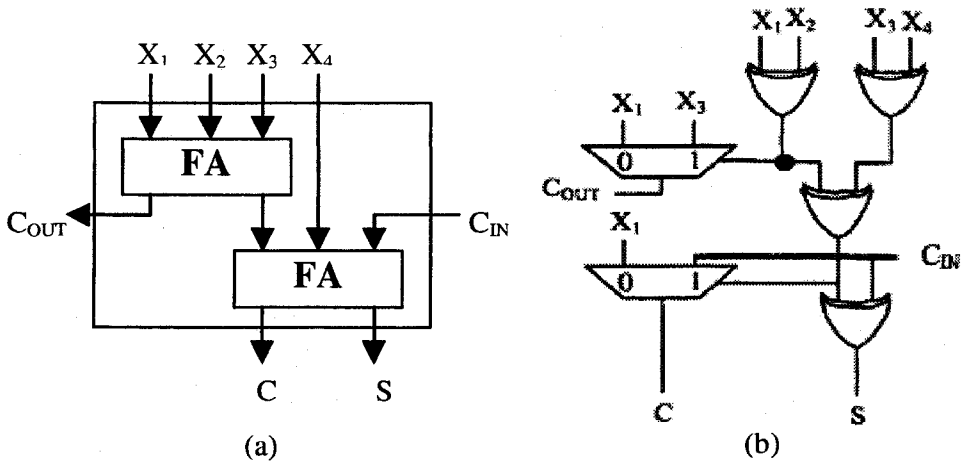


Figure 3.14 (a) FA representation of the 4:2 compressor
(b) XOR/MUX representation of the 4:2 compressor

3.7 4:2 Compressor Circuitry

3.7.1 Standard CMOS 4:2 Compressors

Two different standard CMOS 4:2 compressors are studied. The first (CMOS1) is based in the initial cascaded full-adder model of the compressor in Figure 3.14(a). It consists of two standard 28-transistor full-adders. Although this design lends no savings at the transistor level, this configuration will be used as a basis of comparison for the other designs due to its highly optimized CMOS FA design.

The second standard CMOS design studied (CMOS2) is shown in Figure 3.15. This design uses standard 10-transistor CMOS XOR/XNOR and MUX cells [3] to implement the representation of the 4:2 compressor as shown in Figure 3.14(b). This design lends itself to dual rail logic structures due to its need for complemented inputs and it's ability to provide complemented outputs. Due to the added interconnections associated with dual rail structures, the architectural benefits of the 4:2 compressor are in essence nullified in this design. Coupled with the growing concerns dealing with effects of interconnections in current and future technologies, dual rail circuits are not considered in this study.

Therefore, inverters are added at the inputs of the circuit to provide the needed complements of the input signals.

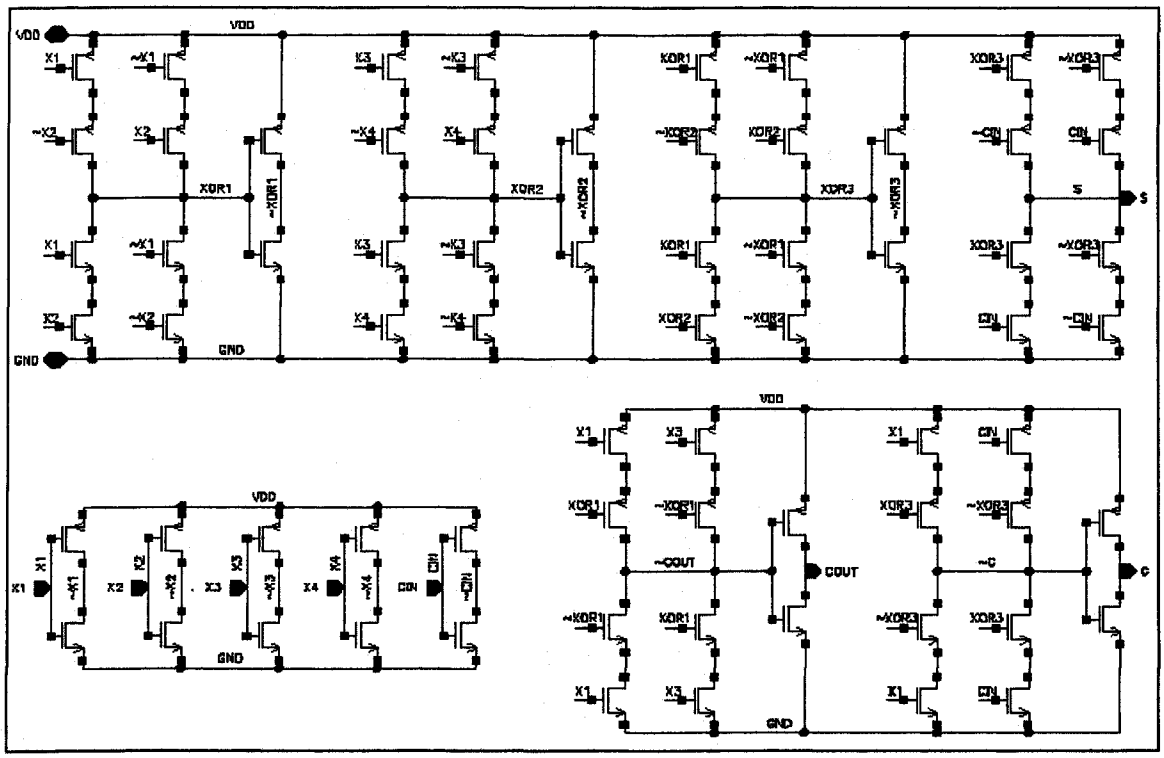


Figure 3.15 Standard CMOS XOR/MUX 4:2 Compressor (CMOS2)

3.7.2 Transmission Gate and Pass-Logic 4:2 Compressors

Another circuit implemented in the form of Figure 3.14(a) is shown in Figure 3.16. It uses two 16-transistor transmission-logic based full-adders (TGATE) [24]. This circuit was studied due to its low transistor count and output signal integrity. This study reveals that this circuit has the best operation in terms of power consumption amongst other FA circuits with similar transistor counts.

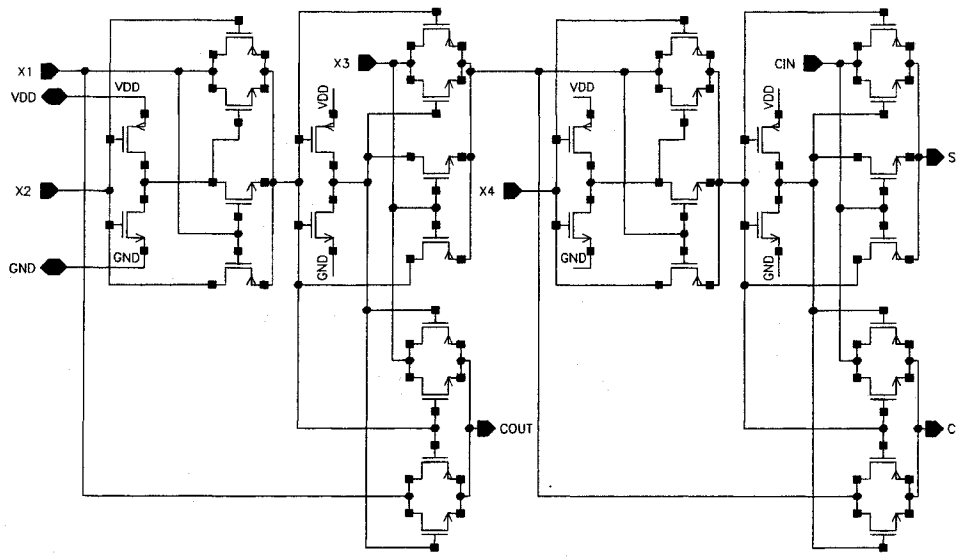


Figure 3.16 Transmission Gate 4:2 Compressor (TGATE)

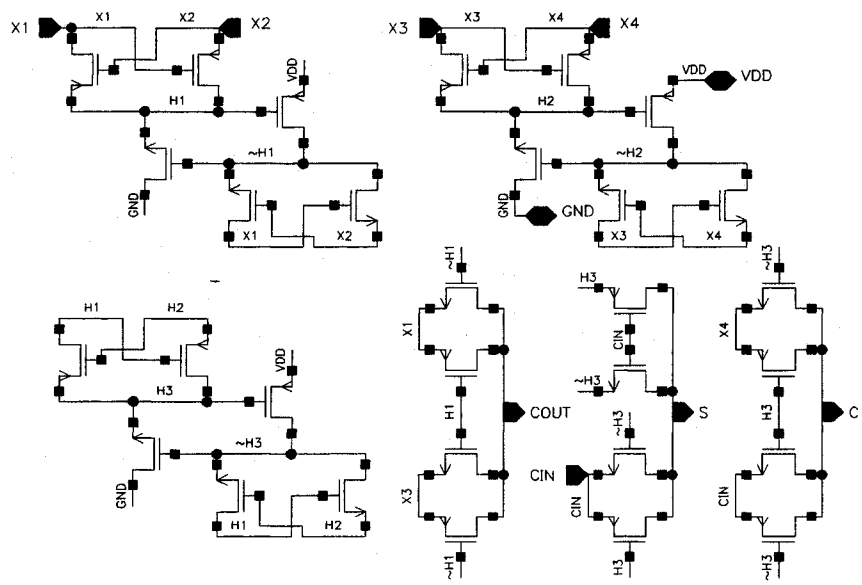


Figure 3.17 30-Transistor Pass-Logic (PASS1)

Three pass-transistor-logic implementations of the 4:2 compressor were examined. The first two of these circuits being extremely minimized in terms of transistor count. The first pass-logic 4:2 compressor to be investigated (PASS1) pictured in Figure 3.17 is a 30-transistor model using novel transmission style output circuitry [12]. This output circuitry forms the basis for the other two 4:2 compressor designs.

The second pass-logic compressor studied (PASS2), Figure 3.18, utilizes the same output circuitry as that of PASS1 but uses a different configuration for the XOR/XNOR cells from [29]. Several different pass-logic XOR/XNOR gates having 6-transistors or less were tested in this circuit before declaring this particular configuration to be the most ideal. These cells provide full-swing XOR output but may experience voltage threshold gain on the XNOR output. Fortunately, by way of the output circuitry configuration, there is little apparent effect on the output of the compressor.

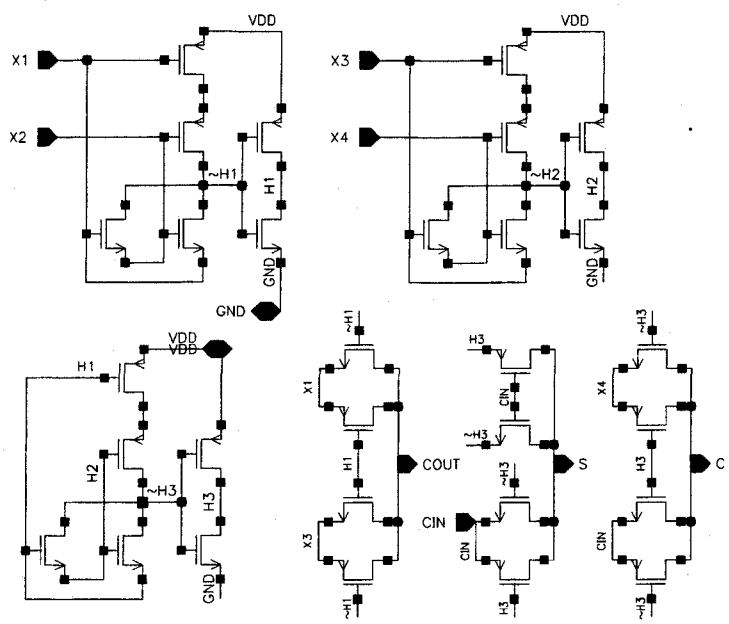


Figure 3.18 Proposed 30-Transistor Pass-Logic (PASS2)

The final pass-logic configuration of the 4:2 compressor (PASS3) presented in Figure 3.19 uses the same XOR gates as PASS2 but extra XNOR gates are added with full-swing output voltage. This was done to ensure that the non-full-swing XNOR signals internal to the PASS2 compressor were not adversely affecting the performance of the circuit.

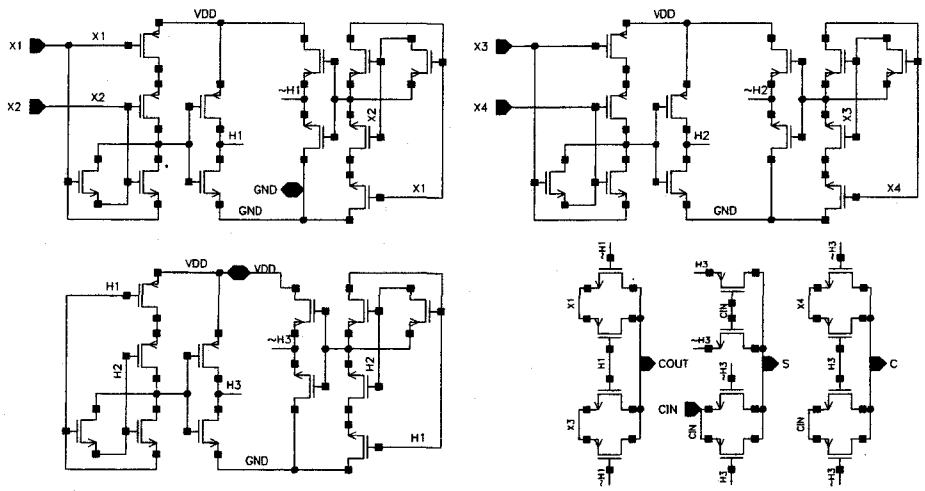


Figure 3.19 Proposed 38-Transistor Pass-Logic (PASS3)

Other pass-logic designs for the 4:2 compressor are certainly possible and several other configurations were in fact tested. However, many of them either experienced non-full-swing outputs, enormous short circuit conditions or just plain poor quality outputs. Since in partial product reduction, the outputs of one compressor cell most likely provide inputs to the next stage of compressors, proper output signal integrity is of paramount importance. A design exemplifying these shortcomings is presented in [32]. This is a very well conceived compressor using novel XOR and MUX gates, however this circuit is a dual-rail design and for the reasons previously stated with regards to such designs, it has been omitted from this study.

3.7.3 Proposed Hybrid CMOS 4:2 Compressors

The final two compressors studied are hybrid designs combining standard CMOS with transmission and pass-logic. The first of these designs (HYBRID1) uses the XOR/MUX configuration of the 4:2 compressor. This design is similar to the CMOS2, the difference being that the output CMOS MUX are replaced by transmission-gate style multiplexers shown in Figure 3.20. This substantially cut down on the number of transistors in the circuit. Furthermore, the full-swing nature of transmission gates combined with the fact that the outputs will only ever be passed though one transmission gate, this circuit still maintains similar signal integrity to the standard CMOS compressor.

The second hybrid design presented (HYBRID2) shown in Figure 3.21 utilizes the output circuitry used in both PASS1 and PASS2 cells. In doing so it reduces the size of the compressor by 6 transistors when compared to HYBRID1, though it does come at the cost of passing the input C_{IN} to the output S, gated by only one transistor. However, this weak output is met by a standard CMOS MUX cell at the input of the next compressor stage, and so the circuit will not face the level of signal degradation or reduced driving capabilities associated with many other pass-logic circuits.

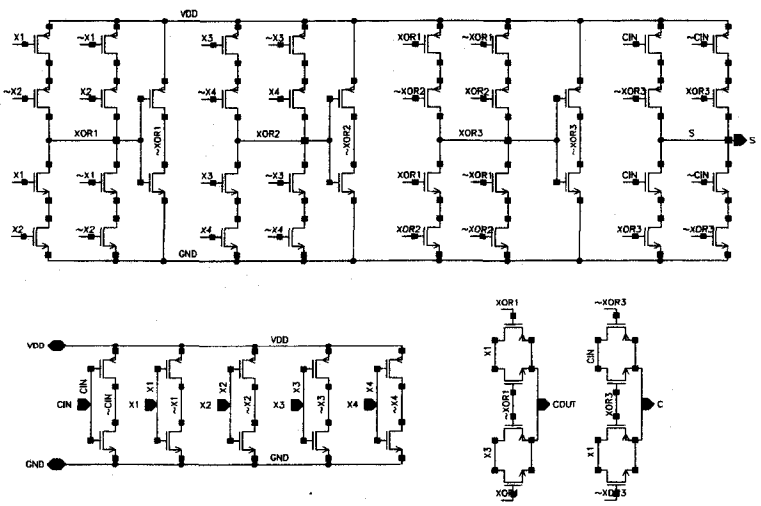


Figure 3.20 Standard CMOS XOR with Transmission-Gate MUX (HYBRID1)

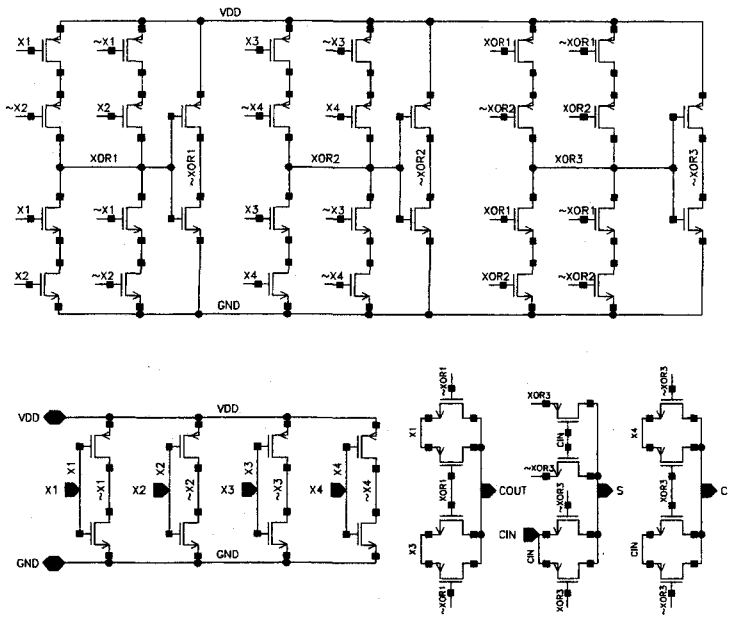


Figure 3.21 Standard CMOS XOR with Pass-Transmission Outputs (HYBRID2)

3.8 Simulation

3.8.1 Transistor Sizing

When dealing with pass-logic circuits, transistor sizing is a very important aspect of design in order to obtain properly performing circuits and reliable simulation results. In this study, a method very similar to the one proposed in [12] is used. It involves determining the critical path of the circuit in question and experimentally sizing the transistors along that path. Once a performance plateau is reached, attention moves to the next most critical path for sizing. This iterative process continues until all transistors have been properly sized.

3.8.2 Power and Delay Measurement

Power Measurement

Total power dissipation in an integrated circuit is equal to the summation of dynamic, short-circuit leakage power described by equation (*3.1*). Dynamic power ($P_{dynamic}$) is the power needed to charge and discharge internal capacitances in the logic array and interconnect networks when switching a circuit from one state to the next (e.g., from a logic 0 to logic 1). Short circuit power (P_{short}) is the result of conditions within the circuit sometimes referred to as "***glitch***" conditions. These conditions occur when an output switches states and for a brief, almost instantaneous period of time, a path to both the source voltage and ground are present. The final element in total power for a circuit, leakage power ($P_{leakage}$) is a product.

$$P_{total} = P_{dynamic} + P_{short} + P_{leakage} \qquad (3.1)$$

Measuring the power dissipation of a standard CMOS cell is quite straight forward; namely to multiply the current drawn by the circuit from the power source by the supply voltage. In the case of pass-transistor logic, this job becomes somewhat more complicated since the inputs are not terminating directly within the cell, but instead they may be passed through several transistors and then to the output. This means that all inputs must be treated as sources. The current and voltage must then be measured at every input and power subsequently calculated. This also means that each input may be driving a secondary circuit at the output. In this case, simply measuring the current drawn by the circuit is not acceptable; the current drawn at the output of the circuit must also be determined. Equations (3.2) and (3.3) present an analytical interpretation of this issue. By measuring the power into the circuit and subtracting the power out, the actual power used by the compressor is determined apart from that of the surrounding circuits.

$$P = P_{in} - P_{out} = V_{in}i_{in} - V_{out}i_{out} \qquad (3.2)$$

$$V_{in}i_{in} = V_{dd}i_{dd} + V_{X1}i_{X1} + V_{X2}i_{X2} + V_{X3}i_{X3} + V_{X4}i_{X4} + V_{Cin}i_{Cin} \qquad (3.3)$$

Delay

Depicted in Figure 3.22, the delay of the circuits is defined in this study as the time it takes for the output signal to reach 90% of its full potential measured from the point at which the inputs reach 10% of their maximum swing voltage. This is measured for every transition over every possible input combination.

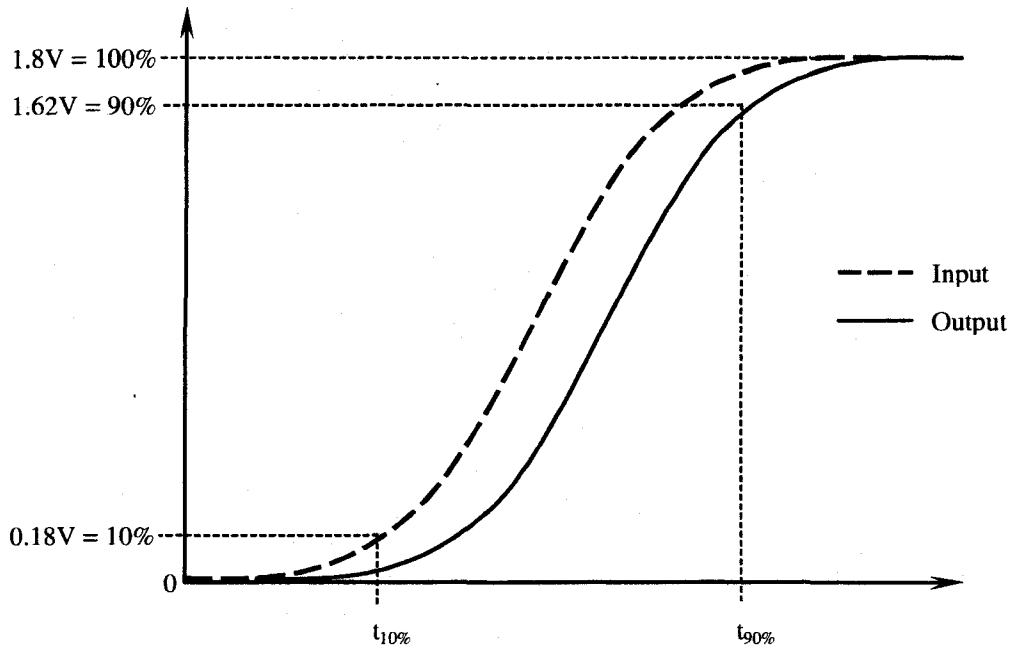


Figure 3.22 Delay Measurement

3.8.3 Test Bench and Conditions

In order to gain reliable results, much care must be given in the design of the testbench shown in Figure 3.23. The testbench used in this study uses input buffers after each ideal input waveform in order to replicate the non-ideal inputs expected in a true application. Furthermore, buffers are used at the output of cells to simulate output load conditions found in real world applications where these cells would be cascaded through several stages of reduction. In order to separate the performance characteristics of the compressor under study from that of the testbench, separate dedicated voltage sources are provided for

both the test circuitry and the compressor. The input pattern in Figure 3.24 covers all possible input vector combinations at a frequency of 250 MHz and 1.8v source voltage. Performance of the circuit during each of these transitions was manually measured and calculated in order to validate the functionality of the circuit, the reliability of the results, and the conclusions of this study.

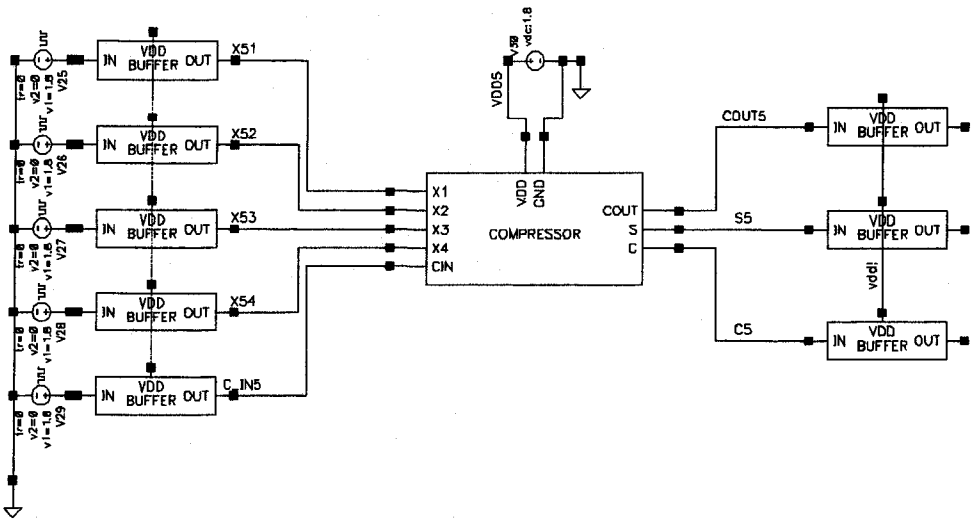


Figure 3.23 4:2 Compressor Test Bench Schematic

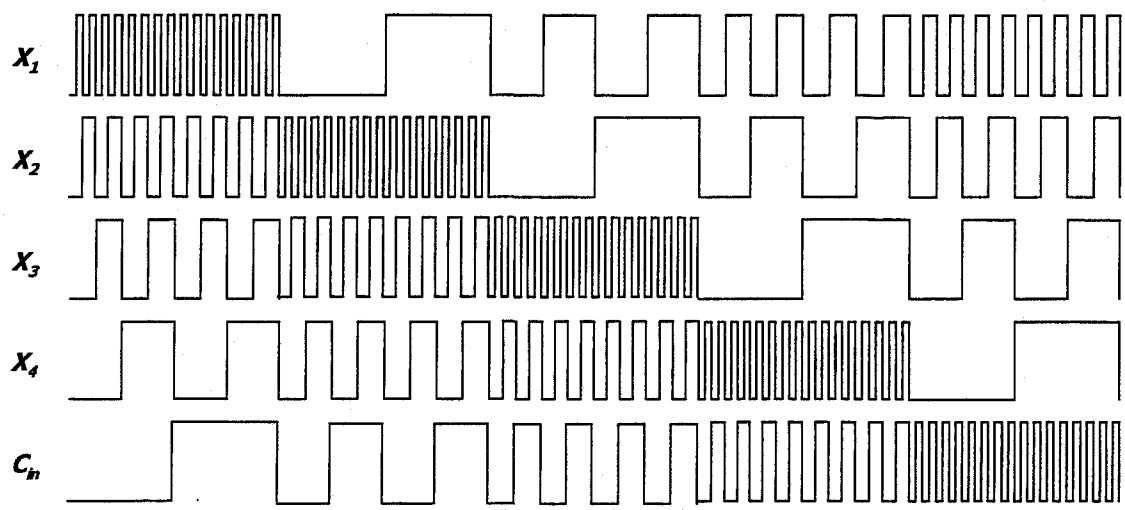


Figure 3.24 Input Test Vectors

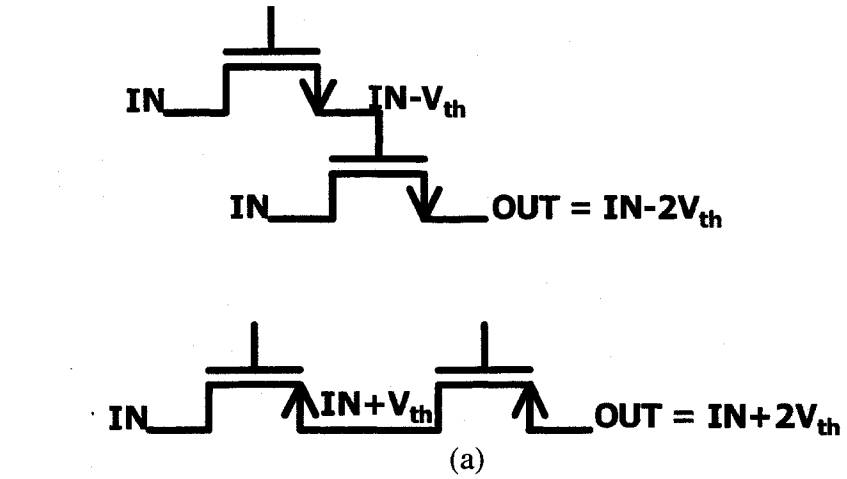
3.8.4 Full-Adder Simulation

Prior to studying 4:2 compressor cells, simulations were performed on several existing full-adder designs to evaluate the function of many of the design aspects of these cells. Due to their many similarities, once the operation of these circuits were validated, certain aspects of their designs could be cascaded into the design of the 4:2 compressors. Namely the XOR/XNOR circuitry used in the 14-transistor pass-logic #2 circuit was used in the new PASS2 4:2 compressor. The simulations focussed on power consumption and on basic functionality rather than speed. Table 3.3 summarizes the results in terms of power consumption. The percent savings referred to in this table are with respect to the standard CMOS full-adder.

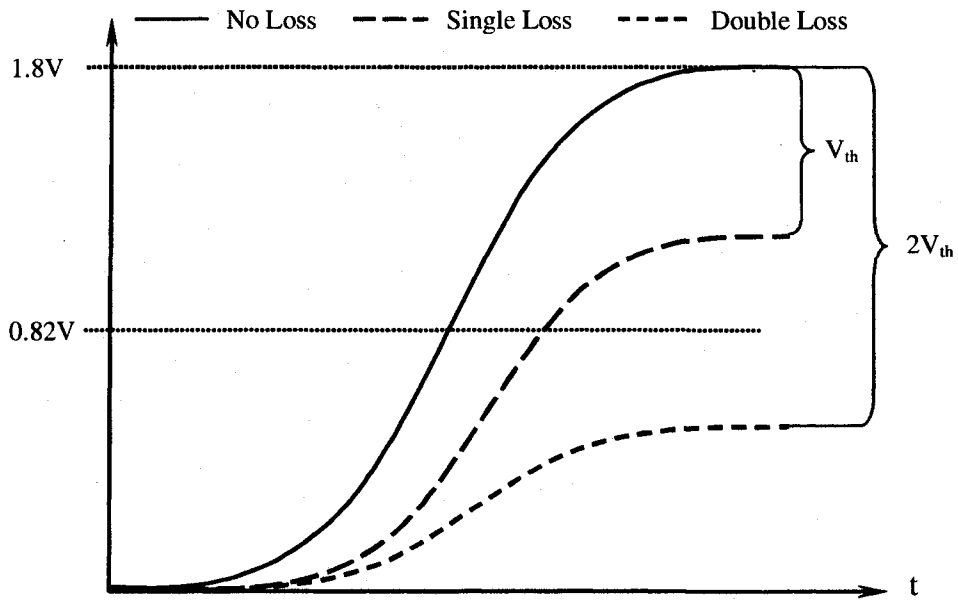
Table 3.3Simulation Results for Full-Adders

Cell name	Pwer Dissipation	% Savings
Standard CMOS	7.974 uW	N/A
TG-Logic	4.693 uW	41.15%
14-Transistor Pass-Logic #1	5.052 uW	36.64%
14-Transistor Pass-Logic #2	4.732 uW	40.66%
10-Transistor Pass-Logic	3.651 uW	54.21%

These results show that the 10-transistor pass-logic full-adder has the lowest power consumption of all studied. This does not mean it is the best design. In fact this design will not operate properly when implemented in a larger design. This is due to voltage threshold loss. As shown in Figure 3.25(a), when a signal is passed through two nFETs in series or two cascaded pFETs, the voltage threshold loss or gain occurs twice respectively. The effect on the output signal is shown in Figure 3.25(b) for a "high" signal passed through two nFETs in series. It is easy to see in this figure how after double threshold voltage loss, the output would be perceived as a false "low". In this ten transistor full-adder circuit and in all others studied, this is inevitable. For this reason, the 10-transistor pass-logic full-adder circuits are not suitable for any implementation in this study and will not be further examined.



(a)



(b)

Figure 3.25 Voltage Threshold Loss / Gain
(a) Signal Path and Effect (b) Output Waveform for nFET Case

3.8.5 4:2 Compressor Simulation Results

The conventional metrics that have been used in comparing the compressor cells are power, area, and delay. ***Power delay product (PDP)***A is the metric considered by many to be the best for gauging performance of low power circuitry. The downfall of this measurement in the past has been the inconsistency with which power dissipation is

calculated. This is especially the case when dealing with low power architectures employing pass transistor logic. This survey offers an unpartisan approach to power measurement, and an accurate reflection of the vantage points of each logic style. Moreover, with a growing interest in the applications of asynchronous circuitry, average delay, in addition to worst case delay, has been considered.

Table 3.4 summarizes the performance of the 4:2 compressors under study. The shaded regions signify "top-performance" for each individual metric. These results are depicted graphically in Figures 3.26, 3.27 and 3.28 for further ease of comparison. Output waveforms for all 4:2 compressor circuits are provided in Appendix B.

Table 3.4Simulation Results For 4:2 Compressors

Cell Name	Power Dissipation	Average Delay	Worst Case Delay
CMOS1	7.15E-05	0.27	0.64
CMOS2	5.27E-05	0.43	0.98
TGATE	1.73E-05	0.41	1.81
PASS1	3.91E-05	0.52	1.7
PASS2	1.74E-05	0.29	0.7
PASS3	5.83E-05	0.46	1.59
HYBRID1	4.24E-05	0.48	1.04
HYBRID2	2.51E-05	0.44	1.25
	Average PDP	**Worst Case PDP**	**Transistor Count**
CMOS1	1.96E-05	3.86E-05	56
CMOS2	2.29E-05	5.14E-05	68
TGATE	7.09E-06	3.13E-05	32
PASS1	2.02E-05	6.65E-05	30
PASS2	5.09E-06	1.22E-05	30
PASS3	2.69E-05	9.28E-05	48
HYBRID1	2.04E-05	4.41E-05	56
HYBRID2	1.09E-05	3.13E-05	1

While none of the 4:2 compressor designs are able to match the standard CMOS compressor in terms of speed, both the transmission gate 4:2 compressor (TGATE) and the new pass-logic 4:2 compressor (PASS2) exhibit tremendous gains in terms of power dissipation and circuit size. Due to it's extremely close power performance to the TGATE compressor, PASS2 is also considered a winner in this aspect. Furthermore, where power delay product is concerned, PASS2 is the clear winner.

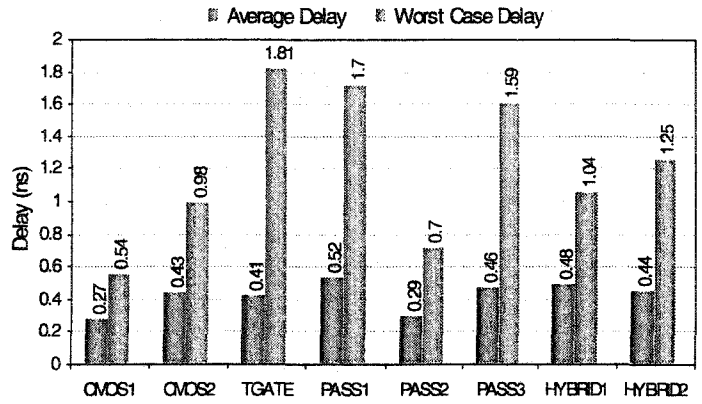


Figure 3.26 Average and Worst Case Delay

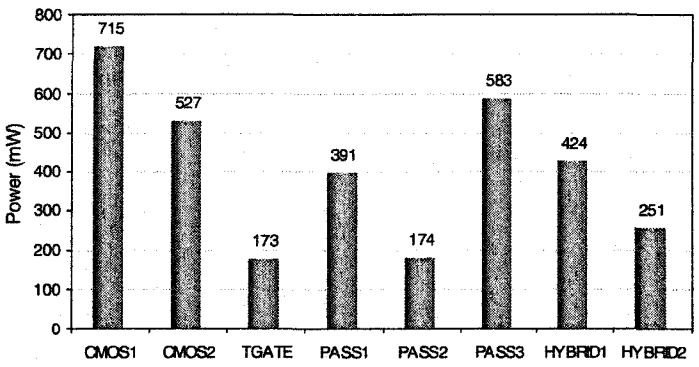


Figure 3.27 Power Dissipation

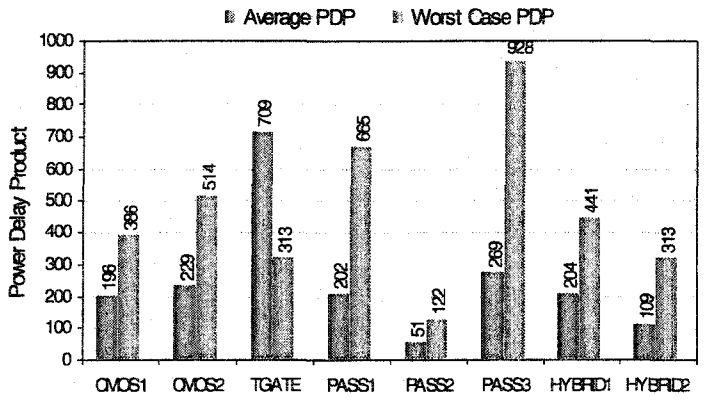


Figure 3.28 Average and Worst Case Power-Delay-Product

Chapter 4

Multiplier Design and Test

Through study and test the best performing full-adders and 4:2 compressors have been determined, these cells may now be proven for their intended use in a complete multiplier design. The 16-bit parallel multiplier studied here is a fairly standard design. It includes a partial product generation array, partial product reduction tree, and a final fast adder. The initial generation of the partial products uses regular CMOS *AND* [33] gates. A standard carry-look-ahead adder [3] is used to sum the two partial products into a final product. The main discriminating factor in this design as opposed to traditional parallel multipliers is the use of newly devised low-power 4:2 compressor cells, low-power full-adders as well as a novel partial product reduction layout methodology [2] which has not been realised at the hardware level until this time. This chapter deals with the steps taken in the design and test of this multiplier.

4.1 Basic Design of Multiplier

The multiplier designed in this study contains all the major aspects described for parallel multipliers in Chapter 2. The difference is in the use of 4:2 compressor cells over (3,2) counters (FA). As discussed above, this design implements a 4:2 reduction scheme

which has never before been realized at the hardware level. The 4:2 compressor used will be PASS2 which showed the greatest overall performance with regards to the findings presented in Chapter 3. Furthermore the full-adder circuit to be implemented in this design is the 14-transistor pass-logic #2 which was also presented in Chapter 3. This full-adder design did not have the lowest power consumption but it's signal integrity was superior to that of all other low power full-adders. Furthermore, design elements used in the 14-transistor pass-logic #2 circuit paralleled those of the 4:2 compressor chosen for implementation.

4.1.1 Partial Product Reduction Scheme

Following the 4:2 compressor layout scheme set forth in Chapter 2, a reduction framework is obtained and depicted in Table 4.1. This design requires a total of ninety-eight 4:2 compressors, seven half-adders, and seven full-adders. Determining the layout for these compressors is actually a much simpler task than what the equations in the methodology might let on. Once the first row of each stage is determined, all consecutive rows follow in a symmetric pattern, reducing in row length by four columns. Figure 4.1 better illustrates the regularity of this scheme.

Table 4.1 Compressor Row Layout Plan

STAGE	ROW	START COLUMN	END COLUMN
1	1	8	23
	2	10	21
	3	12	19
	4	14	17
2	1	4	27
	2	6	25
3	1	2	29

With the reduction tree defined, the rest of the design elements for the multiplier are somewhat trivial. Referring to Chapter 2 for the elements of a parallel multiplier, a 16x16 array of AND gates is required to generate the partial products and a carry-lookahead-adder is needed to sum the last two remaining partial product rows to obtain the final product of the multiplication.

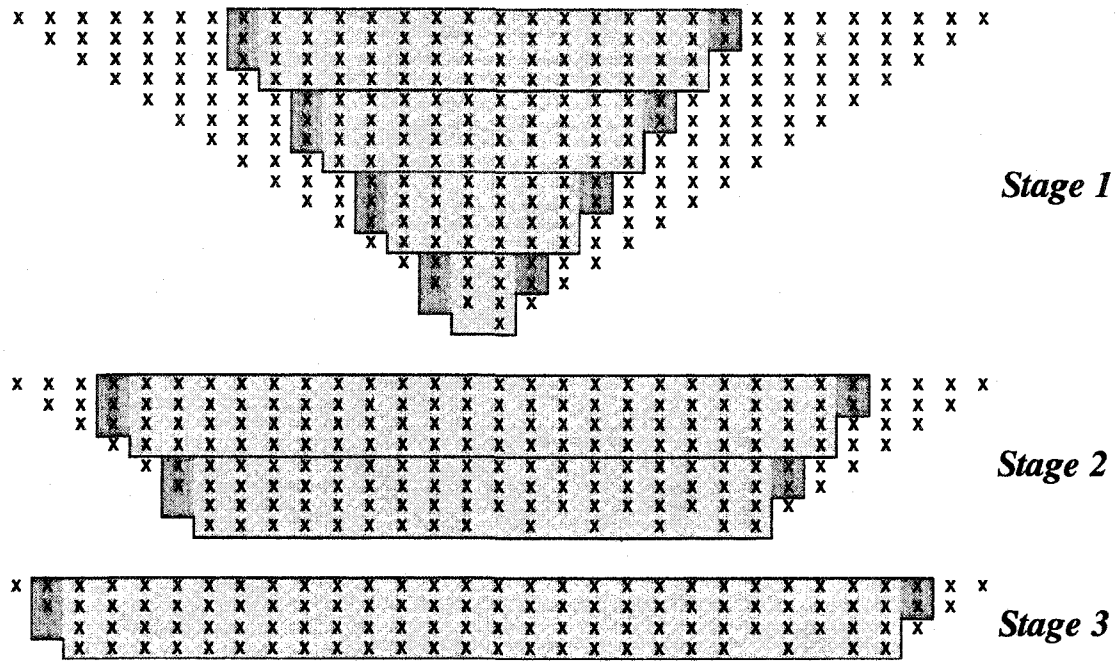


Figure 4.1 Reduction Layout Scheme

4.2 Schematic Generation

Once the basics for the design are set, the next step in any large design is to generate ***hardware descriptor language (HDL)*** code from which the design may be initially tested for theoretical soundness and then imported through a synthesis tool to generate a schematic and eventually a fully layed out circuit.

4.2.1 HDL Coding

With the evolution of digital circuits in the 1980's, gate count grew to such enormous proportions that the task of describing such circuits and their interconnection required teams of engineers. As a result, the U.S. Department of Defense's VHSIC (Very High Speed Integrated Circuit) program funded a project including a team of engineers from three companies - IBM, Texas Instruments, and Intermetrics [34]. The purpose of this project was to create a new language-based design description method. This led to the inception of VHDL (VHSIC Hardware Descriptor Language).

Verilog HDL, the other of the two major HDL's, was introduced in 1985 by Gateway Design Systems Corporation which is now a part of Cadence Design Systems Inc.'s Systems Division. With the formation of Open Verilog International (OVI) in May 1990, Verilog HDL was made available to the Public Domain with the expectation that with a broader acceptance of the language, the market for Verilog related software products would grow. Verilog is the top HDL used by over 10,000 designers at such hardware vendors as Sun Microsystems, Apple Computer and Motorola [35]. Verilog is also the HDL of choice for this study.

4.2.2 Behavioural Coding of Multiplier

Verilog HDL allows the user to use a variety of ***behavioural constructs*** enabling the designer to perform high-level simulations to verify the functionality of the circuit before delving into the more low-level design phases. The following is the simplest example of a 16-bit multiplier coded with behavioural verilog.

```
module MULTIPLIER(OUT,A,B);  // Declares module MULTIPLER

  input [15,0] A,B;        // Defines 16-bit inputs A and B
  output [31,0] OUT;       // Defines 32-bit output OUT
  assign OUT = A*B;        // Assigns out to be the product of A and B

endmodule                  // Declares the end of the module
```

At this level there is no control over how the multiplication is carried out. It simply takes two numbers and multiplies them all into one module. Theoretically, this code could be inputted into a synthesis tool such as Synopsys which would generate a multiplier automatically. This code is not suitable for this design because control must exist over which and how many cells are used and how they are layed out in order to meet the design plan previously set forth.

4.2.3 Verilog Code

Complete verilog code for the 16-bit multiplier is available in Appendix A. This is large code containing explicit instructions on the connectivity of every single cell within the design. Code for a test bench designed to validate the functionality of the multiplier code prior to schematic capture is included at the end of Appendix A.

4.2.4 Schematic Generation

Once the design has been verified through Verilog simulation, it may be imported through the Cadence schematic tool to automatically generate the required cells and interconnection structure. The AND gate array and partial product reduction circuitry are grouped into one cell where the final fast adder is kept separate. This is done to facilitate the data collection with regards to latency of the partial products rather than just the multiplier a a whole. This is important because when measuring the delay of the entire multiplier, a great deal of this delay will be attributed to the fast-adder itself. In fact, because of the nature of fast-addition or any addition for that matter, the most significant digit will always be the one with the greatest latency.

Figure 4.2 shows the schematic generated including AND gates and the 4:2 compressor forming the reduction tree. It is very hard at this level to recognize the reduction scheme because of the way the importation tools seemingly randomly place the cells. The complexity of the circuit a multiplier of even this relatively small size is very apparent in this figure. The fast adder used in this design is a standard 2-stage 32-bit carry-lookahead-adder pictured in Figure 4.3. The complexity of these circuits is still not completely represented by these already highly complex schematics. Each of these schematics contain cells internal to which is a whole other set of circuitry.

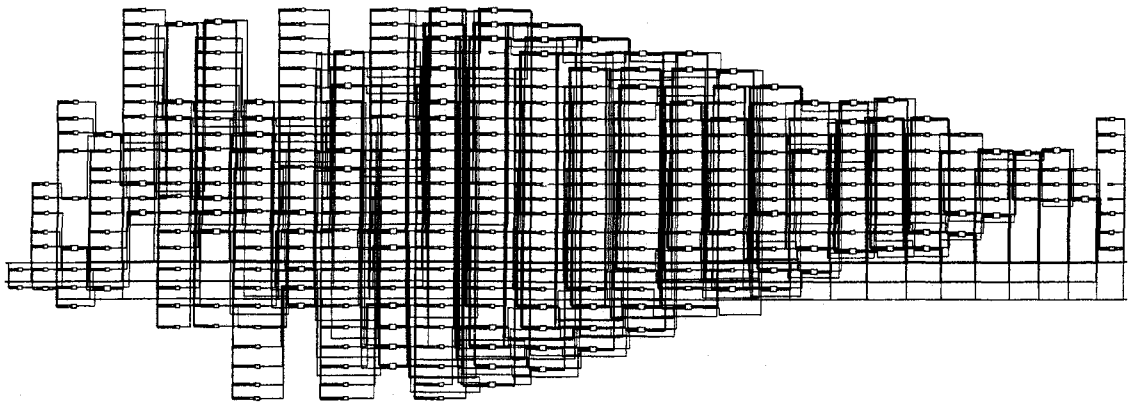


Figure 4.2 Schematic Capture of 16-bit Partial Product Generator and Reduction Tree

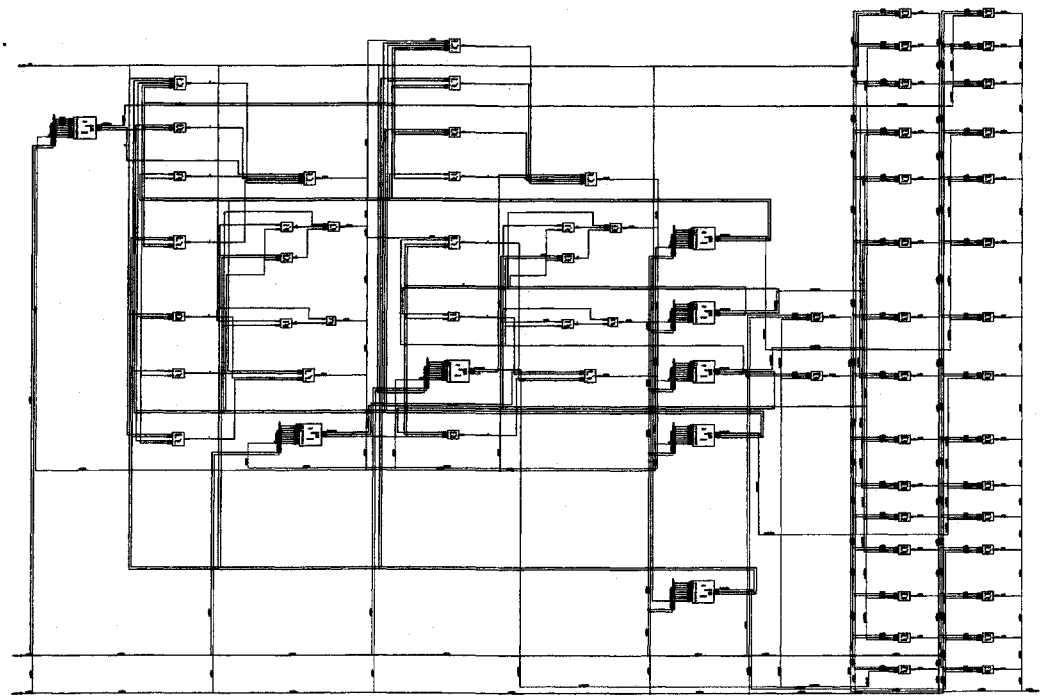


Figure 4.3 Schematic Capture of 32-bit CLA

4.3 Circuit Layout

4.3.1 Why Custom Layout?

Traditionally, when any digital circuit is to be implemented at the layout level, a design compiler such as Synopsys is used to generate a netlist from behavioural verilog code which is then layed out using an automated tool such as AreaPDP. When requested to produce a 16-bit multiplier, Synopsys will use available pre-defined cells included in our existing design library to create a netlist defining what cells are needed to complete the physical structure and how they are to be connected. This netlist is then inputted to a floorplanning tool such as Qplace which, according to user defined constraints such as area and speed, will perform the layout and routing of the multiplier at the chip level. Figure 4.4 provides an overview of the complete automated chip design process. Most all steps in this process are essentially bypassed when creating a custom layout design.

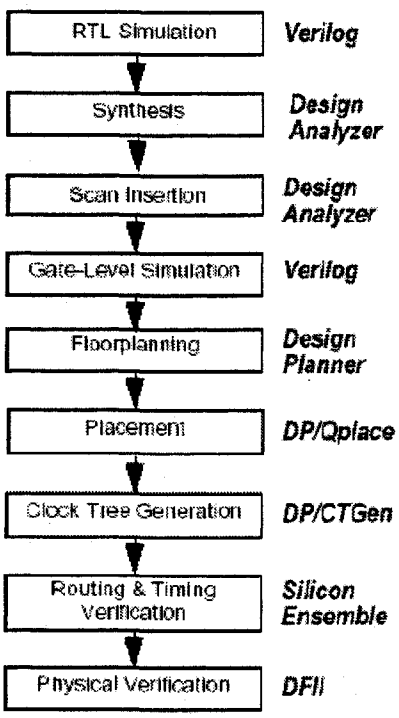


Figure 4.4 Cadence Digital IC Design Flow

In most cases automated layout is the best method for efficient design of digital circuits. It can cut design time down from months, as is the case in custom layout, to just days depending on the complexity of the circuit. For this study there are reasons why automated layout is unfortunately not an option. First of all, there are no cells available in the standard libraries for 4:2 compressors. Even if there were, this is a completely new compressor design being used. It is possible in the coding stages to define a 4:2 compressor such that it is made up of a combination of existing standard cells such as two full-adders or a combination of XOR/XNOR and MUX gates. This would require that the interconnections between all cells be defined subsequently sacrificing the time advantage associated with Synopsys automatically defining these relationships. Secondly, the standard cells being used to generate the compressors are just that, standard. As previously shown, the most efficient design of the 4:2 compressor occurs when the compressor is designed at the transistor level. When standard cells are used, transistor level optimization is not possible.

Basically, when the automated tools are used, the designer maintains control over general functionality of the chip but loses much control over the actual circuitry and layout. Because this multiplier will implement non-standard, newly devised circuitry in a new minimized configuration, designer control is very important, therefore automated layout is not an option and full custom layout must be performed.

4.3.2 Layout Basics

When dealing with digital integrated circuits, the most basic, abundant and most important element in any chip is the transistor. Figure 4.5 presents fully labelled schematic and layout views of a CMOS inverter with all elements labelled for comparison. This circuit is ideal for presenting the basic elements of layout because of it's simplicity and abundance in most designs.

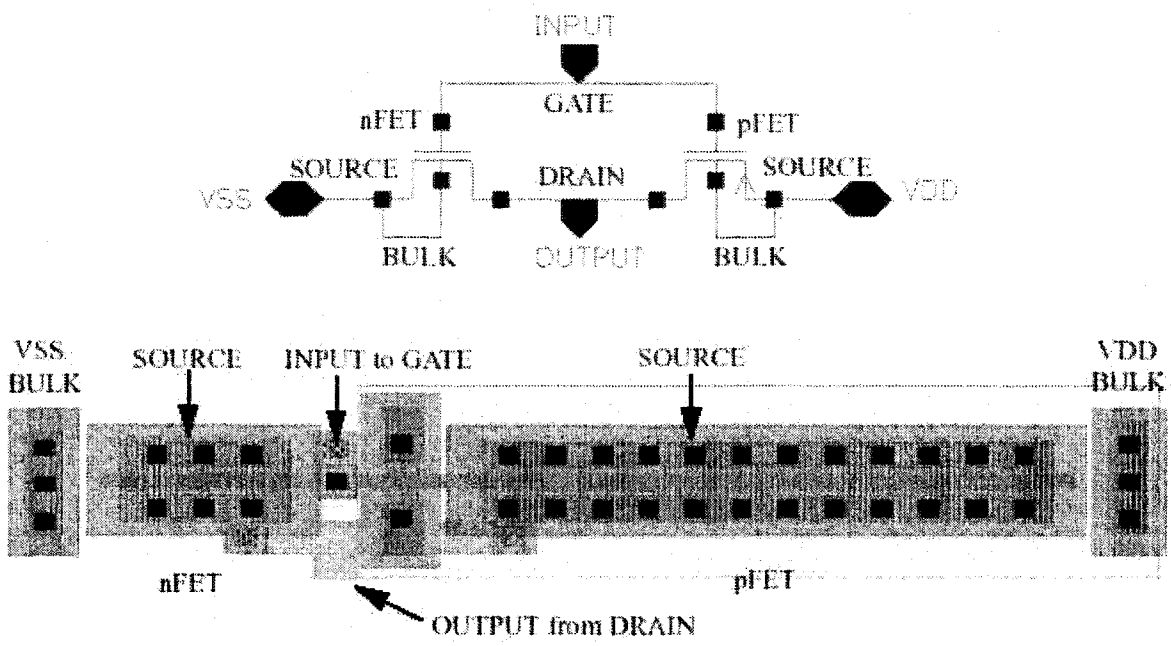


Figure 4.5 Schematic and Layout Components of a CMOS Inverter

4.3.3 Compressor Layout

Much care was taken in the layout of all the arithmetic cells used in this design in order minimize the congestion and length of interconnections. Figure 4.6 illustrates the silicon level layout of the 4:2 compressor cell. Notice that all inputs with exception of C_{in} are accessible from the left side of the compressor and all outputs are accessed from the right side with the exception of C_{out}. This is done in order to keep a uniform flow of data throughout the entire multiplier. Input C_{in} and output C_{out} are situated directly across from each other. This facilitates a short direct path for the propagation of carry signals as shown in Figure 4.7.

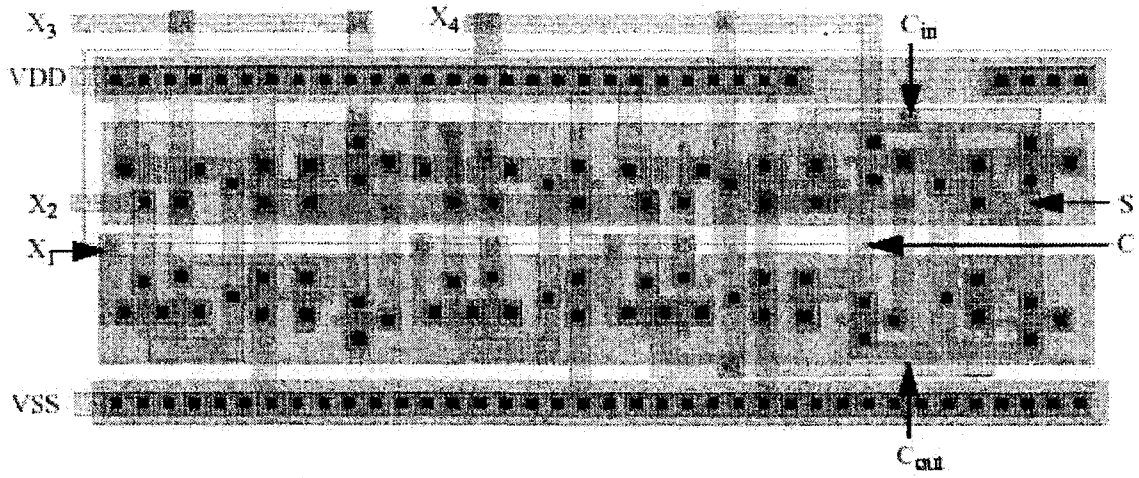


Figure 4.6 4:2 4:2 Compressor Layout

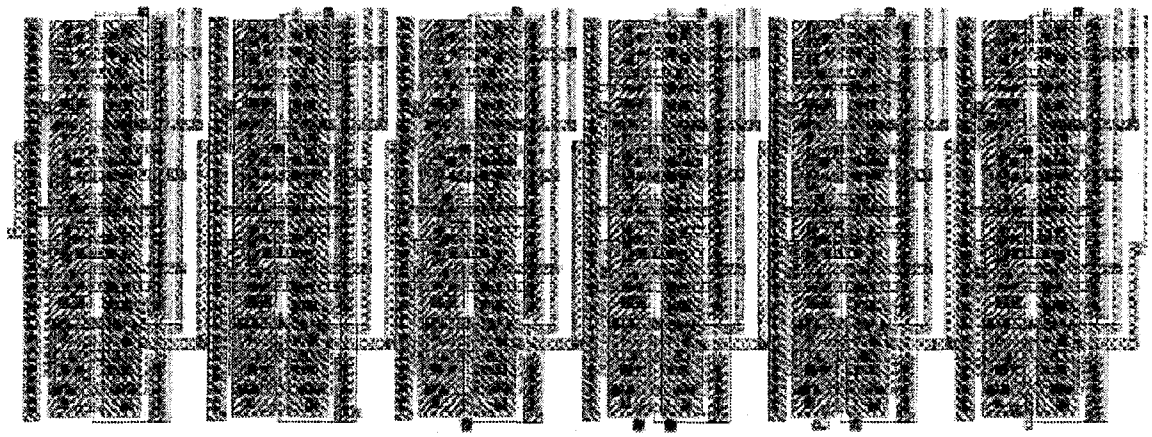


Figure 4.7 Interconnected Compressors

4.3.4 Layout of Partial Product Reduction Rows

As stated in the previous section, great care was taken in the design of the compressor cells to provide as much regularity in the multiplier design as possible. The same care was taken in the layout of all circuits used in this design. This is apparent in Figure 4.8 which depicts a complete row of compressors as well as the *AND* gates supplying initial partial products for reduction. This is in fact the third row in the first stage of the reduction tree

defined in Figure 4.1. Note that the width of all cells are equivalent providing direct interconnection of both source voltages and grounds.

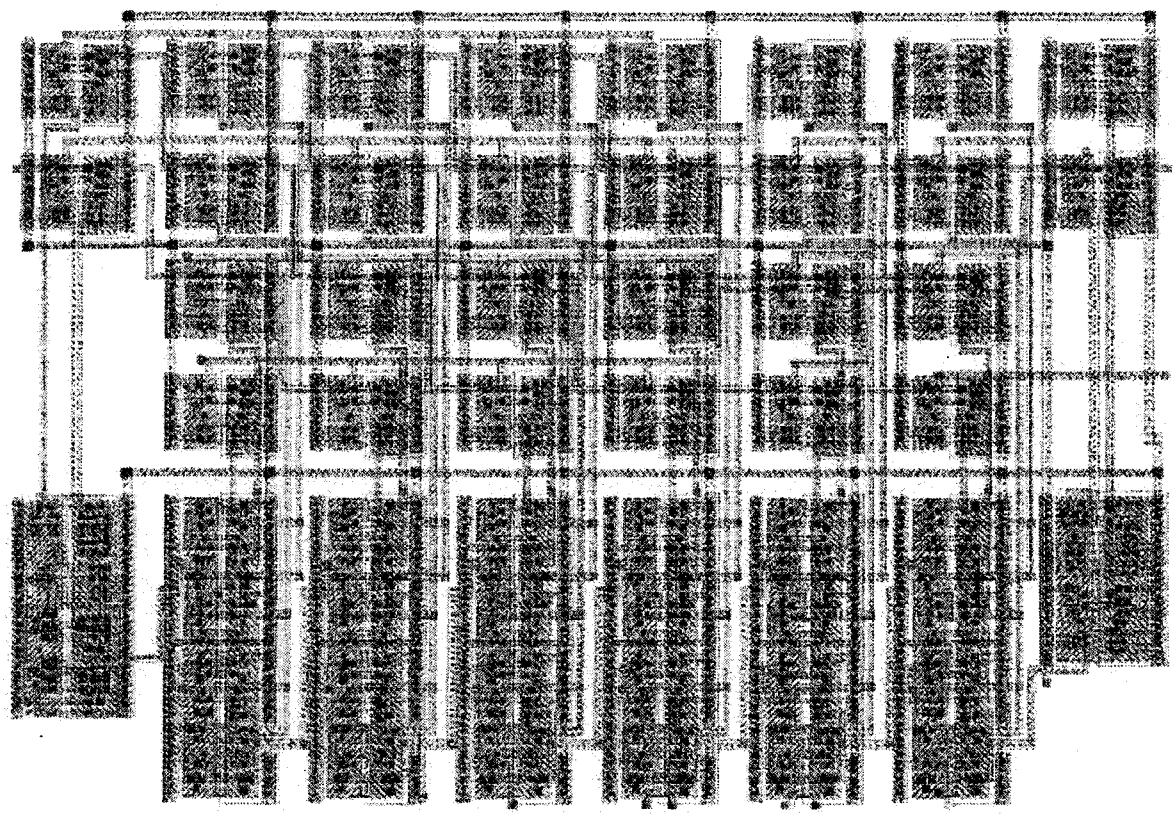


Figure 4.8 Complete Compressor Row Including Partial Product Generation

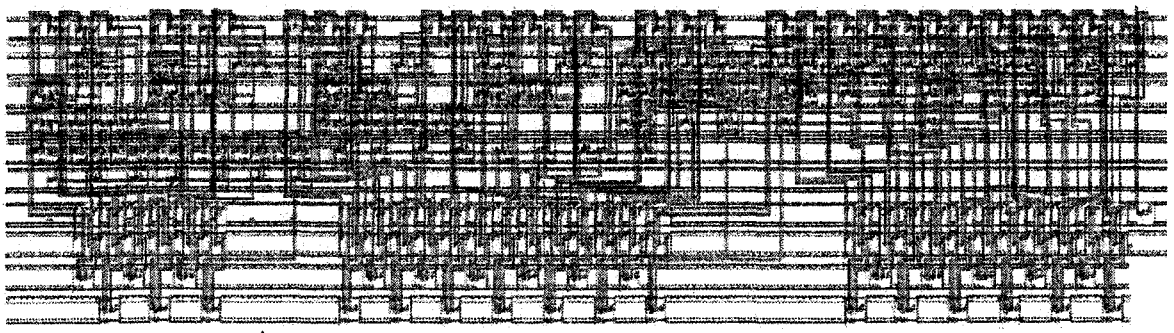


Figure 4.9 31-bit CLA Layout

4.3.5 Final Fast Adder

As previously discussed, the fast-adder used in this design is a 2-stage 31-bit carry-lookahead-adder. Construction of this adder utilizes standard CMOS cells provided with the Cadence design kit cell library. The layout of these standard cells can be quite difficult when done in a manual or custom manner because they are completely *black box*. This means that the designer can only view the connection points a bounding box inside of which the actual circuitry is hidden. This leads to a very tedious design process of creating connections and running design rule checks (DRCs) to verify that these connections do not overlap or come too close to these hidden components within the box. Figure 4.9 shows the completed layout of this fast-adder.

4.4 Design for Fabrication

Due to the high demand for silicon realestate, this design was subject to tight area constraints. The largest elements on this chip are the I/O cells and pads. For a 16x16-bit parallel multiplier, a minimum of thirty-two inputs and thirty-two outputs are required for proper parallel operation. On top of these many I/Os, a chip this size requires multiple power and ground pads to supply both ring and core voltages within the chip. Ring voltage refers to the voltage required by the I/O circuitry and the core voltage provides the actual multipliers's operating voltage. Providing at least two sources for each of these voltage sources and two grounds, the total required pin count to fabricate this design adds up to seventy total pins. Unfortunately the total space allotted to this design for it's assigned fabrication run was 1.2 x 1.2 mm. This space only allows for a maximum total of twenty-four I/O pins.

4.4.1 Input Design

In order to make due with the limited total number of pins available and still provide parallel inputs to the multiplier circuit, the inputting of the operands had to be carried out in a non-ideal fashion. To do this, two 16-bit serial to parallel converters are used. Figure 4.10 depicts a 4-bit serial to parallel converter with arrows marking the data paths. These

converters consist of a D flip-flop shift register with outputs from each flip-flop feeding both the next flip-flop in the register as well as d flip-flop latches. Once the register is full, the advance is set "high" and the D flip-flop latches simultaneously release all values stored in the register in parallel form. This allows for the release of inputs to the multiplier in true parallel manner while reducing the number of input pins to four. Four pins because the multiplier and multiplicand each have their own converter requiring separate serial inputs but common clock and advance signals.

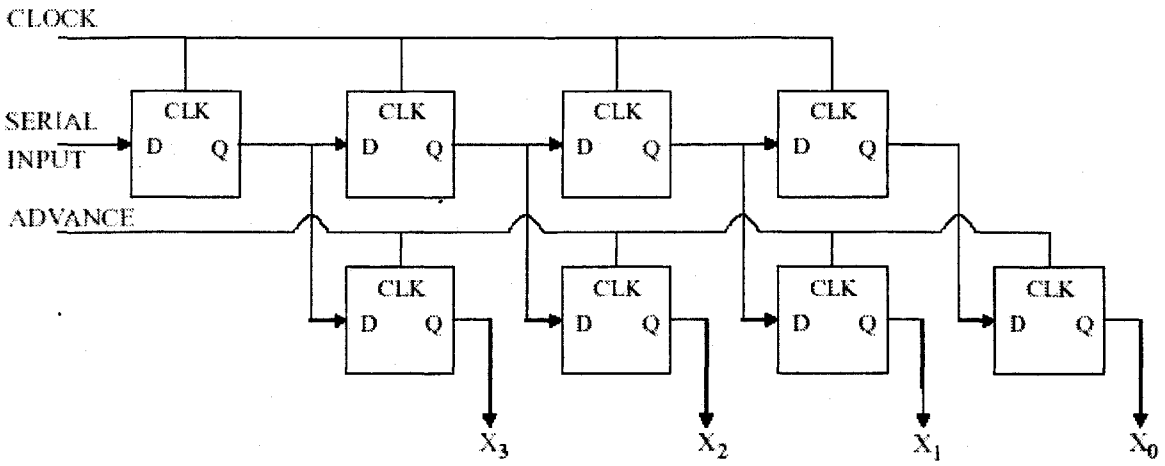


Figure 4.10 4-bit Serial to Parallel Converter

4.4.2 Output Design

Although a serial to parallel converter works out for pin reduction in the input circuitry, parallel to serial is out of the question for the outputs. This being a parallel multiplier, the outputs must be received in parallel in order to maintain some idea of the multiplier's performance. In the case of parallel to serial converters, any signal degradation and definitely all delay experienced at the output could most likely be a product of the output register. What was used instead of such a converter is an output-select design. This circuitry involves the use of eight 4-input multiplexers. This circuitry provides the ability to select which outputs are to received during a certain input test pattern. The way the outputs are divided up in this design allow the tester to select one of four sets of bits as

described in Table 4.2. This now reduces the number of output related pins to 10 including the eight output and two control bits.

Table 4.2Operation of Output-Select Circuitry

S1	S0	Output Bits Selected
0	0	OUT0 to OUT7
0	1	OUT8 to OUT15
1	0	OUT16 to OUT23
1	1	OUT24 to OUT32

4.4.3 Final Multiplier Design and Packaging

The multiplier was fabricated using TSMC 0.18μm technology through a grant from the Canadian Microelectronics Corporation. The packaging chosen for this multiplier was a standard 40-pin DIP shown in Figure 4.11. Figure 4.12 is a photograph of 4 fabricated chips along side a standard mechanical pencil. Table 4.3 provides a list of the 40 pins and the connections to the multiplier associated with them. The completed design for the 16x16-bit multiplier studied here is depicted in Figure 4.13. The darkest area in the center of the chip is the partial product generation array and reduction tree.

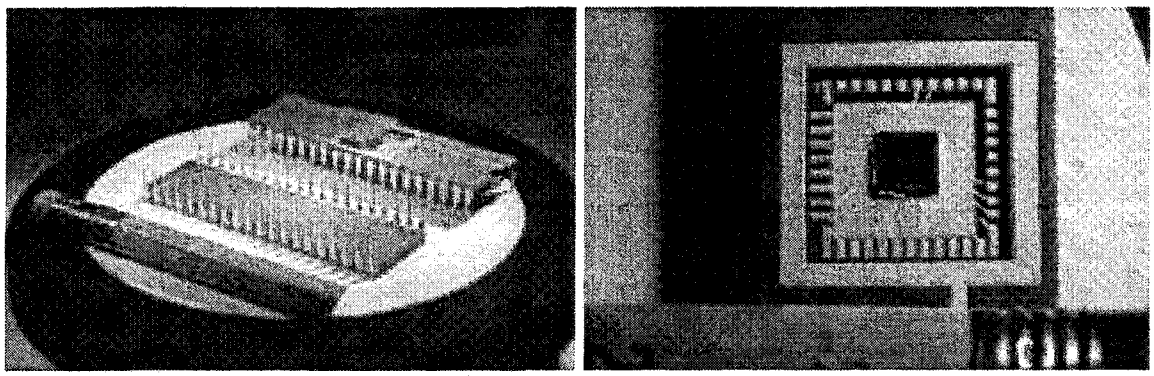


Figure 4.11 Standard 40-Pin DIP Packaging

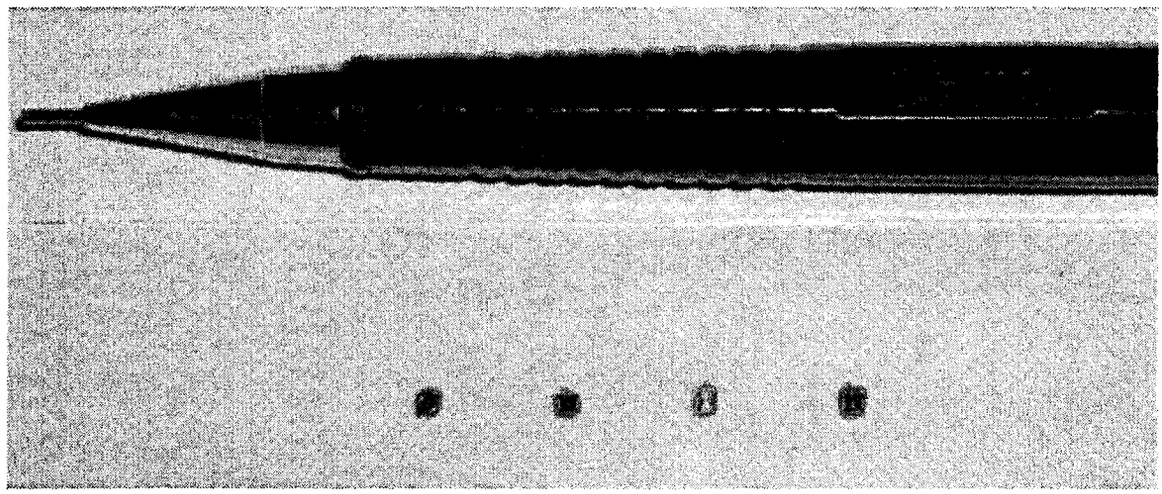


Figure 4.12 Four Fabricated Multipliers and Standard Mechanical Pencil

Table 4.3 I/O Pins and Their Functions

PIN #	I/O TYPE	FUNCTION	PIN #	I/O TYPE	FUNCTION
1	POWER	VSScore	21	POWER	VSScore
2	OUTPUT	OUT7	22	OUTPUT	OUT0
3	OUTPUT	OUT6	23	POWER	ADVANCE
4	SPARE	SPARE	24	SPARE	SPARE
5	SPARE	SPARE	25	SPARE	SPARE
6	SPARE	SPARE	26	SPARE	SPARE
7	SPARE	SPARE	27	SPARE	SPARE
8	OUTPUT	OUT5	28	INPUT	INPUTB
9	OUTPUT	OUT4	29	INPUT	INPUTA
10	POWER	VSScore	30	POWER	VDDring
11	OUTPUT	OUT3	31	POWER	VDDcore
12	OUTPUT	OUT2	32	CLOCK	CLK
13	POWER	VDDring	33	POWER	VSSring
14	SPARE	SPARE	34	SPARE	SPARE
15	SPARE	SPARE	35	SPARE	SPARE
16	SPARE	SPARE	36	SPARE	SPARE
17	SPARE	SPARE	37	SPARE	SPARE
18	OUTPUT	OUT1	38	INPUT	OS1
19	POWER	VSSring	39	INPUT	OS0
20	POWER	VDDcore	40	POWER	VDDcore

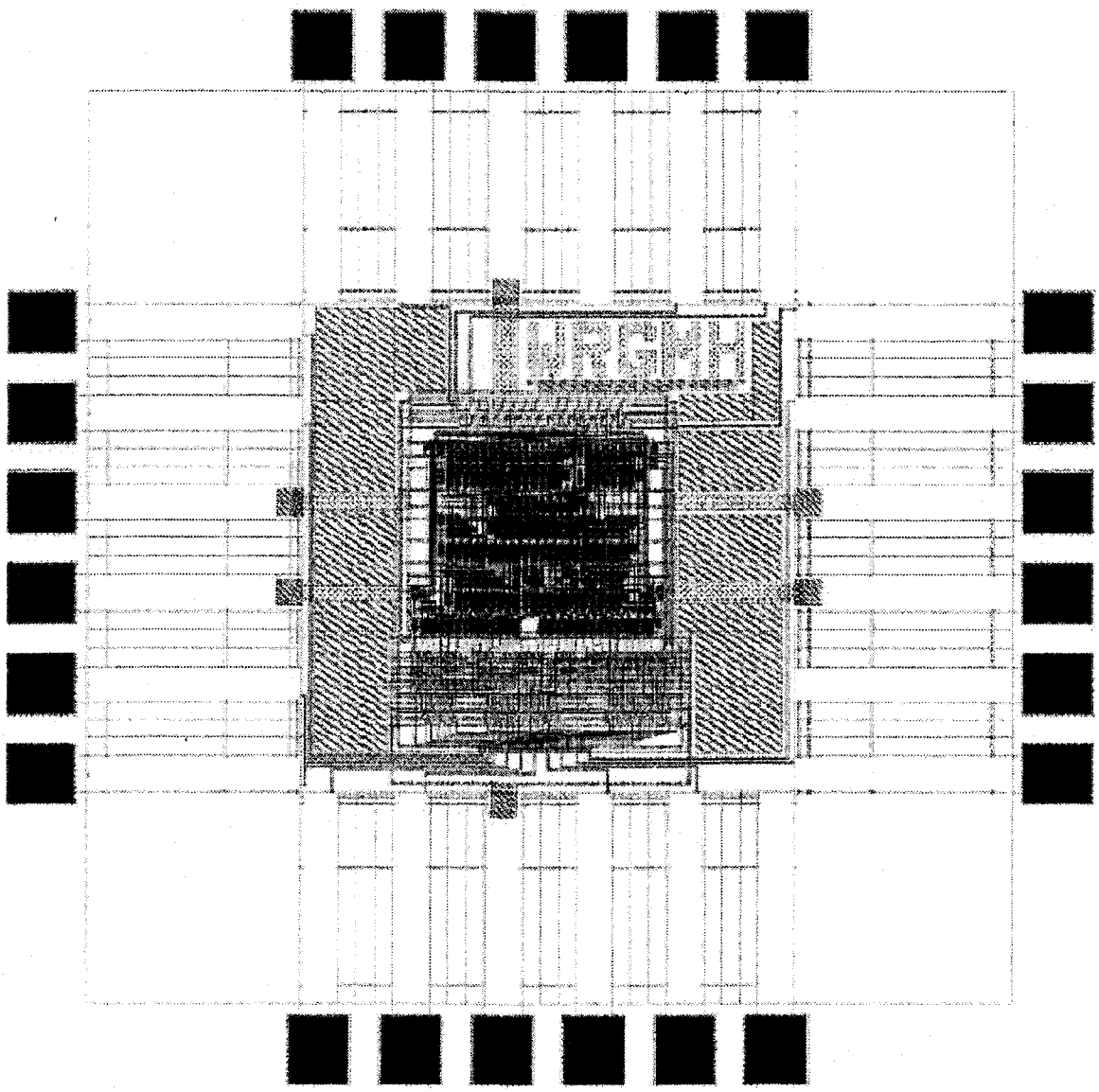


Figure 4.13 Completed Multiplier Design for Fabrication

4.5 Computer Simulation

The complete multiplier design was tested for functionality, power, and speed prior to fabrication. As a benchmark, a second multiplier was also constructed and tested using the standard CMOS compressor cell CMOS1.

4.5.1 Test Bench and Conditions

Figure 4.13 illustrates the structure of the test bench used for both the multiplier using the PASS2 cell as well as the one using the CMOS1 4:2 compressor cell. Inputs are provided by way of sixteen individual waveform generators for both the multiplier and the multiplicand. These inputs feed the partial products generators and reduction tree all combined into another cell. This cell outputs the two 31-bit final partial products, *PPA* and *PPB*, to the 31-bit fast-adder, which in turn outputs the completed sum of the multiplication. By dividing the reduction circuitry and the fast-adder, the delay may be measured both at the completion of partial product reduction before the final addition and at the actual output of the sum. This is important because the large delay associated with fast-addition can cause the results to reflect the performance of the adder rather than that of the reduction. This becomes very important when a comparison of different reduction circuitry is the goal.

As when testing the full-adders and 4:2 compressors, all outputs must be measured manually for delay. Due to the size of the operands, manual measurement is only reasonable for a small number of test vectors. For this reason, a small number of test vectors are chosen which when inputted one after another tend to provide switching activity over the entire reduction tree. Table 4.4 lists these test vectors in the order in which they are applied.

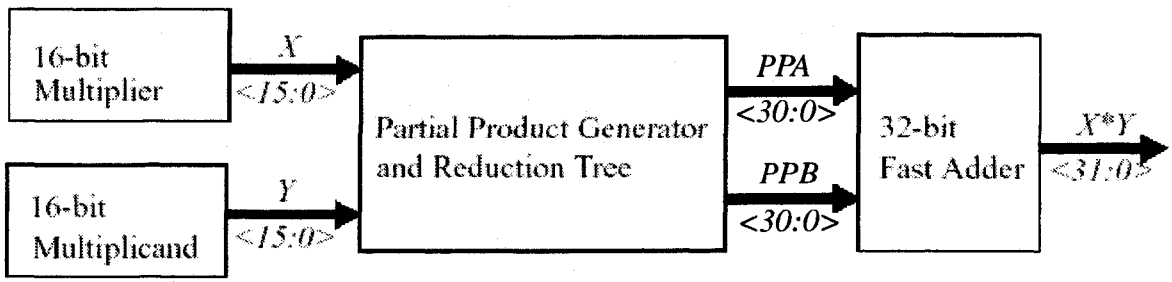


Figure 4.14 Test Bench Structure

Table 4.4Input Test Vectors for Both Multipliers

	Binary Bits																
Multiplicand A	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Decimal
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65535
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	34832
	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	1	2633
	1	0	1	0	0	0	0	1	0	1	0	1	1	1	0	1	41309
	0	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	27076
	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	0	60374
	0	1	0	1	0	0	0	1	1	1	1	0	1	0	1	1	20971
	1	1	0	1	1	0	0	1	1	1	1	1	1	0	1	1	55803
	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	3584
	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	42004
	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	1	11341
	1	0	1	0	1	1	1	1	0	1	0	1	1	1	0	1	44893
	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	16832
	1	1	0	0	1	0	0	1	1	1	0	1	0	0	1	0	51666
	0	1	0	1	1	0	1	1	1	1	1	0	1	0	1	1	23531
Multiplier B	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	Decimal
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65535
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	8465
	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	4626
	0	0	1	1	0	0	1	1	1	0	1	0	0	0	1	1	13219
	1	1	0	0	1	1	1	0	1	0	0	1	0	1	0	0	52884
	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	59285
	0	1	0	1	0	1	0	0	0	0	0	0	1	1	1	0	21518
	0	1	1	1	0	1	0	1	0	0	1	1	1	1	1	1	30015
	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	592
	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	1	11201
	1	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	37586
	1	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	46067
	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	17476
	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	25941
	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0	24158

4.5.2 Simulation Results

The results in terms of delay obtained through simulation of the 16-bit multiplier partial product reduction are shown in tables 4.5 and 4.6. Alternatively, figures 4.7 and 4.8 depict the results for the complete 16-bit multipliers. Blank spaces in these tables represent input transitions where there are no changes in the individual output bit. It must be noted that even though the output bit may not change, this does not reflect any limit on the amount of switching activity within the reduction tree itself. Glitch conditions where there are either voltage spikes or drops for an input transition leading to no change in output are measured

as a delay. When one of the multipliers experience this glitch condition and the other does not, the one with no glitch has a delay of zero recorded.

Table 4.516-Bit Partial Product Reduction Delay Using CMOS1 4:2 Compressors

OUTPUT	DELAY (ns)															AVERAGE	MAX
PPA30	0.193	0.4	0.19	0.4	0.19											0.27	0.4
PPA29	0.605	0.75	0.53	0.81	0.51	0.45	0.51	0.48								0.58	0.81
PPA28	0.391	0.53	0.46	0.53	0.46	1.07	0.7	0.39	0.53	0.6	0.63	0.45	0.53	0.45		0.56	1.07
PPA27	0.719	0.56	1.02	0.93	0.82	1.01	0.93	1.22	1.1	0.66	0.87					0.89	1.22
PPA26	0.417	0.9	0.96	1.36	1.16	0.99	1.17	1.28	1.04	1.18						1.05	1.36
PPA25	1.2	1.79	1.28	1.42	1.08	0.86	0.84	1.27	1.03	1.15	0.88					1.16	1.79
PPA24	0.664	0.8	1.05	0.78	1.7	1.33	1.48	0.75	0.65	1.3	1.01	0.68	0.85			1.04	1.7
PPA23	1.152	0.76	0.88	1.69	1.23	1.38	1.09	1.47	1.19	1.27	1.27	0	0			1.03	1.69
PPA22	1.149	1.57	1.66	1.2	0.86	1.66	1.07	1.25	1.29	1.52	0.82	1.3	1.66			1.34	1.66
PPA21	0.745	1	1.56	1.88	1.42	1.21	1.6	0.88	1.7	1.69	1.18	1.82	1.69			1.41	1.88
PPA20	0.991	0.98	1.04	1.62	0.85	1.91	1.5	1.52	0.82	1.76	0.79	1.46	1.91	1.15	0	1.17	1.91
PPA19	0.748	1.24	0.86	1.61	1.51	1.66	1.39	1.37	1.22	1.64	1.7	1	0.77	1.48		1.30	1.7
PPA18	0.748	1.68	1.66	0.83	1.11	1.62	0.87	1.4	1.19	1.27	1.79	1.81	1.47	1.7		1.33	1.79
PPA17	0.823	1.09	1.76	1.44	1.2	0.78	1.37	1.06	1.39	1.42	1.85	1.43	1.52			1.32	1.85
PPA16	0.96	1.53	1.62	1.29	1.52	1.55	1.48	1.54	1.66	1.9	1.7	1.47	1.45			1.48	1.7
PPA15	1.021	1.48	1.53	0.84	1.57	1.18	0.67	1.77	1.34	1.04	1.46	1.36	1.66			1.30	1.77
PPA14	1.162	1.09	1.61	1.39	1.67	1	1.62	1.43	1.57	1.5	1.15	1.77	1.02			1.37	1.77
PPA13	1.658	1.11	1.28	1.19	1.55	1.65	1.76	1.56	1.57	1.25	1.43	1.82	1.67			1.50	1.82
PPA12	1.09	1.4	1.65	1.83	1.06	1.56	1.64	1.81	1.93	1.61						1.45	1.93
PPA11	0.86	1.28	1.68	1.15	1.56	1.18	1.51	1.45	1.55	1.51	1.09	1.05	1.55	1.66		1.36	1.68
PPA10	1.11	1.67	1.42	1.84	1.36	1.39	1.76	1.5	1.62	1.62	1.32	1.6				1.49	1.84
PPA9	1.688	1.11	1.88	1.3	1.89	1.11	1.21	1.78	1.1	0.96	1.03	1.39	1.36	1.53		1.38	1.89
PPA8	0.77	1.49	0.96	1.37	0.92	1.24	0.97	1.23	1.13	0.98	1.09					1.09	1.49
PPA7	1.11	0.9	1.28	1.33	0.88	1.11	1.19	1.27	1.13	1.27						1.15	1.33
PPA6	0.96	0.86	0.98	0.52	0.7	0.96	0.63	1.19								0.69	1.19
PPA5	1.153	0.74	1.12	1.26	0.87	0.84	0.76	1.12	1.26	0.89						1.00	1.26
PPA4	0.55	0.99	0.88	0.47	0.99	0.94	0.85	0.66	1.01	0.56	0.48	0.56	0.74			0.72	1.01
PPA3	0.623	0.74	0.77	0.65	0.87	0.63	0.64	0.62	0.71	0.65	0.87					0.71	0.87
PPA2																0.00	0
PPA1	0.199	0.43	0.2	0.43	0.2	0.43	0.2	0.43								0.31	0.43
PPA0	0.199	0.41	0.2	0.41	0.2	0.41	0.2									0.28	0.41
																1.03	**1.93**
PPB30	0.854	0.68	0.41													0.48	0.68
PPB29	0.664	0.63	0.37	0.54	0.49	0.55	0.37	0.55	0.37	0	0	0				0.38	0.664
PPB28	0.543	0.89	0.81	0.93	0.51	0.58	0.97	0								0.56	0.89
PPB27	0.583	1.01	1.29	1.07	0.93	0.4	1.09									0.91	1.29
PPB26	0.775	1.18	1.83	0.92	0.94	0.71	1.06	1.05								1.00	1.83
PPB25	0.725	1.36	1.05	0.57	1.27	1.21	0.84	0.75								0.97	1.36
PPB24	0.79	0.84	1.7	1.94	0.98	1	1.02	1.4	1.19	1.23						1.15	1.7
PPB23	0.922	1.49	0.92	1.15	1.15	0.81	0.73	1.57								1.09	1.57
PPB22	0.826	1.49	1.79	1.83	1.52	1.61	1.23	1.61	1.09	1.73	1.6					1.44	1.79
PPB21	0.757	1.44	0.87	0.95	1.41	0.75	0.68	1.66	1.26	1.52	1.83					1.19	1.83
PPB20	1.52	1.37	1.56	1.45	1.3	1.29	1.63	1.17	1.36	1.28						1.39	1.63
PPB19	0.831	1.54	0.81	1.33	1.1	1.18	1.73	0.94	1.38	1.62						1.25	1.73
PPB18	0.944	1.35	0.91	1.47	1.33	1.34	1.76	1.33	1.45							1.32	1.76
PPB17	0.833	1.45	1.54	1.2	1.09	1.42	1.17	1.22	1.62	1.26	1.36					1.29	1.62
PPB16	0.584	1.54	0.75	1.09	1.4	1.56	1.26	1.25	1.36	1.09	1.28	1.58				1.25	1.58
PPB15	0.654	1.14	1.3	1.49	0.92	1.63	1.53	1.34	1.13	1.44	1.33					1.26	1.63
PPB14	0.694	1.24	1.49	1.55	1.67	1.36	1.72	1.6								1.42	1.72
PPB13	0.58	1.56	1.18	0.76	0.99	1.39	1.34	1.85	1.52							1.24	1.85
PPB12	0.611	1.2	1.61	0.85	1.47	1.08	1.42	0.79	1.47	1.36	1.52	1.17	1.6			1.24	1.61
PPB11	0.597	1.34	1.3	1.28	1.25	1.42	1.13	1.23								1.19	1.42
PPB10	0.614	1.76	0.92	1.62	1.05	0.79	0.63	0.87	0.96	1.27	1.47					1.11	1.82
PPB9	0.621	0.78	0.62	1.15	0.88	0.96										0.84	1.15
PPB8	0.685	0.76	1.25	0.79	0.65	1.18										0.89	1.25
PPB7	0.39	0.89	0.42	0.9	0.63	0.36	0.9	0.42	0.77	0.43						0.61	0.9
PPB6	0.45	0.74	0.88	1.04												0.78	1.04
PPB5	0.361	0.81	0.38	0.57	0.91	0.57	0.46	0.69								0.59	0.91
PPB4	0.425	0.67	0.58	0.82	0.55	0.94	0.56	0.59	0.82							0.65	0.82
PPB3																0.00	0
PPB2	0.2	0.43	0.25	0.41	0.2	0.43	0.25	0.41	0.2							0.31	0.43
PPB1	0.2	0.43	0.25	0.41	0.2	0.43	0.25									0.31	0.43
PPB0																0.00	0
																0.90	**1.85**

Table 4.616-Bit Partial Product Reduction Delay Using PASS2 4:2 Compressors

OUTPUT	DELAY (ns)														AVERAGE	MAX
PPA30	0.199	0.4	0.19	0.4	0.19										0.27	0.4
PPA29	0.526	0.81	0.53	0.82	0.51	0.45	0.5	0.48							0.58	0.82
PPA28	0.728	0.92	0.53	0.9	0.72	0.69	0.33	0.92	0.67	0.85	0.71	0.66	0	0	0.64	0.92
PPA27	1.05	1.18	0.9	1.5	1.08	1.96	1.05	0.95	0	0	0				0.88	1.96
PPA26	0.93	1.39	1.15	1.41	1.43	0.86	1.24	0	0	0					0.94	1.43
PPA25	0.78	1.66	1.64	1.79	1.07	1.67	0.87	1.3	0.92	1.97	1.52				1.38	1.97
PPA24	1.48	1.66	1.4	1.47	1.57	1.54	1.28	1.99	1.44	1.14	1.49	0	0		1.26	1.99
PPA23	1.44	1.43	1.33	1.18	1.54	2.34	1.2	1.51	1.56	1.64	1.66	1.76	2.18		1.60	2.34
PPA22	1.22	1.75	1.31	1.53	1.3	1.98	2.04	1.5	1.68	1.61	1.743	0	0		1.31	2.04
PPA21	1.3	1.93	1.61	1.44	1.87	1.31	1.92	0.86	1.32	1.69	1.02	0	0		1.25	1.93
PPA20	1.26	1.05	1.76	1.82	1.82	1.77	1.75	1.46	1.59	1.29	1.47	1.65	1.22	1.09	1.72	1.63	1.99
PPA19	1.35	1.73	1.28	2.08	1.45	1.55	1.47	1.53	1.56	1.34	1.82	1.89	0	0		1.36	2.08
PPA18	0.67	1.44	1.39	1.67	1.5	1.95	1.51	1.69	0	0	0	0	0	0		0.84	1.95
PPA17	1.14	2.05	1.65	1.91	1.4	1.56	1.13	2.09	1.47	1.72	0	0	0			1.24	2.09
PPA16	1.66	1.79	1.6	1.72	1.88	1.9	2.06	1.4	1.57	0	0	0	0			1.16	2.06
PPA15	1.4	1.69	1.15	1.63	1.19	1.77	1.3	1.57	1.72	1.84	2.11	0	0			1.34	2.11
PPA14	1.87	1.21	1.69	1.6	1.73	1.47	1.06	1.72	1.61	1.78	1.52	1.52	2.14			1.62	2.14
PPA13	1.31	1.64	1.7	1.66	1.76	1.22	1.53	1.49	1.84	0	0	0	0			1.09	1.84
PPA12	1.6	1.29	1.4	1.97	1.7	1.67	1.47	0	0	0						1.04	1.6
PPA11	1.3	1.43	1.79	1.58	1.85	1.7	0	0	0	0	0	0	0	0		0.69	1.85
PPA10	1.85	1.56	2.15	1.41	1.6	1.76	1.89	1.34	1.61	1.7	0	0				1.42	2.15
PPA9	1.78	1.27	1.46	1.55	1.16	1.39	1.88	1.29	1.73	1.34	2.11	1.58	0	0		1.32	2.11
PPA8	1.51	1.2	1.35	0.88	1.2	1.49	1.14	1.29	0	0	0					0.91	1.51
PPA7	1.35	1.21	1.08	1.41	1.3	1.01	1.4	1.03	0	0						0.98	1.41
PPA6	1.45	1.49	0.63	1.44	0	0	0	0								0.63	1.49
PPA5	1.01	1.08	0.93	1.44	0.88	1.02	1.32	0.96	1.47	0						1.01	1.47
PPA4	1.06	0.79	1.91	0.8	1.11	0.42	1.07	0.84	1.06	0.77	1.07	0.87	0			0.84	1.11
PPA3	0.35	1.03	0.81	1	0.7	1.01	0	0	0	0	0					0.45	1.03
PPA2																0.00	0
PPA1	0.2	0.43	0.2	0.43	0.2	0.43	0.2	0.43								0.32	0.43
PPA0	0.2	0.41	0.2	0.41	0.2	0.41	0.2									0.29	0.41
																0.97	**2.34**
PPB30	0.35	0.74	0.41													0.50	0.74
PPB29	0.25	0.67	0.67	0.7	0.25	0.7	0.26	0.67	0.79	0.39	0.82	0.26				0.54	0.82
PPB28	0.25	0.68	1.17	0.79	1.15	0.51	0.79	0.42								0.72	1.17
PPB27	0.32	1.55	0.95	1.45	0.64	0	0									0.70	1.55
PPB26	0.44	2.48	1.52	2.06	0.93	2.34	0	0								1.22	2.48
PPB25	0.68	2.01	0.73	2.3	0.96	0	0	0								0.84	2.3
PPB24	0.77	1.58	2.54	1.67	2.31	1.44	0	0	0	0						1.02	2.54
PPB23	0.56	1.7	0.82	0	0	0	0	0								0.39	1.7
PPB22	0.95	2.13	1.67	1.74	0	0	0	0	0	0	0					0.59	2.13
PPB21	0.62	2.35	1.41	1.81	0.7	2.29	1.13	1.99	1.34	0	0					1.24	2.35
PPB20	0.14	2.39	2.26	1.54	0	0	0	0	0	0						0.63	2.39
PPB19	1.06	2.02	1.81	2.17	1.72	2.34	1.1	0	0	0						1.22	2.34
PPB18	1.04	1.9	1.11	2.11	2.16	2.07	0	0	0							1.19	2.07
PPB17	0.97	2.1	1.27	2.94	0.94	2.31	1.65	2.26	1.42	0	0					1.44	2.94
PPB16	0.91	1.96	0.95	1.64	1.52	2.13	1.27	1.74	0	0	0	0				1.01	2.13
PPB15	0.88	2.23	1.51	0.97	1.93	2.38	0.95	1.57	0	0	0					1.13	2.38
PPB14	0.83	1.45	2.24	1.04	2.09	1.37	0	0								1.13	2.24
PPB13	0.84	1.43	0.57	0	0	0	0	0	0							0.32	1.43
PPB12	0.73	2.16	1.31	2.33	0.82	1.52	0.65	1.6	0	0	0	0	0			0.86	2.33
PPB11	0.83	2.65	1.15	2.34	1.48	0	0	0								1.06	2.65
PPB10	0.73	1.85	1.12	1.72	1.28	1.85	0.74	1.03	1.18	1.68	2.04					1.46	2.04
PPB9	0.77	1.6	0.72	1.98	1.09	0										1.03	1.98
PPB8	0.54	1.4	1.1	1.98	0.7	1.26										1.17	1.98
PPB7	0.54	2.02	0.45	1.03	0.5	2.01	0.45	0.62	0.42	0						0.80	2.02
PPB6	0.49	1.12	0.91	0												0.63	1.12
PPB5	0.44	0.7	0.29	0.91	0.47	0.79	0.29	1.15								0.63	1.15
PPB4	0.29	0.63	0.29	0.61	0.68	0.9	0.39	0	0							0.56	0.9
PPB3																0.00	0
PPB2	0.2	0.43	0.25	0.41	0.2	0.43	0.25	0.41	0.2							0.31	0.43
PPB1	0.2	0.43	0.25	0.41	0.2	0.43	0.25									0.31	0.43
PPB0																0.00	0
																0.79	**2.94**

Table 4.716-Bit Multiplier Delay Using CMOS1 4:2 Compressors

OUTPUT	DELAY (ns)														AVERAGE	MAX
SUM31	0.7	1.24	1.33	1.16	0.85	1.22	0.97	1.08	2.24	1.31	0.79	1.19			1.17	2.24
Sum30	1.023	1.04	1.56	1.45	1.36	0.85	1.2	1.37	2.21	1.25	1.15	1.27			1.31	2.21
Sum29	1.62	1.9	1.52	1.97	1.36	1.06	1.97	2.16	2.1	1.57	1.28	1.87			1.65	2.16
Sum28	0.93	1.26	1.4	0.81	1.42	1.46	0.88	1.07	1.65	1.82	1.84	1.29	0.95	2.48	1.38	2.48
Sum27	1.155	1.35	1.77	1.73	1.17	1.76	1.22	1.59	1.79	1.02	1.27	2.48	1.47		1.52	2.48
Sum26	1.24	1.64	1.56	2.46	0.71	1.76	1.99	1.25	2.46	1.27					1.63	2.46
Sum25	1.16	2.1	2.06	1.77	0.83	2.41	1.06	1.71	1.66	1.04	2.19	2.58	2.62		1.84	2.62
Sum24	0.94	1.26	2.09	1.59	1.71	1.75	2.1	1.07	1.47	1.06	1.5	1.9	2.04	2.38	1.63	2.38
Sum23	0.91	1.97	1.97	1.61	1.73	1.97	1.73	1.25	1.76	2.15	1.31	0.85	2.36		1.63	2.36
Sum22	2.35	1.6	1.97	1.7	1.65	1.21	2.02	1.69	1.7	1.64	2.17	2.29	0		1.69	2.35
Sum21	2.29	1.91	2.1	2.46	1.79	1.69	1.9	1.91	1.35	0.91	1.69	2.29	1.69		1.91	2.29
Sum20	1.12	2.2	2.33	1.57	2.09	1.86	1.47	1.7	1.97	2.33	2.01	1.69	2.09	1.79	1.87	2.33
Sum19	1.46	2.22	2.09	2.01	1.86	1.97	1.47	1.54	1.84	2.16	1.61	1.75			1.86	2.29
Sum18	2.2	1.44	1.47	1.96	1.76	1.41	1.13	2.27	1.44	1.86	1.53	1.92	2		1.72	2.27
Sum17	1.24	2.19	1.95	1.86	1.84	2.24	1.93	2.2	2.23	1.29	1.84	1.62	1.06	1.64	1.80	2.24
Sum16	1.389	1.13	1.49	1.74	1.74	1.58	1.87	1.45	1.96	1.84	1.36	1.64	1.68	2.36	2.2	1.70	2.36
Sum15	1.16	2.07	1.22	1.93	1.01	2.19	1.08	1.21	2.16	2.56	2.09	2.36	1.64		1.89	2.56
Sum14	1.59	1.35	1.1	1.88	2.04	1.76	1.76	1.64	1.93	2.56	2.01	2.17	2.01		1.83	2.56
Sum13	1.866	1.45	1.98	2.15	2.27	1.98	2.54	1.71	1.78	1.87	2.47	1.91	1.95	1.92	1.99	2.54
Sum12	1.212	1.23	1.54	2.03	1.8	1.97	1.1	2.17	1.57	2	2.18	1.62	1.65	2.33	1.74	2.33
Sum11	1.28	1.01	1.42	2.15	1.43	1.86	1.88	1.69	1.86	1.7	1.59	2.15	0.89	1.94	1.09	1.51	2.15
Sum10	1.21	1.32	2.05	1.55	1.98	1.48	1.58	1.96	1.64	0.94	1.55	1.63	2.12		1.62	2.12
Sum9	1.89	1.25	1.5	1.61	2.1	1.11	1.63	1.99	1.25	1.17	1.78	1.52	1.29	1.73	1.55	2.1
Sum8	0.87	0.92	1.69	1.1	1.53	1.35	1.41	0.97	1.02	1.5	1.27	1.55	1.84		1.31	1.84
Sum7	1.09	1.32	1.59	1.48	1.39	1.45	0.77	1.05	1.52	1.48	1.29	1.84			1.34	1.84
Sum6	1.05	1.52	1	0.83	1.2	1.3	0.81	1.49	0.85	1.04	1.39				1.13	1.52
Sum5	1.36	0.89	1.46	0.96	1.12	1.19	0.71	0.91	1.2	1.46	1.03	1.04			1.12	1.46
Sum4	0.74	0.7	1.2	1.04	0.6	0.85	1.1	1.04	0.85	0.7	1.22	0.7	0.71		0.88	1.22
Sum3	0.95	0.98	0.9	0.84	0.78	0.89	0.92	0.6							0.86	0.98
Sum2	0.6	0.49	0.35	0.47	0.3	0.71	0.59	0.49	0.35	0.47	0.3				0.47	0.71
Sum1	0.416	0.59	0.42	0.49	0.59	0.5	0.41	0.59	0.42	0.49	0.59				0.50	0.59
Sum0	0.417	0.57	0.42	0.57	0.42	0.57	0.42								0.48	0.57
															1.45	2.62

Table 4.816-Bit Multiplier Delay Using PASS2 4:2 Compressors

OUTPUT	DELAY (ns)														AVERAGE	MAX
Sum31	1.12	1.05	1.44	1.36	1	1.2	1.93	1.35	1.15	1.08	1.02	0			1.09	1.44
Sum30	1.12	1.26	1.2	1.32	1.46	0.89	0.67	1.41	1.41	0	0	0			0.90	1.46
Sum29	1.98	2.07	1.7	1.71	1.17	1.17	0.66	0.68	1.35	1.98	0	0			1.17	2.07
Sum28	1.38	1.55	1.48	1.88	1.11	1.52	1.41	1.21	1.07	1.32	0.99	2.3	1.8	0	1.36	2.3
Sum27	1.12	1.84	1.85	1	1.2	1.33	1.35	1.51	0.86	2.0	0	0	0		1.07	2.0
Sum26	0.482	1.82	1	1.83	1	2.03	1.39	1.08	2.27	1.41					1.43	2.27
Sum25	1.65	1.72	1.81	0.72	1.07	0.91	2.05	2.11	1.92	1.52	2.19	1.68	0		1.46	2.19
Sum24	0.98	1.55	1.45	0.57	1.42	1.41	1.21	1.82	1.78	1.28	1.22	1.88	0	0	1.18	1.88
Sum23	0.442	1.28	1.99	1.64	1.48	1.18	1.66	1.76	1.99	1.75	1.99	0	0		1.32	1.99
Sum22	1.66	1.91	1.48	1.96	1.48	1.74	1.09	1.62	1.7	1.84	1.84	1.52	1.74		1.66	1.96
Sum21	1.615	1.9	1.67	1.56	1.9	1.84	1.98	1.56	1.87	1.87	1.44	2.27	2.21		1.71	2.27
Sum20	1.4	1.51	1.7	1.95	1.34	1.65	1.57	1.41	1.14	1.6	2.11	2.13	0	0	1.39	2.13
Sum19	1.35	1.7	1.22	1.78	2.06	1.36	0.94	1.96	1.93	1.78	1.77	1.99			1.66	2.06
Sum18	0.757	2.09	1.74	1.56	1.86	1.87	1.22	1.69	1.94	1.89	1.47	1.74	0		1.53	2.09
Sum17	1.29	2	1.69	1.85	1.46	1.97	1.81	2.31	1.18	1.46	1.68	1.87	1.65	2.03	1.73	2.31
Sum16	1.29	1.59	1.72	1.27	1.74	1.4	1.43	1.95	1.01	1.17	1.8	1.4	1.53	1.83	0	1.41	1.95
Sum15	1.86	1.36	1.72	1.07	1.59	1.85	1.87	1.32	1.73	1.73	2.82	2.04	1.65		1.72	2.82
Sum14	1.26	1.45	1.77	1.71	1.68	0.56	1.04	0.82	2.22	2.82	1.96	1.39	0		1.44	2.82
Sum13	1.17	1.74	1.27	1.61	1.81	2.25	1.94	1.6	2.52	1.39	0	0	0	0	1.28	2.52
Sum12	1.24	1.75	1.47	1.31	0.85	2.03	2.44	1.02	1.32	2.23	1.13	0	0	0	1.20	2.44
Sum11	1.21	1.25	1.05	1.37	1.07	1.18	1.01	1.82	1.69	1.2	2.41	0.75	2.04	1.11	1.65	1.42	2.41
Sum10	1.21	1.17	1.04	1.64	0.96	2.02	1.5	1.28	1.69	1.18	2.21	1.99	0		1.38	2.21
Sum9	1.01	1.74	1.41	1.92	1.5	1.91	1.49	1.84	1.42	1.86	1.29	2.04	1.04	0	1.44	2.04
Sum8	0.93	1.46	1.59	1.28	1.18	1.56	1.3	1.45	1.63	1.05	1.38	1.27	0		1.24	1.63
Sum7	1.09	1.82	1.25	1.24	1.42	1.11	1.86	1.52	0.56	0.44	1.39	0			1.13	1.86
Sum6	1.05	1.44	1.59	0.82	0.99	1.53	0.74	0.84	0	0	0				0.82	1.59
Sum5	0.99	0.68	1.22	1.1	1.38	1.02	0.69	0.88	1	1.15	1.48	0.87			1.05	1.43
Sum4	0.75	1.01	0.93	0.56	0.96	0.94	0.96	0.57	1.01	0.97	0.94	1.01	1.01		0.89	1.01
Sum3	0.707	0.97	0.92	0.96	0.89	0.97	0	0							0.69	0.97
Sum2	0.6	0.49	0.35	0.48	0.3	0.71	0.59	0.49	0.35	0.47	0.3				0.47	0.71
Sum1	0.42	0.59	0.42	0.49	0.59	0.5	0.42	0.59	0.42	0.49	0.59				0.50	0.59
Sum0	0.42	0.57	0.42	0.57	0.42	0.57	0.42								0.48	0.57
															1.22	2.62

Table 4.8 presents a summary of the power associated with both of these multipliers per transition, average and worst case power delay products, average and worst case delays and percent savings.

Table 4.9 Summarized Simulation Results

		CMOS1	PASS2	% Savings
Complete Multiplier	Power (W)	8.79E-03	4.44E-03	49.45
	Average Delay (ηs)	1.45	1.22	15.86
	Max Delay (ηs)	2.62	2.82	-7.63
	Average PDP (ηs*W)	1.27E-02	5.42E-03	57.47
	Max PDP (ηs*W)	2.30E-02	1.25E-02	45.59
Reduction Tree	Average Delay (ηs)	0.97	0.88	9.28
	Max Delay (ηs)	1.93	2.94	-52.33

4.6 Hardware Testing

The fabricated chip was remotely tested. The test equipment was physically located at the Canadian Microelectronics Corporation (CMC) test facility at Queens University in Kingston Ontario. Testing was carried out through a secure shell connection using ***IMS Screens*** software displaying to a workstation in the University of Windsor RCIM lab.

With this multiplier chip, tests regarding speed and power are not feasible due to the large power and delay overhead associated with the I/O circuitry. This overhead being the added multiplexers and shift registers needed in order to reduce the amount of I/O circuitry so that realestate limitations could be met. For this reason, tests were only performed with regards to functionality.

4.6.1 Test Code

The IMS Screens test environment requires test conditions to be inputted in the following form:

"input vector" "clock condition" "expected output"

10000 1 00000000

The first two bits of the input vector are control bits for the 4:1 output multiplexers selecting which of the most significant, second most significant, second least significant and least significant bits of the *32-bit* product to output. The third bit is the advance signal which tells the circuit when all of the sixteen multipliers and multiplicand bits have been serially inputted and multiplication may begin. The last two bits are the actual serially inputted multiplier and multiplicand bits. The clock condition bit selects the transition at which the input vector is to be applied. A clock condition "1" means a transition from low to high. Finally, the expected output bits provide the tester with a base to compare the actual outputs.

4.6.2 Test Results

Test of the digital multiplier provided unexpected results. 4 chips were returned from fabrication and they all yielded slightly different outputs. Unfortunately none of these chips provided the correct expected results. IMS Screens test code and an example of output data is included in Appendix C. A selection of results for two of the tested chips are included in Appendix D.

Because the multiplier simulated flawlessly prior to fabrication, it is believed that the error lies in the added input and output circuitry required to fabricate the multiplier with a reduced number of pins. It is most likely that the added overhead in terms of interconnection complexity and delay as a result of this added I/O circuitry causes some undetermined states within the circuit. This may explain why each chip seems to provide slightly different results.

Because of this added I/O circuitry, it is impossible to conclude at this point that there is indeed a flaw in the multiplier itself. Again, considering the positive results obtained through simulation, it is the strong belief that the error lies in the I/O circuitry and does not reflect on the functionality of the multiplier architecture or the new compressor cells.

Chapter 5

Conclusions

5.1 Summary of Contributions

The purpose of this study has been to explore digital multiplication at the circuit level focussing on the sub-arithmetic cells utilized in partial product reduction. This has led to several contributions in the areas of low-power circuit design and test as well as digital multiplication architecture.

5.1.1 Simulation and Test

Up to this point, no technique for measuring power dissipation in pass-logic circuits has taken into account the fact that some of the current drawn by these circuits is, as the name implies, passed through to the following circuitry. This leads to inaccurate results reflecting not only the power dissipated by the circuit under study but also that of the circuits to which it outputs. Results of this nature may not be fairly compared to results derived the same way from circuits using other logic styles such as standard CMOS which does not pass any signals. Within this work, a more robust standard method of measuring power dissipation has been suggested and implemented. This method may be used for test of both standard CMOS as well as pass-logic circuits without any discrepancies.

5.1.2 Circuit Design and Layout

(3,2) Counters

An overview of selected (3,2) counter circuitries has been presented which includes a proper power comparison using the new proposed methodology. From these results, an optimal configuration for the (3,2) counter has been established. An analysis of the limitations of 10-transistor (3,2) counters has provided some insight as to why these cells are not suitable for traditional applications.

4:2 Compressors

Several 4:2 compressors have been studied and the HSPICE simulation results based on a proposed power measurement methodology have been provided. The summarized results facilitate the selection of a compressor style based on a specific application in terms of speed, power, and size requirements.

Digital Multipliers

Two low-power 16-bit digital multipliers using both a standard CMOS and a new low-power 4:2 compressor cell have been constructed implementing a recently devised compressor layout methodology for the first time to date. Simulation at the hardware level has provided much needed validation for this methodology. Furthermore, delay and power comparisons have shown the superiority of the new compressor cell over the current standard.

5.2 Conclusions

For high speed circuitry, where delay is the key driver, the standard CMOS cell (CMOS1) outperforms all other logic styles. Although these findings may appear to contradict the beliefs of many pass-logic supporters, this result is intuitive when we consider the fact that the technology used is optimized for CMOS logic. Pass-transistor logic is recommended

only where size and power are of importance. Furthermore, it has been observed through this unbiased analysis that the best overall compressor in terms of power dissipation, delay (worst case and average), power-delay product and size is the 30 transistor PASS2 configuration. Table 5.1 summarizes the recommended 4:2 compressors for applications based on overall performance metrics.

Table 5.14:2 Compressor Recommendations in Terms of System Requirements

REQUIREMENT	DELAY (W.C.)	DELAY (AVG)	POWER	SIZE
DELAY (W.C.)	*CMOS1*	*CMOS1*	*PASS2*	*PASS2*
DELAY (AVG)	*CMOS1*	*CMOS1*	*PASS2*	*PASS2*
POWER	*PASS2*	*PASS2*	*TGATE*	*PASS2*
SIZE	*PASS2*	*PASS2*	*PASS2*	*PASS1/2*

REFERENCES

[1] A. Burks, H. Goldstine and J. von Neumann, "Discussion of the logical design of an electronic computing instrument", Datamation, vol. 8, no. 9, pp. 24-31, September, 1962

[2] P. Mokrian, G.M. Howard, G. Jullien and M. Ahmadi, "On the use of 4:2 Compressors for Partial Product Reduction", IEEE CCECE, vol. 1, pp.121-124, May 2003

[3] B. Parhami, "Computer Arithmetic: Algorithms and Hardware Designs", Oxford University Press, New York, 2000

[4] C.S. Wallace, "A suggestion for a fast multiplier", IEEE Transactions on Electronic Computers, vol. 13, pp 14-17, 1964.

[5] Bickerstaff, Schulte and E.E. Swartzlander Jr., "Analysis of column compression multipliers", IEEE Symposium on Computer Arithmetic, pp. 33 -39, 2001

[6] L. Dadda, "The evolution of computer architectures", CompEuro '91, European Computer Conference on Advanced Computer Technology, Reliable Systems and Applications. pp. 9-16, 1991

[7] A. Weinberger, "4:2 Carry-Save Adder Module", IBM Technical Disclosure Bulletin, vol. 23, Jan. 1981

[8] Shen-Fu Hsiao, Ming-Roun Jiang and Jia-Sien Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers", IEEE Electronics Letters, vol. 34, no. 4, pp. 341-343, 1998

[9] K. Prasad and K.K. Parhi, "Low-power 4-2 and 5-2 compressors", Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, vol. 1, pp.129-133, Nov. 2001

[10] P. Kornerup, "Reviewing 4-to-2 adders for multi-operand addition", Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, pp. 218-229, 2002

[11] Chi-Hsiang Yeh and B. Parhami, "Efficient designs for multi-input counters", Asilomar Conference on Signals, Systems, and Computers, vol.2, pp. 1340-1344, 1999

[12] D. Radhakrishnan and A.P. Preethy, "Low power CMOS pass logic 4-2 compressor for high-speed multiplication", Proceeding of the 43rd IEEE Midwest Symposium on Circuits and Systems, pp. 1296-1298, 2000

[13] M. Margala and N.G. Durdle, "Low-Power Low-Voltage 4-2 Compressors for VLSI Applications", Proceedings of the Workshop on Low Power Design, 1999

[14] R.V.K. Pillai, D. Al-Khalili and A.J. Al-Khalili, "Energy delay analysis of partial product reduction methods for parallel multiplier implementation", International Symposium on Low Power Electronics and Design, pp. 201 -204, 1996

[15] Y. Hagihara, S. Inui, A. Yoshikawa, S. Nakazato, S. Iriki, R. Ikeda, Y. Shibue, T. Inaba, M. Kagamihara and M. Yamashina, "A 2.7ns 0.25um CMOS 54x54b multiplier", Proceedings of the IEEE International Solid-State Circuits Conference, 1998

[16] A. Habibi and P.A. Wintz, "Fast Multipliers", IEEE Transactions on Computers, vol. C-19, pp. 153-157, 1970

[17] M. Mehta, V. Parmar and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits", IEEE Symposium on Computer Arithmetic, pp. 43-50, 1991

[18] T. Rhyne and N.R. Strader, "A signed bit-sequential multiplier", IEEE Transactions on Computers, vol. C-35, no. 10, pp. 896-901, 1986

[19] W.J. Stenzel, W.J. Kubitz and G.H. Garcia, "A compact high-speed parallel multiplication scheme", IEEE Transactions on Computers, vol. C-26, no. 10, pp.948-957, 1977

[20] Kwon, Nowka and E.E. Swartzlander Jr., "A 16-bit x 16-bit MAC Design using fast 5:2 Compressors", IEEE International Conference on Application-Specific Systems, Architectures, and Processors, pp. 235 -243, 2000

[21] R.V.K. Pillai, D. Al-Khalili and A.J. Al-Khalili, "Energy delay analysis of partial product reduction methods for parallel multiplier implementation", International Symposium on Low Power Electronics and Design, pp. 201 -204, 1996

[22] Chi-Hsiang Yeh, B. Parhami and Y. Wang, "Designs of counters with near minimal-counting/sampling period and hardware complexity", Asilomar Conference of Signals, Systems and Computers, vol. 2, pp.894-898 , 2000

[23] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume: 10 Issue: 6 , pp. 806-823, December 2002

[24] N. Zhuang and H. Wu, "A new design of the CMOS full adder", IEEE Journal of Solid-State Circuits, vol. 27, no. 5, pp. 840-844, 1992

[25] Radhakrishnan, "Low voltage CMOS full adder cells", Electronics Letters, vol. 35, pp.1792-1794, October 1999

[26] A.M. Shams, T.K. Darwish and M.A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, pp. 20-29, Feb 2002

[27] Lu Junming, Shu Yan, Lin Zhenghui and Wang Ling, "A novel 10-transistor low-power high-speed full adder cell", International Conference on Solid-State and Integrated-Circuit Technology, vol.2, pp.1155-1158, 2001

[28] Sayed and W. Badawy, "Performance analysis of single-bit full adder cells using 0.18, 0.25 and 0.35 um CMOS technologies", IEEE International Symposium on Circuits and Systems (ISCAS),vol. 3, pp.III-559 - III-562, May 2002

[29] H.T. Bui, Y. Wang and Y. Jiang, "Design and analysis of low-power 10-transistor full-adders using novel XOR-XNOR Gates", IEEE Transaction on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 49, no. 1, pp. 25-30, 2002

[30] Lu Junming, Shu Yan, Lin Zhenghui and Wang Ling, "A novel 10-transistor low-power high-speed full adder cell", International Conference on Solid-State and Integrated-Circuit Technology, Volume: 2, pp.1155-1158, 2001

[31] A.M. Shams, T.K. Darwish and M.A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 1, pp. 20-29, 2002

[32] Law, Rofail and Yeo, "Low-Power circuit implementation for partial product addition using pass-transistor logic", IEE Proceedings Circuits Devices Systems, vol. 146, pp. 124-129 June 1999

[33] J.P. Uyemura, "CMOS Logic Circuit Design", Kluwer Academic Publishers, Boston, 1999

[34] "Accolade VHDL Reference Guide", 200-2001, Retrieved January 2005 from Accolade website, http://www.acc-eda.com/vhdlref/

[35] D.C. Hyde, "CSCI 320 Computer Architecture Handbook on Verilog HDL", August 23, 1997, Retrieved January 2005, http://www.eg.bucknell.edu/~cs320/1995-fall/verilog-manual.html

Appendix A

Verilog Code

A.1 16x16-bit Multiplier

```
/*****************************************************************************

  The Following is code for a 16-bit multiplier outputting two final partial
  products to be added using the 32-bit 2-stage CLA ADD32

*****************************************************************************/


module MAIN (A,B,OUTA,OUTB);

   input   [15:0] A;
   input   [15:0] B;
   output  [30:0] OUTA;
   output  [30:0] OUTB;


// Initial partial products

   wire
PPA0,PPA1,PPA2,PPA3,PPA4,PPA5,PPA6,PPA7,PPA8,PPA9,PPA10,PPA11,PPA12,PPA13,
PPA14,PPA15,PPA16,PPA17,PPA18,PPA19,PPA20,PPA21,PPA22,PPA23,PPA24,PPA25,PPA26,
PPA27,PPA28,PPA29,PPA30;
   wire
PPB1,PPB2,PPB3,PPB4,PPB5,PPB6,PPB7,PPB8,PPB9,PPB10,PPB11,PPB12,PPB13,
PPB14,PPB15,PPB16,PPB17,PPB18,PPB19,PPB20,PPB21,PPB22,PPB23,PPB24,PPB25,PPB26,
PPB27,PPB28,PPB29;
   wire
PPC2,PPC3,PPC4,PPC5,PPC6,PPC7,PPC8,PPC9,PPC10,PPC11,PPC12,PPC13,
PPC14,PPC15,PPC16,PPC17,PPC18,PPC19,PPC20,PPC21,PPC22,PPC23,PPC24,PPC25,PPC26,
PPC27,PPC28;
   wire
PPD3,PPD4,PPD5,PPD6,PPD7,PPD8,PPD9,PPD10,PPD11,PPD12,PPD13,
PPD14,PPD15,PPD16,PPD17,PPD18,PPD19,PPD20,PPD21,PPD22,PPD23,PPD24,PPD25,PPD26,
PPD27;
   wire
PPE4,PPE5,PPE6,PPE7,PPE8,PPE9,PPE10,PPE11,PPE12,PPE13,
PPE14,PPE15,PPE16,PPE17,PPE18,PPE19,PPE20,PPE21,PPE22,PPE23,PPE24,PPE25,PPE26;
```

```
   wire
PPF5,PPF6,PPF7,PPF8,PPF9,PPF10,PPF11,PPF12,PPF13,PPF14,PPF15,PPF16,PPF17,PPF18,P
PF19,PPF20,PPF21,PPF22,PPF23,PPF24,PPF25;
   wire
PPG6,PPG7,PPG8,PPG9,PPG10,PPG11,PPG12,PPG13,PPG14,PPG15,PPG16,PPG17,PPG18,PPG19,
PPG20,PPG21,PPG22,PPG23,PPG24;
   wire
PPH7,PPH8,PPH9,PPH10,PPH11,PPH12,PPH13,PPH14,PPH15,PPH16,PPH17,PPH18,PPH19,PPH20
,PPH21,PPH22,PPH23;
   wire
PPI8,PPI9,PPI10,PPI11,PPI12,PPI13,PPI14,PPI15,PPI16,PPI17,PPI18,PPI19,PPI20,
PPI21,PPI22;
   wire
PPJ9,PPJ10,PPJ11,PPJ12,PPJ13,PPJ14,PPJ15,PPJ16,PPJ17,PPJ18,PPJ19,PPJ20,PPJ21;
   wire
PPK10,PPK11,PPK12,PPK13,PPK14,PPK15,PPK16,PPK17,PPK18,PPK19,PPK20;
   wire
PPL11,PPL12,PPL13,PPL14,PPL15,PPL16,PPL17,PPL18,PPL19;
   wire
PPM12,PPM13,PPM14,PPM15,PPM16,PPM17,PPM18;
   wire
PPN13,PPN14,PPN15,PPN16,PPN17;
   wire
PPO14,PPO15,PPO16;
   wire
PPP15;


   // Define GROUND and set to zero

   wire       GROUND;
   assign     GROUND = 1'b0;


   assign OUTA[0] = PPA0;
   assign OUTA[1] = PPA1;
   assign OUTA[30] = PPA30;
   assign OUTB[0] = GROUND;
   assign OUTB[1] = PPB1;
   assign OUTB[2] = PPC2;
   assign OUTB[30] = GROUND;


   // Generating initial partial products

   ANDGATE andA0 (PPA0,A[0],B[0]);
   ANDGATE andA1 (PPA1,A[0],B[1]);
   ANDGATE andA2 (PPA2,A[0],B[2]);
   ANDGATE andA3 (PPA3,A[0],B[3]);
   ANDGATE andA4 (PPA4,A[0],B[4]);
   ANDGATE andA5 (PPA5,A[0],B[5]);
   ANDGATE andA6 (PPA6,A[0],B[6]);
   ANDGATE andA7 (PPA7,A[0],B[7]);
   ANDGATE andA8 (PPA8,A[0],B[8]);
   ANDGATE andA9 (PPA9,A[0],B[9]);
   ANDGATE andA10 (PPA10,A[0],B[10]);
   ANDGATE andA11 (PPA11,A[0],B[11]);
   ANDGATE andA12 (PPA12,A[0],B[12]);
   ANDGATE andA13 (PPA13,A[0],B[13]);
```

```
ANDGATE andA14 (PPA14,A[0],B[14]);
ANDGATE andA15 (PPA15,A[0],B[15]);
ANDGATE andA16 (PPA16,A[1],B[15]);
ANDGATE andA17 (PPA17,A[2],B[15]);
ANDGATE andA18 (PPA18,A[3],B[15]);
ANDGATE andA19 (PPA19,A[4],B[15]);
ANDGATE andA20 (PPA20,A[5],B[15]);
ANDGATE andA21 (PPA21,A[6],B[15]);
ANDGATE andA22 (PPA22,A[7],B[15]);
ANDGATE andA23 (PPA23,A[8],B[15]);
ANDGATE andA24 (PPA24,A[9],B[15]);
ANDGATE andA25 (PPA25,A[10],B[15]);
ANDGATE andA26 (PPA26,A[11],B[15]);
ANDGATE andA27 (PPA27,A[12],B[15]);
ANDGATE andA28 (PPA28,A[13],B[15]);
ANDGATE andA29 (PPA29,A[14],B[15]);
ANDGATE andA30 (PPA30,A[15],B[15]);

ANDGATE andB1 (PPB1,A[1],B[0]);
ANDGATE andB2 (PPB2,A[1],B[1]);
ANDGATE andB3 (PPB3,A[1],B[2]);
ANDGATE andB4 (PPB4,A[1],B[3]);
ANDGATE andB5 (PPB5,A[1],B[4]);
ANDGATE andB6 (PPB6,A[1],B[5]);
ANDGATE andB7 (PPB7,A[1],B[6]);
ANDGATE andB8 (PPB8,A[1],B[7]);
ANDGATE andB9 (PPB9,A[1],B[8]);
ANDGATE andB10 (PPB10,A[1],B[9]);
ANDGATE andB11 (PPB11,A[1],B[10]);
ANDGATE andB12 (PPB12,A[1],B[11]);
ANDGATE andB13 (PPB13,A[1],B[12]);
ANDGATE andB14 (PPB14,A[1],B[13]);
ANDGATE andB15 (PPB15,A[1],B[14]);
ANDGATE andB16 (PPB16,A[2],B[14]);
ANDGATE andB17 (PPB17,A[3],B[14]);
ANDGATE andB18 (PPB18,A[4],B[14]);
ANDGATE andB19 (PPB19,A[5],B[14]);
ANDGATE andB20 (PPB20,A[6],B[14]);
ANDGATE andB21 (PPB21,A[7],B[14]);
ANDGATE andB22 (PPB22,A[8],B[14]);
ANDGATE andB23 (PPB23,A[9],B[14]);
ANDGATE andB24 (PPB24,A[10],B[14]);
ANDGATE andB25 (PPB25,A[11],B[14]);
ANDGATE andB26 (PPB26,A[12],B[14]);
ANDGATE andB27 (PPB27,A[13],B[14]);
ANDGATE andB28 (PPB28,A[14],B[14]);
ANDGATE andB29 (PPB29,A[15],B[14]);

ANDGATE andC1 (PPC2,A[2],B[0]);
ANDGATE andC2 (PPC3,A[2],B[1]);
ANDGATE andC3 (PPC4,A[2],B[2]);
ANDGATE andC4 (PPC5,A[2],B[3]);
ANDGATE andC5 (PPC6,A[2],B[4]);
ANDGATE andC6 (PPC7,A[2],B[5]);
ANDGATE andC7 (PPC8,A[2],B[6]);
ANDGATE andC8 (PPC9,A[2],B[7]);
ANDGATE andC9 (PPC10,A[2],B[8]);
ANDGATE andC10 (PPC11,A[2],B[9]);
ANDGATE andC11 (PPC12,A[2],B[10]);
```

```
ANDGATE andC12 (PPC13,A[2],B[11]);
ANDGATE andC13 (PPC14,A[2],B[12]);
ANDGATE andC14 (PPC15,A[2],B[13]);
ANDGATE andC15 (PPC16,A[3],B[13]);
ANDGATE andC16 (PPC17,A[4],B[13]);
ANDGATE andC17 (PPC18,A[5],B[13]);
ANDGATE andC18 (PPC19,A[6],B[13]);
ANDGATE andC19 (PPC20,A[7],B[13]);
ANDGATE andC20 (PPC21,A[8],B[13]);
ANDGATE andC21 (PPC22,A[9],B[13]);
ANDGATE andC22 (PPC23,A[10],B[13]);
ANDGATE andC23 (PPC24,A[11],B[13]);
ANDGATE andC24 (PPC25,A[12],B[13]);
ANDGATE andC25 (PPC26,A[13],B[13]);
ANDGATE andC26 (PPC27,A[14],B[13]);
ANDGATE andC27 (PPC28,A[15],B[13]);

ANDGATE andD1 (PPD3,A[3],B[0]);
ANDGATE andD2 (PPD4,A[3],B[1]);
ANDGATE andD3 (PPD5,A[3],B[2]);
ANDGATE andD4 (PPD6,A[3],B[3]);
ANDGATE andD5 (PPD7,A[3],B[4]);
ANDGATE andD6 (PPD8,A[3],B[5]);
ANDGATE andD7 (PPD9,A[3],B[6]);
ANDGATE andD8 (PPD10,A[3],B[7]);
ANDGATE andD9 (PPD11,A[3],B[8]);
ANDGATE andD10 (PPD12,A[3],B[9]);
ANDGATE andD11 (PPD13,A[3],B[10]);
ANDGATE andD12 (PPD14,A[3],B[11]);
ANDGATE andD13 (PPD15,A[3],B[12]);
ANDGATE andD14 (PPD16,A[4],B[12]);
ANDGATE andD15 (PPD17,A[5],B[12]);
ANDGATE andD16 (PPD18,A[6],B[12]);
ANDGATE andD17 (PPD19,A[7],B[12]);
ANDGATE andD18 (PPD20,A[8],B[12]);
ANDGATE andD19 (PPD21,A[9],B[12]);
ANDGATE andD20 (PPD22,A[10],B[12]);
ANDGATE andD21 (PPD23,A[11],B[12]);
ANDGATE andD22 (PPD24,A[12],B[12]);
ANDGATE andD23 (PPD25,A[13],B[12]);
ANDGATE andD24 (PPD26,A[14],B[12]);
ANDGATE andD25 (PPD27,A[15],B[12]);

ANDGATE andE1 (PPE4,A[4],B[0]);
ANDGATE andE2 (PPE5,A[4],B[1]);
ANDGATE andE3 (PPE6,A[4],B[2]);
ANDGATE andE4 (PPE7,A[4],B[3]);
ANDGATE andE5 (PPE8,A[4],B[4]);
ANDGATE andE6 (PPE9,A[4],B[5]);
ANDGATE andE7 (PPE10,A[4],B[6]);
ANDGATE andE8 (PPE11,A[4],B[7]);
ANDGATE andE9 (PPE12,A[4],B[8]);
ANDGATE andE10 (PPE13,A[4],B[9]);
ANDGATE andE11 (PPE14,A[4],B[10]);
ANDGATE andE12 (PPE15,A[4],B[11]);
ANDGATE andE13 (PPE16,A[5],B[11]);
ANDGATE andE14 (PPE17,A[6],B[11]);
ANDGATE andE15 (PPE18,A[7],B[11]);
ANDGATE andE16 (PPE19,A[8],B[11]);
```

```
ANDGATE andE17 (PPE20,A[9],B[11]);
ANDGATE andE18 (PPE21,A[10],B[11]);
ANDGATE andE19 (PPE22,A[11],B[11]);
ANDGATE andE20 (PPE23,A[12],B[11]);
ANDGATE andE21 (PPE24,A[13],B[11]);
ANDGATE andE22 (PPE25,A[14],B[11]);
ANDGATE andE23 (PPE26,A[15],B[11]);

ANDGATE andF1 (PPF5,A[5],B[0]);
ANDGATE andF2 (PPF6,A[5],B[1]);
ANDGATE andF3 (PPF7,A[5],B[2]);
ANDGATE andF4 (PPF8,A[5],B[3]);
ANDGATE andF5 (PPF9,A[5],B[4]);
ANDGATE andF6 (PPF10,A[5],B[5]);
ANDGATE andF7 (PPF11,A[5],B[6]);
ANDGATE andF8 (PPF12,A[5],B[7]);
ANDGATE andF9 (PPF13,A[5],B[8]);
ANDGATE andF10 (PPF14,A[5],B[9]);
ANDGATE andF11 (PPF15,A[5],B[10]);
ANDGATE andF12 (PPF16,A[6],B[10]);
ANDGATE andF13 (PPF17,A[7],B[10]);
ANDGATE andF14 (PPF18,A[8],B[10]);
ANDGATE andF15 (PPF19,A[9],B[10]);
ANDGATE andF16 (PPF20,A[10],B[10]);
ANDGATE andF17 (PPF21,A[11],B[10]);
ANDGATE andF18 (PPF22,A[12],B[10]);
ANDGATE andF19 (PPF23,A[13],B[10]);
ANDGATE andF20 (PPF24,A[14],B[10]);
ANDGATE andF21 (PPF25,A[15],B[10]);

ANDGATE andG1 (PPG6,A[6],B[0]);
ANDGATE andG2 (PPG7,A[6],B[1]);
ANDGATE andG3 (PPG8,A[6],B[2]);
ANDGATE andG4 (PPG9,A[6],B[3]);
ANDGATE andG5 (PPG10,A[6],B[4]);
ANDGATE andG6 (PPG11,A[6],B[5]);
ANDGATE andG7 (PPG12,A[6],B[6]);
ANDGATE andG8 (PPG13,A[6],B[7]);
ANDGATE andG9 (PPG14,A[6],B[8]);
ANDGATE andG10 (PPG15,A[6],B[9]);
ANDGATE andG11 (PPG16,A[7],B[9]);
ANDGATE andG12 (PPG17,A[8],B[9]);
ANDGATE andG13 (PPG18,A[9],B[9]);
ANDGATE andG14 (PPG19,A[10],B[9]);
ANDGATE andG15 (PPG20,A[11],B[9]);
ANDGATE andG16 (PPG21,A[12],B[9]);
ANDGATE andG17 (PPG22,A[13],B[9]);
ANDGATE andG18 (PPG23,A[14],B[9]);
ANDGATE andG19 (PPG24,A[15],B[9]);

ANDGATE andH1 (PPH7,A[7],B[0]);
ANDGATE andH2 (PPH8,A[7],B[1]);
ANDGATE andH3 (PPH9,A[7],B[2]);
ANDGATE andH4 (PPH10,A[7],B[3]);
ANDGATE andH5 (PPH11,A[7],B[4]);
ANDGATE andH6 (PPH12,A[7],B[5]);
ANDGATE andH7 (PPH13,A[7],B[6]);
ANDGATE andH8 (PPH14,A[7],B[7]);
ANDGATE andH9 (PPH15,A[7],B[8]);
```

```
ANDGATE andH10 (PPH16,A[8],B[8]);
ANDGATE andH11 (PPH17,A[9],B[8]);
ANDGATE andH12 (PPH18,A[10],B[8]);
ANDGATE andH13 (PPH19,A[11],B[8]);
ANDGATE andH14 (PPH20,A[12],B[8]);
ANDGATE andH15 (PPH21,A[13],B[8]);
ANDGATE andH16 (PPH22,A[14],B[8]);
ANDGATE andH17 (PPH23,A[15],B[8]);

ANDGATE andI1 (PPI8,A[8],B[0]);
ANDGATE andI2 (PPI9,A[8],B[1]);
ANDGATE andI3 (PPI10,A[8],B[2]);
ANDGATE andI4 (PPI11,A[8],B[3]);
ANDGATE andI5 (PPI12,A[8],B[4]);
ANDGATE andI6 (PPI13,A[8],B[5]);
ANDGATE andI7 (PPI14,A[8],B[6]);
ANDGATE andI8 (PPI15,A[8],B[7]);
ANDGATE andI9 (PPI16,A[9],B[7]);
ANDGATE andI10 (PPI17,A[10],B[7]);
ANDGATE andI11 (PPI18,A[11],B[7]);
ANDGATE andI12 (PPI19,A[12],B[7]);
ANDGATE andI13 (PPI20,A[13],B[7]);
ANDGATE andI14 (PPI21,A[14],B[7]);
ANDGATE andI15 (PPI22,A[15],B[7]);

ANDGATE andJ1 (PPJ9,A[9],B[0]);
ANDGATE andJ2 (PPJ10,A[9],B[1]);
ANDGATE andJ3 (PPJ11,A[9],B[2]);
ANDGATE andJ4 (PPJ12,A[9],B[3]);
ANDGATE andJ5 (PPJ13,A[9],B[4]);
ANDGATE andJ6 (PPJ14,A[9],B[5]);
ANDGATE andJ7 (PPJ15,A[9],B[6]);
ANDGATE andJ8 (PPJ16,A[10],B[6]);
ANDGATE andJ9 (PPJ17,A[11],B[6]);
ANDGATE andJ10 (PPJ18,A[12],B[6]);
ANDGATE andJ11 (PPJ19,A[13],B[6]);
ANDGATE andJ12 (PPJ20,A[14],B[6]);
ANDGATE andJ13 (PPJ21,A[15],B[6]);

ANDGATE andK1 (PPK10,A[10],B[0]);
ANDGATE andK2 (PPK11,A[10],B[1]);
ANDGATE andK3 (PPK12,A[10],B[2]);
ANDGATE andK4 (PPK13,A[10],B[3]);
ANDGATE andK5 (PPK14,A[10],B[4]);
ANDGATE andK6 (PPK15,A[10],B[5]);
ANDGATE andK7 (PPK16,A[11],B[5]);
ANDGATE andK8 (PPK17,A[12],B[5]);
ANDGATE andK9 (PPK18,A[13],B[5]);
ANDGATE andK10 (PPK19,A[14],B[5]);
ANDGATE andK11 (PPK20,A[15],B[5]);

ANDGATE andL1 (PPL11,A[11],B[0]);
ANDGATE andL2 (PPL12,A[11],B[1]);
ANDGATE andL3 (PPL13,A[11],B[2]);
ANDGATE andL4 (PPL14,A[11],B[3]);
ANDGATE andL5 (PPL15,A[11],B[4]);
ANDGATE andL6 (PPL16,A[12],B[4]);
ANDGATE andL7 (PPL17,A[13],B[4]);
ANDGATE andL8 (PPL18,A[14],B[4]);
```

```
   ANDGATE andL9 (PPL19,A[15],B[4]);

   ANDGATE andM1 (PPM12,A[12],B[0]);
   ANDGATE andM2 (PPM13,A[12],B[1]);
   ANDGATE andM3 (PPM14,A[12],B[2]);
   ANDGATE andM4 (PPM15,A[12],B[3]);
   ANDGATE andM5 (PPM16,A[13],B[3]);
   ANDGATE andM6 (PPM17,A[14],B[3]);
   ANDGATE andM7 (PPM18,A[15],B[3]);

   ANDGATE andN1 (PPN13,A[13],B[0]);
   ANDGATE andN2 (PPN14,A[13],B[1]);
   ANDGATE andN3 (PPN15,A[13],B[2]);
   ANDGATE andN4 (PPN16,A[14],B[2]);
   ANDGATE andN5 (PPN17,A[15],B[2]);

   ANDGATE andO1 (PPO14,A[14],B[0]);
   ANDGATE andO2 (PPO15,A[14],B[1]);
   ANDGATE andO3 (PPO16,A[15],B[1]);

   ANDGATE andP1 (PPP15,A[15],B[0]);


   // First reduction level, first stage

   wire
L1A8,L1A9,L1A10,L1A11,L1A12,L1A13,L1A14,L1A15,L1A16,L1A17,L1A18,L1A19,L1A20,
L1A21,L1A22;
   wire
PL1A8,PL1A9,PL1A10,PL1A11,PL1A12,PL1A13,PL1A14,PL1A15,PL1A16,PL1A17,PL1A18,PL1A1
9,PL1A20,PL1A21,PL1A22,PL1A23,PL1A24;
   wire
PL1B10,PL1B11,PL1B12,PL1B13,PL1B14,PL1B15,PL1B16,PL1B17,PL1B18,PL1B19,PL1B20,
PL1B21,PL1B22,PL1B23;

   HALFADDER  halfL1A8  (PPA8,PPB8,L1A8,PL1A8);
   COMPRESSOR compL1A9  (PPA9,PPB9,PPC9,PPD9,L1A8,L1A9,PL1A9,PL1B10);
   COMPRESSOR compL1A10 (PPA10,PPB10,PPC10,PPD10,L1A9,L1A10,PL1A10,PL1B11);
   COMPRESSOR compL1A11 (PPA11,PPB11,PPC11,PPD11,L1A10,L1A11,PL1A11,PL1B12);
   COMPRESSOR compL1A12 (PPA12,PPB12,PPC12,PPD12,L1A11,L1A12,PL1A12,PL1B13);
   COMPRESSOR compL1A13 (PPA13,PPB13,PPC13,PPD13,L1A12,L1A13,PL1A13,PL1B14);
   COMPRESSOR compL1A14 (PPA14,PPB14,PPC14,PPD14,L1A13,L1A14,PL1A14,PL1B15);
   COMPRESSOR compL1A15 (PPA15,PPB15,PPC15,PPD15,L1A14,L1A15,PL1A15,PL1B16);
   COMPRESSOR compL1A16 (PPA16,PPB16,PPC16,PPD16,L1A15,L1A16,PL1A16,PL1B17);
   COMPRESSOR compL1A17 (PPA17,PPB17,PPC17,PPD17,L1A16,L1A17,PL1A17,PL1B18);
   COMPRESSOR compL1A18 (PPA18,PPB18,PPC18,PPD18,L1A17,L1A18,PL1A18,PL1B19);
   COMPRESSOR compL1A19 (PPA19,PPB19,PPC19,PPD19,L1A18,L1A19,PL1A19,PL1B20);
   COMPRESSOR compL1A20 (PPA20,PPB20,PPC20,PPD20,L1A19,L1A20,PL1A20,PL1B21);
   COMPRESSOR compL1A21 (PPA21,PPB21,PPC21,PPD21,L1A20,L1A21,PL1A21,PL1B22);
   COMPRESSOR compL1A22 (PPA22,PPB22,PPC22,PPD22,L1A21,L1A22,PL1A22,PL1B23);
   FULLADDER  fullL1A23 (PPA23,PPB23,L1A22,PL1A23,PL1A24);


   // First reduction level, second stage

   wire
L1C10,L1C11,L1C12,L1C13,L1C14,L1C15,L1C16,L1C17,L1C18,L1C19,L1C20;
```

```
   wire
PL1C10,PL1C11,PL1C12,PL1C13,PL1C14,PL1C15,PL1C16,PL1C17,PL1C18,PL1C19,PL1C20,
PL1C21,PL1C22;
   wire
PL1D12,PL1D13,PL1D14,PL1D15,PL1D16,PL1D17,PL1D18,PL1D19,PL1D20,PL1D21;

   HALFADDER  halfL1C10 (PPE10,PPF10,L1C10,PL1C10);
   COMPRESSOR compL1C11 (PPE11,PPF11,PPG11,PPH11,L1C11,L1C10,PL1C11,PL1D12);
   COMPRESSOR compL1C12 (PPE12,PPF12,PPG12,PPH12,L1C11,L1C12,PL1C12,PL1D13);
   COMPRESSOR compL1C13 (PPE13,PPF13,PPG13,PPH13,L1C12,L1C13,PL1C13,PL1D14);
   COMPRESSOR compL1C14 (PPE14,PPF14,PPG14,PPH14,L1C13,L1C14,PL1C14,PL1D15);
   COMPRESSOR compL1C15 (PPE15,PPF15,PPG15,PPH15,L1C14,L1C15,PL1C15,PL1D16);
   COMPRESSOR compL1C16 (PPE16,PPF16,PPG16,PPH16,L1C15,L1C16,PL1C16,PL1D17);
   COMPRESSOR compL1C17 (PPE17,PPF17,PPG17,PPH17,L1C16,L1C17,PL1C17,PL1D18);
   COMPRESSOR compL1C18 (PPE18,PPF18,PPG18,PPH18,L1C17,L1C18,PL1C18,PL1D19);
   COMPRESSOR compL1C19 (PPE19,PPF19,PPG19,PPH19,L1C18,L1C19,PL1C19,PL1D20);
   COMPRESSOR compL1C20 (PPE20,PPF20,PPG20,PPH20,L1C19,L1C20,PL1C20,PL1D21);
   FULLADDER  fullL1C21 (PPE21,PPF21,L1C20,PL1C21,PL1C22);


   // First reduction level, third stage

   wire
L1E12,L1E13,L1E14,L1E15,L1E16,L1E17,L1E18;
   wire
PL1E12,PL1E13,PL1E14,PL1E15,PL1E16,PL1E17,PL1E18,PL1E19,PL1E20;
   wire
PL1F14,PL1F15,PL1F16,PL1F17,PL1F18,PL1F19;

   HALFADDER  halfL1E12 (PPI12,PPJ12,L1E12,PL1E12);
   COMPRESSOR compL1E13 (PPI13,PPJ13,PPK13,PPL13,L1E12,L1E13,PL1E13,PL1F14);
   COMPRESSOR compL1E14 (PPI14,PPJ14,PPK14,PPL14,L1E13,L1E14,PL1E14,PL1F15);
   COMPRESSOR compL1E15 (PPI15,PPJ15,PPK15,PPL15,L1E14,L1E15,PL1E15,PL1F16);
   COMPRESSOR compL1E16 (PPI16,PPJ16,PPK16,PPL16,L1E15,L1E16,PL1E16,PL1F17);
   COMPRESSOR compL1E17 (PPI17,PPJ17,PPK17,PPL17,L1E16,L1E17,PL1E17,PL1F18);
   COMPRESSOR compL1E18 (PPI18,PPJ18,PPK18,PPL18,L1E17,L1E18,PL1E18,PL1F19);
   FULLADDER  fullL1E19 (PPI19,PPJ19,L1E18,PL1E19,PL1E20);


   // First reduction level, fourth stage

   wire
L1G14,L1G15,L1G16,L1G17,L1G18;
   wire
PL1G14,PL1G15,PL1G16,PL1G17,PL1G18,PL1G19,PL1G20;
   wire
PL1H16,PL1H17;

   HALFADDER  halfL1G14 (PPM14,PPN14,L1G14,PL1G14);
   COMPRESSOR compL1G15 (PPM15,PPN15,PPO15,PPP15,L1G14,L1G15,PL1G15,PL1H16);
   COMPRESSOR compL1G16 (PPM16,PPN16,PPO16,GROUND,L1G15,L1G16,PL1G16,PL1H17);
   FULLADDER  fullL1G17 (PPM17,PPN17,L1G16,PL1G17,PL1G18);


   // Second reduction level, first stage

   wire
L2A4,L2A5,L2A6,L2A7,L2A8,L2A9,L2A10,L2A11,L2A12,L2A13,L2A14,L2A15,L2A16,L2A17,
L2A18,L2A19,L2A20,L2A21,L2A22,L2A23,L2A24,L2A25,L2A26;
```

```
   wire
PL2A4,PL2A5,PL2A6,PL2A7,PL2A8,PL2A9,PL2A10,PL2A11,PL2A12,PL2A13,PL2A14,PL2A15,
PL2A16,PL2A17,PL2A18,PL2A19,PL2A20,PL2A21,PL2A22,PL2A23,PL2A24,PL2A25,PL2A26,
PL2A27,PL2A28;
   wire
PL2B6,PL2B7,PL2B8,PL2B9,PL2B10,PL2B11,PL2B12,PL2B13,PL2B14,PL2B15,PL2B16,PL2B17,
PL2B18,PL2B19,PL2B20,PL2B21,PL2B22,PL2B23,PL2B24,PL2B25,PL2B26,PL2B27;

   HALFADDER  halfL2A4  (PPA4,PPB4,L2A4,PL2A4);
   COMPRESSOR compL2A5  (PPA5,PPB5,PPC5,PPD5,L2A4,L2A5,PL2A5,PL2B6);
   COMPRESSOR compL2A6  (PPA6,PPB6,PPC6,PPD6,L2A5,L2A6,PL2A6,PL2B7);
   COMPRESSOR compL2A7  (PPA7,PPB7,PPC7,PPD7,L2A6,L2A7,PL2A7,PL2B8);
   COMPRESSOR compL2A8  (PL1A8,PPC8,PPD8,PPE8,L2A7,L2A8,PL2A8,PL2B9);
   COMPRESSOR compL2A9  (PL1A9,PPE9,PPF9,PPG9,L2A8,L2A9,PL2A9,PL2B10);
   COMPRESSOR compL2A10 (PL1A10,PL1B10,PL1C10,PPG10,L2A9,L2A10,PL2A10,PL2B11);
   COMPRESSOR compL2A11 (PL1A11,PL1B11,PL1C11,PPI11,L2A10,L2A11,PL2A11,PL2B12);
   COMPRESSOR compL2A12 (PL1A12,PL1B12,PL1C12,PL1D12,L2A11,L2A12,PL2A12,PL2B13);
   COMPRESSOR compL2A13 (PL1A13,PL1B13,PL1C13,PL1D13,L2A12,L2A13,PL2A13,PL2B14);
   COMPRESSOR compL2A14 (PL1A14,PL1B14,PL1C14,PL1D14,L2A13,L2A14,PL2A14,PL2B15);
   COMPRESSOR compL2A15 (PL1A15,PL1B15,PL1C15,PL1D15,L2A14,L2A15,PL2A15,PL2B16);
   COMPRESSOR compL2A16 (PL1A16,PL1B16,PL1C16,PL1D16,L2A15,L2A16,PL2A16,PL2B17);
   COMPRESSOR compL2A17 (PL1A17,PL1B17,PL1C17,PL1D17,L2A16,L2A17,PL2A17,PL2B18);
   COMPRESSOR compL2A18 (PL1A18,PL1B18,PL1C18,PL1D18,L2A17,L2A18,PL2A18,PL2B19);
   COMPRESSOR compL2A19 (PL1A19,PL1B19,PL1C19,PL1D19,L2A18,L2A19,PL2A19,PL2B20);
   COMPRESSOR compL2A20 (PL1A20,PL1B20,PL1C20,PL1D20,L2A19,L2A20,PL2A20,PL2B21);
   COMPRESSOR compL2A21 (PL1A21,PL1B21,PL1C21,PL1D21,L2A20,L2A21,PL2A21,PL2B22);
   COMPRESSOR compL2A22 (PL1A22,PL1B22,PL1C22,PPE22,L2A21,L2A22,PL2A22,PL2B23);
   COMPRESSOR compL2A23 (PL1A23,PL1B23,PPC23,PPD23,L2A22,L2A23,PL2A23,PL2B24);
   COMPRESSOR compL2A24 (PL1A24,PPA24,PPB24,PPC24,L2A23,L2A24,PL2A24,PL2B25);
   COMPRESSOR compL2A25 (PPA25,PPB25,PPC25,PPD25,L2A24,L2A25,PL2A25,PL2B26);
   COMPRESSOR compL2A26 (PPA26,PPB26,PPC26,PPD26,L2A25,L2A26,PL2A26,PL2B27);
   FULLADDER  fullL2A27 (PPA27,PPB27,L2A26,PL2A27,PL2A28);


   // Second reduction level, second stage

   wire
L2C6,L2C7,L2C8,L2C9,L2C10,L2C11,L2C12,L2C13,L2C14,L2C15,L2C16,L2C17,L2C18,L2C19,
L2C20,L2C21,L2C22,L2C23,L2C24;
   wire
PL2C6,PL2C7,PL2C8,PL2C9,PL2C10,PL2C11,PL2C12,PL2C13,PL2C14,PL2C15,PL2C16,PL2C17,
PL2C18,PL2C19,PL2C20,PL2C21,PL2C22,PL2C23,PL2C24,PL2C25,PL2C26;
   wire
PL2D8,PL2D9,PL2D10,PL2D11,PL2D12,PL2D13,PL2D14,PL2D15,PL2D16,PL2D17,PL2D18,
PL2D19,PL2D20,PL2D21,PL2D22,PL2D23,PL2D24,PL2D25;

   HALFADDER  halfL2C6  (PPE6,PPF6,L2C6,PL2C6);
   COMPRESSOR compL2C7  (PPE7,PPF7,PPG7,PPH7,L2C6,L2C7,PL2C7,PL2D8);
   COMPRESSOR compL2C8  (PPF8,PPG8,PPH8,PPI8,L2C7,L2C8,PL2C8,PL2D9);
   COMPRESSOR compL2C9  (PPH9,PPI9,PPJ9,GROUND,L2C8,L2C9,PL2C9,PL2D10);
   COMPRESSOR compL2C10 (PPH10,PPI10,PPJ10,PPK10,L2C9,L2C10,PL2C10,PL2D11);
   COMPRESSOR compL2C11 (PPJ11,PPK11,PPL11,GROUND,L2C10,L2C11,PL2C11,PL2D12);
   COMPRESSOR compL2C12 (PL1E12,PPK12,PPL12,PPM12,L2C11,L2C12,PL2C12,PL2D13);
   COMPRESSOR compL2C13 (PL1E13,PPM13,PPN13,GROUND,L2C12,L2C13,PL2C13,PL2D14);
   COMPRESSOR compL2C14 (PL1A14,PL1F14,PL1G14,PPO14,L2C13,L2C14,PL2C14,PL2D15);
   COMPRESSOR compL2C15 (PL1E15,PL1F15,PL1G15,GROUND,L2C14,L2C15,PL2C15,PL2D16);
   COMPRESSOR compL2C16 (PL1E16,PL1F16,PL1G16,PL1H16,L2C15,L2C16,PL2C16,PL2D17);
   COMPRESSOR compL2C17 (PL1E17,PL1F17,PL1G17,PL1H17,L2C16,L2C17,PL2C17,PL2D18);
   COMPRESSOR compL2C18 (PL1E18,PL1F18,PL1G18,PPM18,L2C17,L2C18,PL2C18,PL2D19);
```

```
   COMPRESSOR compL2C19 (PL1E19,PL1F19,PPK19,PPL19,L2C18,L2C19,PL2C19,PL2D20);
   COMPRESSOR compL2C20 (PL1E20,PPI20,PPJ20,PPK20,L2C19,L2C20,PL2C20,PL2D21);
   COMPRESSOR compL2C21 (PPG21,PPH21,PPI21,PPJ21,L2C20,L2C21,PL2C21,PL2D22);
   COMPRESSOR compL2C22 (PPF22,PPG22,PPH22,PPI22,L2C21,L2C22,PL2C22,PL2D23);
   COMPRESSOR compL2C23 (PPE23,PPF23,PPG23,PPH23,L2C22,L2C23,PL2C23,PL2D24);
   COMPRESSOR compL2C24 (PPD24,PPE24,PPF24,PPG24,L2C23,L2C24,PL2C24,PL2D25);
   FULLADDER  fullL2C25 (PPE25,PPF25,L2C24,PL2C25,PL2C26);


   // Third reduction level, single stage

   wire
L3A2,L3A3,L3A4,L3A5,L3A6,L3A7,L3A8,L3A9,L3A10,L3A11,L3A12,L3A13,L3A14,L3A15,
L3A16,L3A17,L3A18,L3A19,L3A20,L3A21,L3A22,L3A23,L3A24,L3A25,L3A26,L3A27,L3A28;

   HALFADDER  halfL3A2  (PPA2,PPB2,L3A2,OUTA02);
   COMPRESSOR compL3A3  (PPA3,PPB3,PPC3,PPD3,L3A2,L3A3,OUTA03,OUTB04);
   COMPRESSOR compL3A4  (PL2A4,PPC4,PPD4,PPE4,L3A3,L3A4,OUTA04,OUTB05);
   COMPRESSOR compL3A5  (PL2A5,PPE5,PPF5,GROUND,L3A4,L3A5,OUTA05,OUTB06);
   COMPRESSOR compL3A6  (PL2A6,PL2B6,PL2C6,PPG6,L3A5,L3A6,OUTA06,OUTB07);
   COMPRESSOR compL3A7  (PL2A7,PL2B7,PL2C7,GROUND,L3A6,L3A7,OUTA07,OUTB08);
   COMPRESSOR compL3A8  (PL2A8,PL2B8,PL2C8,PL2D8,L3A7,L3A8,OUTA08,OUTB09);
   COMPRESSOR compL3A9  (PL2A9,PL2B9,PL2C9,PL2D9,L3A8,L3A9,OUTA09,OUTB10);
   COMPRESSOR compL3A10 (PL2A10,PL2B10,PL2C10,PL2D10,L3A9,L3A10,OUTA10,OUTB11);
   COMPRESSOR compL3A11 (PL2A11,PL2B11,PL2C11,PL2D11,L3A10,L3A11,OUTA11,OUTB12);
   COMPRESSOR compL3A12 (PL2A12,PL2B12,PL2C12,PL2D12,L3A11,L3A12,OUTA12,OUTB13);
   COMPRESSOR compL3A13 (PL2A13,PL2B13,PL2C13,PL2D13,L3A12,L3A13,OUTA13,OUTB14);
   COMPRESSOR compL3A14 (PL2A14,PL2B14,PL2C14,PL2D14,L3A13,L3A14,OUTA14,OUTB15);
   COMPRESSOR compL3A15 (PL2A15,PL2B15,PL2C15,PL2D15,L3A14,L3A15,OUTA15,OUTB16);
   COMPRESSOR compL3A16 (PL2A16,PL2B16,PL2C16,PL2D16,L3A15,L3A16,OUTA16,OUTB17);
   COMPRESSOR compL3A17 (PL2A17,PL2B17,PL2C17,PL2D17,L3A16,L3A17,OUTA17,OUTB18);
   COMPRESSOR compL3A18 (PL2A18,PL2B18,PL2C18,PL2D18,L3A17,L3A18,OUTA18,OUTB19);
   COMPRESSOR compL3A19 (PL2A19,PL2B19,PL2C19,PL2D19,L3A18,L3A19,OUTA19,OUTB20);
   COMPRESSOR compL3A20 (PL2A20,PL2B20,PL2C20,PL2D20,L3A19,L3A20,OUTA20,OUTB21);
   COMPRESSOR compL3A21 (PL2A21,PL2B21,PL2C21,PL2D21,L3A20,L3A21,OUTA21,OUTB22);
   COMPRESSOR compL3A22 (PL2A22,PL2B22,PL2C22,PL2D22,L3A21,L3A22,OUTA22,OUTB23);
   COMPRESSOR compL3A23 (PL2A23,PL2B23,PL2C23,PL2D23,L3A22,L3A23,OUTA23,OUTB24);
   COMPRESSOR compL3A24 (PL2A24,PL2B24,PL2C24,PL2D24,L3A23,L3A24,OUTA24,OUTB25);
   COMPRESSOR compL3A25 (PL2A25,PL2B25,PL2C25,PL2D25,L3A24,L3A25,OUTA25,OUTB26);
   COMPRESSOR compL3A26 (PL2A26,PL2B26,PL2C26,PPE26,L3A25,L3A26,OUTA26,OUTB27);
   COMPRESSOR compL3A27 (PL2A27,PL2B27,PPC27,PPD27,L3A26,L3A27,OUTA27,OUTB28);
   COMPRESSOR compL3A28 (PL2A28,PPA28,PPB28,PPC28,L3A27,L3A28,OUTA28,OUTB29);
   FULLADDER  fullL3A29 (PPA29,PPB29,L3A28,OUTA29,OUTA30);

endmodule // MAIN


module ANDGATE(OUT,A,B);

   input A,B;
   output OUT;

   assign OUT = A&B;

endmodule // ANDGATE


module HALFADDER(A,B,COUT,SUM);
```

```
   input A,B;
   output COUT,SUM;

   assign SUM = A^B;
   assign COUT = A&B;

endmodule // HALFADDER


module FULLADDER(A,B,CIN,COUT,SUM);

   input A,B,CIN;
   output COUT,SUM;

   assign SUM = A^B^CIN;
   assign COUT = A&B|A&CIN|B&CIN;

endmodule // FULLADDER


module COMPRESSOR(A,B,C,D,CIN,COUT,SUM,CARRY);

   input A,B,C,D,CIN;
   output COUT,SUM,CARRY;
   wire S;

 FULLADDER fa1 (A,B,C,COUT,S);
 FULLADDER fa2 (S,D,CIN,CARRY,SUM);

endmodule // COMPRESSOR
```

A.2 Multiplier Testbench

```
/*****************************************************************************

 The following code is for a tesbench used to verify the integrity of the
 32-bit multiplier module MAIN

*****************************************************************************/

module testloop16;

   wire [31:0] OUT;
   reg [15:0]  A;
   reg [15:0]  B;


MAIN bit16 (A,B,OUTA,OUTB);


// defining test variables

   reg [15:0] var1;
   reg [15:0] var2;
   reg [31:0] result, actual;
   reg [30:0] PRODA, PRODB;
```

```
   reg       CK;

// initializing test variables
initial

   begin
      CK = 0;
      var1 = 0;
      var2 = 1;
      PRODA = 0;
      PRODB = 0;

   end  //initial


// assigning test variables to module inputs
always@(posedge CK)
   begin
       A = var1;
       B = var2;
      PRODA = OUTA;
      PRODB = OUTB;
   end  //always

// main test loop including the error detection and variable incrementation
always@(negedge CK)
   begin

     actual = var1 * var2;
assign  result = PRODA+PRODB;

       $display ("Input a : %d", var1);
       $display ("Input b : %d", var2);
       $display ("Result : %d", result);
       if (result == actual) $display ("correct result");
       else
             begin
             $display ("ERROR IN CALCULATION!");
             $finish;
       end   //else
       $display (" ");
       $display (" ");

// variable incrementation
       var1= var1 + 7;
       var2= var2 + 56;

   end    //always block

//Cycle clock
always #5 CK = !CK;
always #25 $finish;

endmodule
```

A.3 32-bit CLA

```
/*******************************************************************************

  The following is code for a 32-bit 2-stage carry-lookahead-adder

*******************************************************************************/

module ADD32 (x, y, s, GND);

   input [30:0] x,y;
   output [31:0] s;
   input  GND;


   wire [30:0]  x,y;
   wire  gout,pout;
   wire [31:0]  s;
   wire  GND;
   wire [4:1]  c1;

   //First 1st-level 4-bit carry block
   wire [3:0]  cout1;
   wire  g1,p1;

   CARRY4 c4_0 (x[0],x[1],x[2],x[3],y[0],y[1],y[2],y[3],GND,cout1,g1,p1);
   XOR3 xor3_0 (x[0],y[0],GND,s[0]);
   XOR3 xor3_1 (x[1],y[1],cout1[0],s[1]);
   XOR3 xor3_2 (x[2],y[2],cout1[1],s[2]);
   XOR3 xor3_3 (x[3],y[3],cout1[2],s[3]);

   wire a;
   AND and_0    (GND,p1,a);
   OR or_0      (g1,a,c1[1]);

   //Second 1st-level 4-bit carry block
   wire [3:0]  cout2;
   wire  g2,p2;

   CARRY4 c4_1 (x[4],x[5],x[6],x[7],y[4],y[5],y[6],y[7],c1[1],cout2,g2,p2);
   XOR3 xor3_4 (x[4],y[4],cout1[3],s[4]);
   XOR3 xor3_5 (x[5],y[5],cout2[0],s[5]);
   XOR3 xor3_6 (x[6],y[6],cout2[1],s[6]);
   XOR3 xor3_7 (x[7],y[7],cout2[2],s[7]);

   wire b,c;
   AND and_1    (g1,p2,b);
   AND3 and3_1 (GND,p1,p2,c);
   OR3 or3_a    (g2,b,c,c1[2]);

   //Third 1st-level 4-bit carry block
   wire [3:0]  cout3;
   wire  g3,p3;

   CARRY4 c4_2 (x[8],x[9],x[10],x[11],y[8],y[9],y[10],y[11],c1[2],cout3,g3,p3);
   XOR3 xor3_8 (x[8],y[8],cout2[3],s[8]);
   XOR3 xor3_9 (x[9],y[9],cout3[0],s[9]);
```

```
   XOR3 xor3_10 (x[10],y[10],cout3[1],s[10]);
   XOR3 xor3_11 (x[11],y[11],cout3[2],s[11]);

   wire d,e,f;
   AND  and_2  (g2,p3,d);
   AND3 and3_2 (g1,p2,p3,e);
   AND4 and4_2 (GND,p1,p2,p3,f);
   OR4  or4_2   (g3,d,e,f,c1[3]);

   //Fourth 1st-level 4-bit carry block
   wire [3:0]  cout4;
   wire  g4,p4;

   CARRY4 c4_3
(x[12],x[13],x[14],x[15],y[12],y[13],y[14],y[15],c1[3],cout4,g4,p4);
   XOR3 xor3_12 (x[12],y[12],cout3[3],s[12]);
   XOR3 xor3_13 (x[13],y[13],cout4[0],s[13]);
   XOR3 xor3_14 (x[14],y[14],cout4[1],s[14]);
   XOR3 xor3_15 (x[15],y[15],cout4[2],s[15]);

   wire g,h,i,j;
   AND  and_3  (g3,p4,g);
   AND3 and3_3 (g2,p3,p4,h);
   AND4 and4_3 (g1,p2,p3,p4,i);
   AND5 and5_3 (GND,p1,p2,p3,p4,j);
   OR5  or5_3  (g4,g,h,i,j,c1[4]);

   //Second second-level 4-bit carry block

   wire [4:1]  c2;

   //Fifth 1st-level 4-bit carry block
   wire [3:0]  cout5;
   wire  g5,p5;

   CARRY4 c4_5
(x[16],x[17],x[18],x[19],y[16],y[17],y[18],y[19],c1[4],cout5,g5,p5);
   XOR3 xor3_16 (x[16],y[16],cout4[3],s[16]);
   XOR3 xor3_17 (x[17],y[17],cout5[0],s[17]);
   XOR3 xor3_18 (x[18],y[18],cout5[1],s[18]);
   XOR3 xor3_19 (x[19],y[19],cout5[2],s[19]);

   wire k;
   AND and_4   (c1[4],p5,k);
   OR or_4     (g5,k,c2[1]);

   //Sixth 1st-level 4-bit carry block
   wire [3:0]  cout6;
   wire  g6,p6;

   CARRY4 c4_6
(x[20],x[21],x[22],x[23],y[20],y[21],y[22],y[23],c2[1],cout6,g6,p6);
   XOR3 xor3_20 (x[20],y[20],cout5[3],s[20]);
   XOR3 xor3_21 (x[21],y[21],cout6[0],s[21]);
   XOR3 xor3_22 (x[22],y[22],cout6[1],s[22]);
   XOR3 xor3_23 (x[23],y[23],cout6[2],s[23]);

   wire l,m;
   AND and_5   (g5,p6,l);
```

```
   AND3 and3_5 (c1[4],p5,p6,m);
   OR3 or3_b   (g6,1,m,c2[2]);

   //Seventh first-level 4-bit carry block
   wire [3:0]  cout7;
   wire  g7,p7;

   CARRY4 c4_7
(x[24],x[25],x[26],x[27],y[24],y[25],y[26],y[27],c2[2],cout7,g7,p7);
   XOR3 xor3_24 (x[24],y[24],cout6[3],s[24]);
   XOR3 xor3_25 (x[25],y[25],cout7[0],s[25]);
   XOR3 xor3_26 (x[26],y[26],cout7[1],s[26]);
   XOR3 xor3_27 (x[27],y[27],cout7[2],s[27]);

   wire n,o,p;
   AND  and_6  (g6,p7,n);
   AND3 and3_6 (g5,p6,p7,o);
   AND4 and4_6 (c1[4],p5,p6,p7,p);
   OR4  or4_6   (g7,n,o,p,c2[3]);

   //Eighth first-level 4-bit carry block
   wire [3:0]  cout8;
   wire  g8,p8;

   CARRY4 c4_8 (x[28],x[29],x[30],GND,y[28],y[29],y[30],GND,c2[3],cout8,g8,p8);
   XOR3 xor3_28 (x[24],y[24],cout7[3],s[28]);
   XOR3 xor3_29 (x[25],y[25],cout8[0],s[29]);
   XOR3 xor3_30 (x[26],y[26],cout8[1],s[30]);
   XOR3 xor3_31 (x[27],y[27],cout8[2],s[31]);

   wire q,r,t,u;
   AND  and_7  (g3,p4,q);
   AND3 and3_7 (g2,p3,p4,r);
   AND4 and4_7 (g1,p2,p3,p4,t);
   AND5 and5_7 (c1[4],p1,p2,p3,p4,u);
   OR5  or5_7  (g8,q,r,t,u,c2[4]);

endmodule // ADD32

  //CARRY4 is used to generate all 4 carry bits as well as a group
  //carry-propogate/generate related to 4 input bits

module CARRY4(x0,x1,x2,x3,y0,y1,y2,y3,cin,cout,gout,pout);

   input        x0,x1,x2,x3,y0,y1,y2,y3;
   input        cin;
   output [3:0] cout;
   output       gout,pout;

   wire  x0,x1,x2,x3,y0,y1,y2,y3;
   wire         gout,pout;
   wire cin;
   wire [3:0] cout;
   wire  p0,p1,p2,p3,g0,g1,g2,g3;

   GEN_PG pg_0 (x0,y0,cin,p0,g0);
   GEN_PG pg_1 (x1,y1,cin,p1,g1);
   GEN_PG pg_2 (x2,y2,cin,p2,g2);
   GEN_PG pg_3 (x3,y3,cin,p3,g3);
```

```
    //generate cout1
    wire a;
    AND and_0   (cin,p0,a);
    OR or_0     (g0,a,cout[0]);

    //generate cout2
    wire b,c;
    AND and_1   (g0,p1,b);
    AND3 and3_1 (cin,p0,p1,c);
    OR3 or3_1   (g1,b,c,cout[1]);

    //generate cout3
    wire d,e,f;
    AND  and_2  (g1,p2,d);
    AND3 and3_2 (g0,p1,p2,e);
    AND4 and4_2 (cin,p0,p1,p2,f);
    OR4  or4_2   (g2,d,e,f,cout[2]);

    //generate cout4
    wire g,h,i,j;
    AND  and_3  (g2,p3,g);
    AND3 and3_3 (g1,p2,p3,h);
    AND4 and4_3 (g0,p1,p2,p3,i);
    AND5 and5_3 (cin,p0,p1,p2,p3,j);
    OR5  or5_3  (g3,g,h,i,j,cout[3]);

    //generate gout,pout
    OR4  or4_4 (g3,g,h,i,gout);
    AND4 and4_5 (p0,p1,p2,p3,pout);

endmodule  // CARRY4

  //Carry generate/propogate

module GEN_PG(a,b,p,g);

    input a,b;
    output g,p;

    wire a,b;
    wire g,p;

    XOR xor_1 (a,b,p);
    AND and_1 (a,b,g);

endmodule  // mux

module XOR(x,y,z);

    input x,y;
    output z;
    wire   x,y;
    wire   z;

  assign  z=x^y;

endmodule  //xor
```

```
module XOR3(w,x,y,z);

   input w,x,y;
   output z;
   wire   w,x,y;
   wire   z;

  assign  z=x^y^w;

endmodule  //xor3

module AND(x,y,z);

   input x,y;
   output z;
   wire   x,y;
   wire   z;

   assign z=x&y;

endmodule  //and

module AND3(w,x,y,z);

   input w,x,y;
   output z;
   wire   w,x,y;
   wire   z;

   assign z=w&x&y;

endmodule  // AND3

module AND4(v,w,x,y,z);

   input v,w,x,y;
   output z;
   wire  v,w,x,y;
   wire  z;

   assign z=v&w&x&y;

endmodule  // AND4

module AND5(u,v,w,x,y,z);

   input u,v,w,x,y;
   output z;
   wire  u,v,w,x,y;
   wire  z;

   assign z=u&v&w&x&y;

endmodule  // AND5

module OR(x,y,z);

   input x,y;
   output z;
```

```
   wire   x,y;
   wire   z;

endmodule  //OR

module OR3(w,x,y,z);

   input w,x,y;
   output z;
   wire   w,x,y;
   wire   z;

   assign z=w|x|y;

endmodule  // OR3

module OR4(v,w,x,y,z);

   input v,w,x,y;
   output z;
   wire  v,w,x,y;
   wire  z;

   assign z=v|w|x|y;

endmodule  // OR4

module OR5(u,v,w,x,y,z);

   input u,v,w,x,y;
   output z;
   wire  u,v,w,x,y;
   wire  z;

   assign z=u|v|w|x|y;

endmodule  // OR
```

A.4 CLA Test Bench

```
/*********************************************************************************

 The following code is for a tesbench used to verify the integrity of the
 32-bit CLA module ADD32

*********************************************************************************/


module testloop;

   wire [31:0] OUT;
   reg [30:0]  A;
   reg [30:0]  B;
```

```
   ADD32 (A,B,OUT);


// defining test variables

   reg [30:0] var1;
   reg [30:0] var2;
   reg [31:0] result, actual;
   reg        CK;


// initializing test variables
initial

   begin
      CK = 0;
      var1 = 0;
      var2 = 1;

   end  //initial


// assigning test variables to module inputs
always@(posedge CK)
   begin
       A = var1;
       B = var2;
  end  //always

// main test loop including the error detection and variable incrementation
always@(negedge CK)
   begin

      actual = var1 + var2;
assign  result = OUT;

       $display ("Input a : %d", var1);
       $display ("Input b : %d", var2);
       $display ("Result : %d", OUT);
       if (result == actual) $display ("correct result");
       else
              begin
              $display ("ERROR IN CALCULATION!");
              $finish;
       end   //else
       $display (" ");
       $display (" ");

// variable incrementation
       var1= var1 + 54857;
       var2= var2 + 9546;

   end    //always block

// Cycle clock
always #5 CK = !CK;
always #25 $finish;
```

```
endmodule
```

Appendix B

Simulation Waveforms

B.1 4:2 Compressor Waveforms

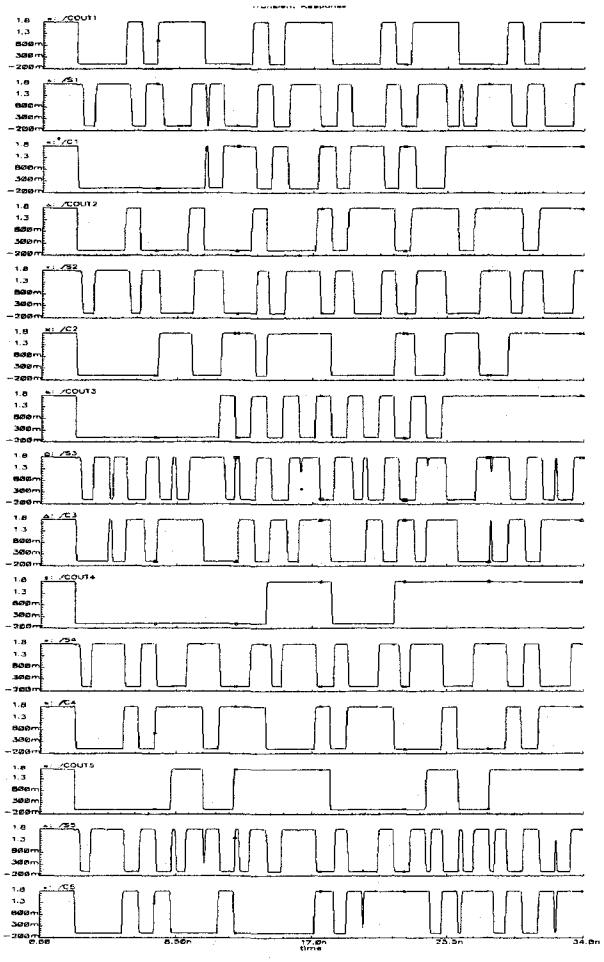


Figure B.1 CMOS1 Output

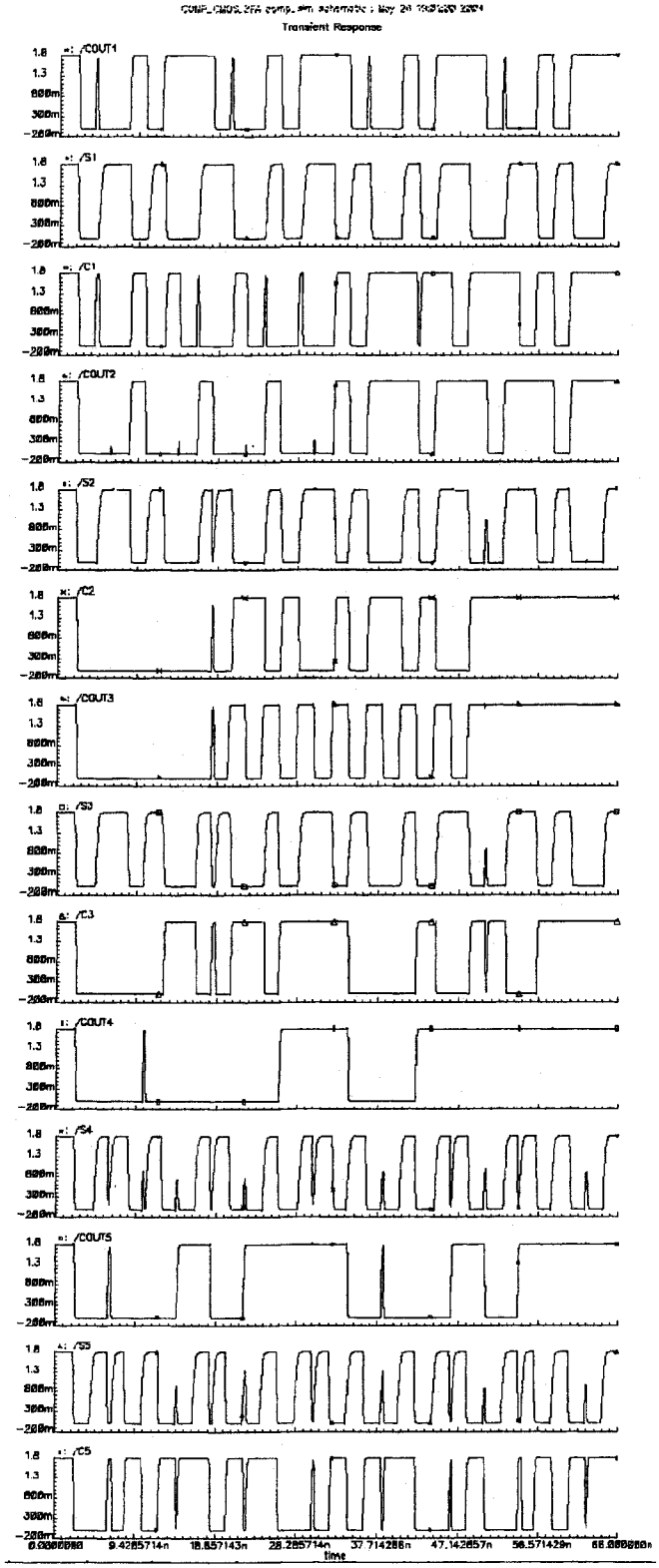


Figure B.2 CMOS2 Output

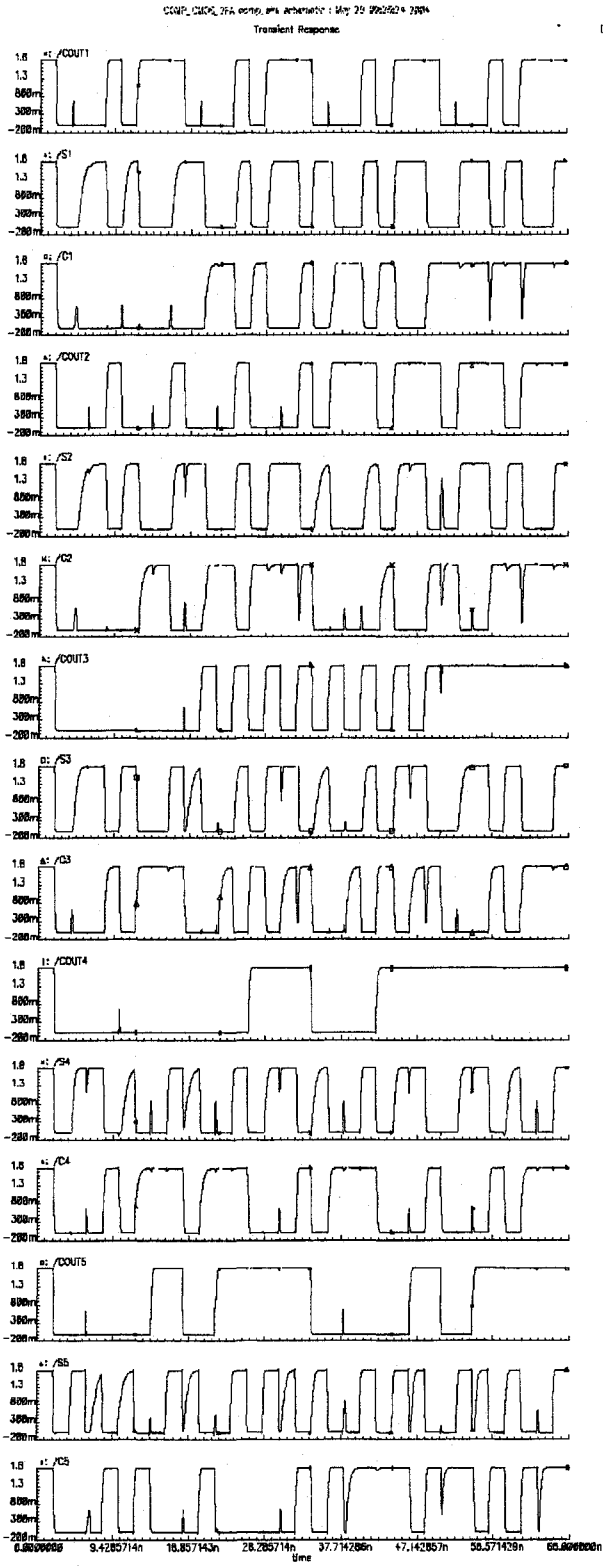


Figure B.3 TGATE Output

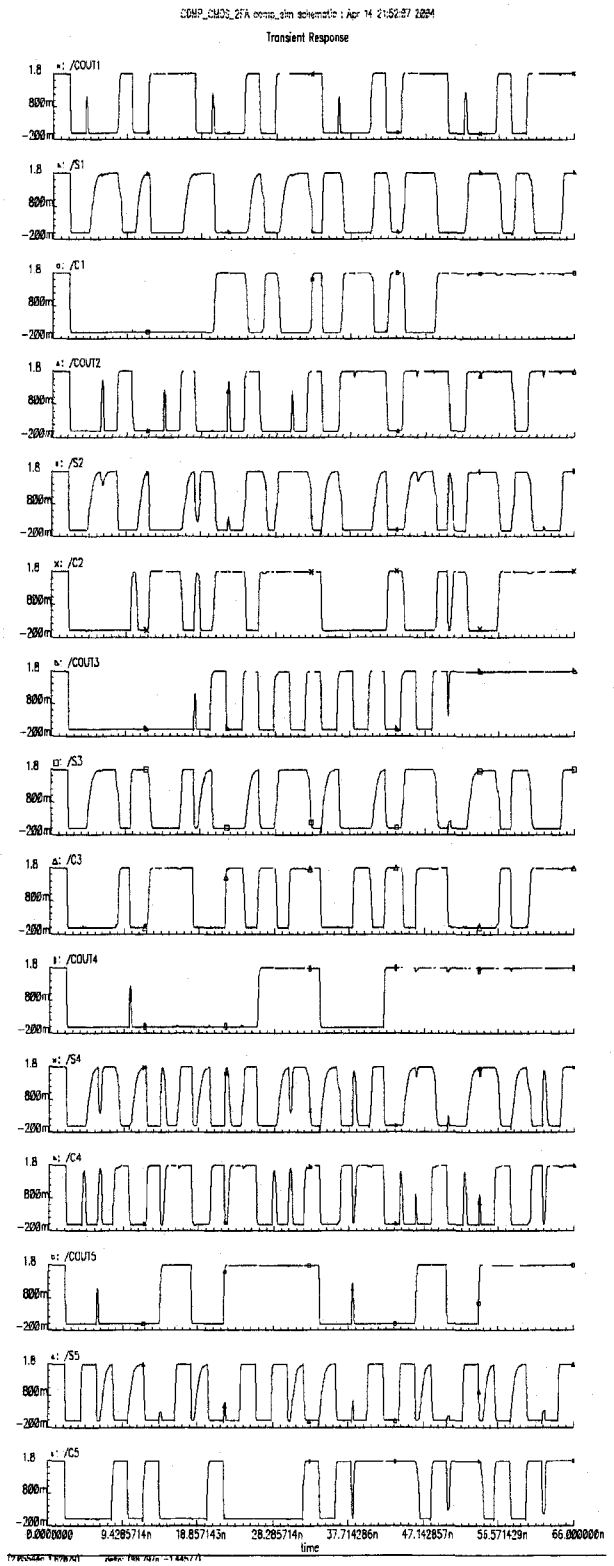


Figure B.4 PASS1 Output

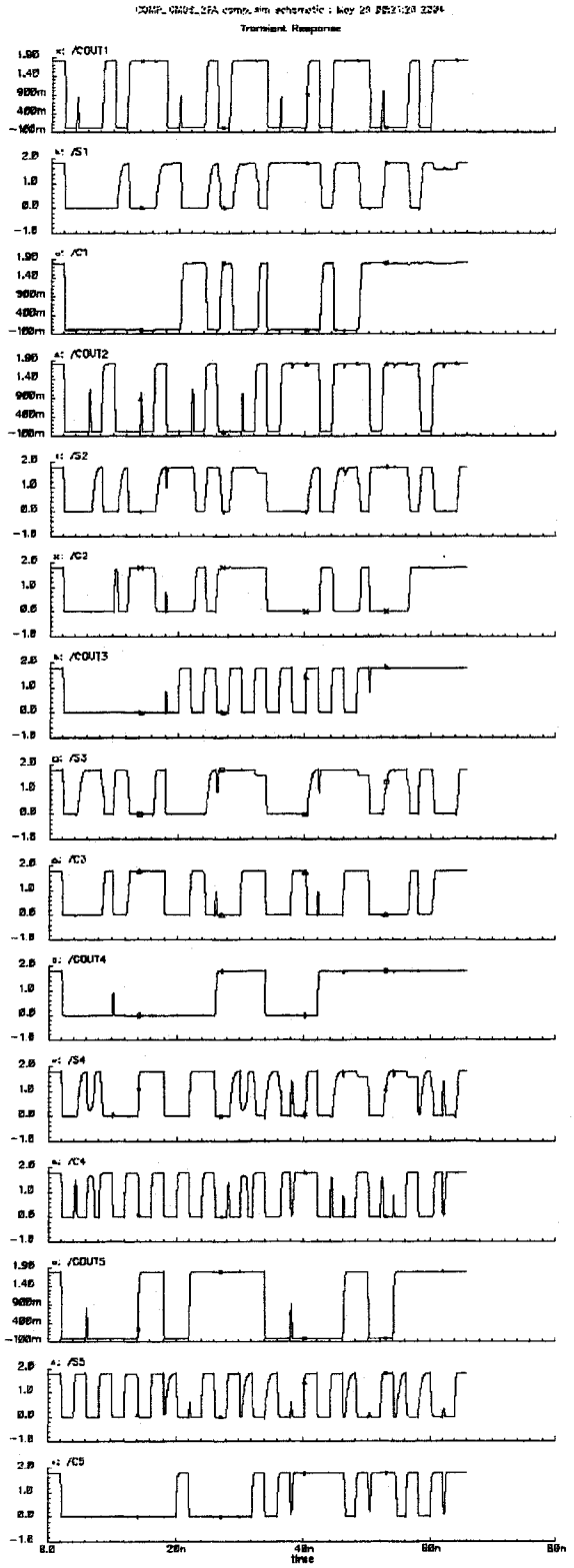


Figure B.5 PASS2 Output

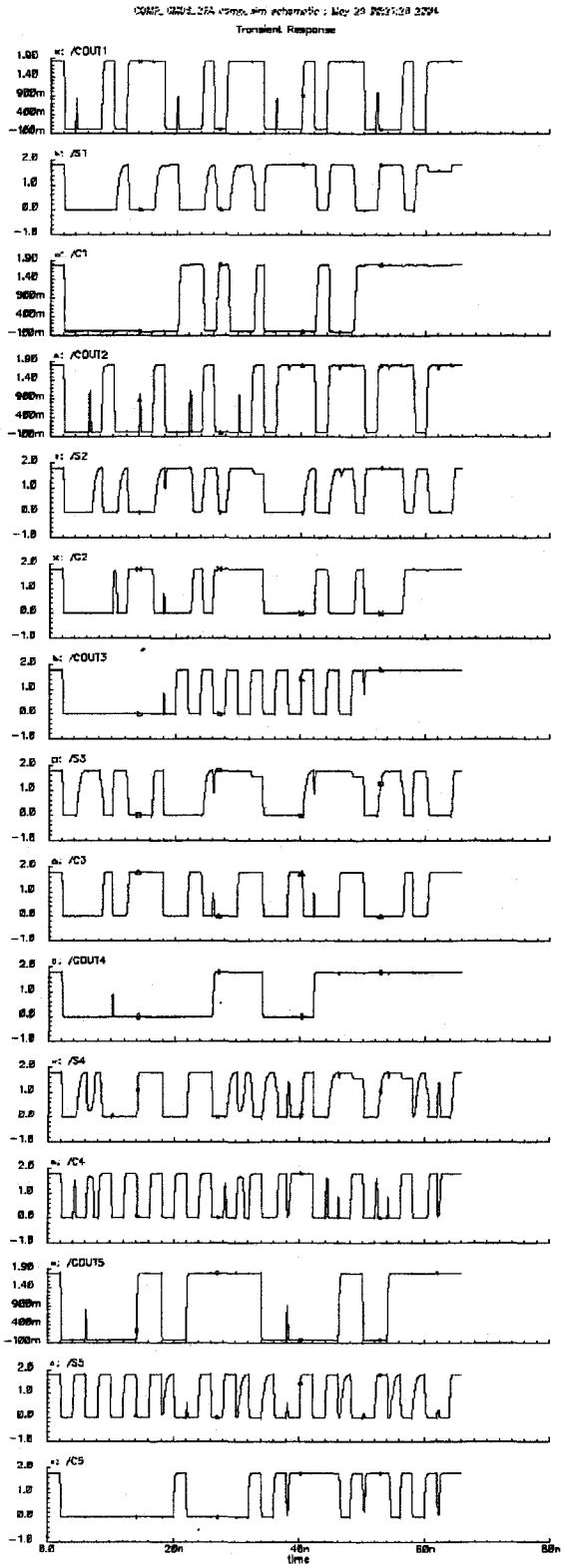


Figure B.6 PASS3 Output

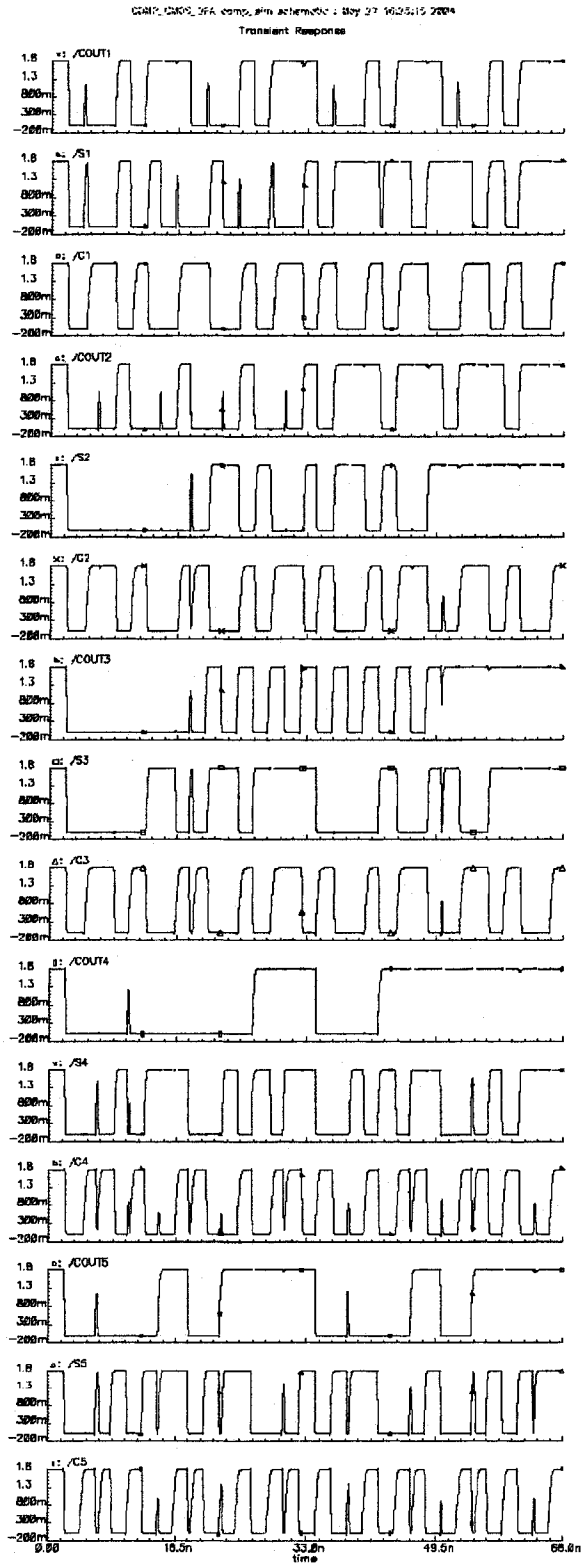


Figure B.7 HYBRID1 Output

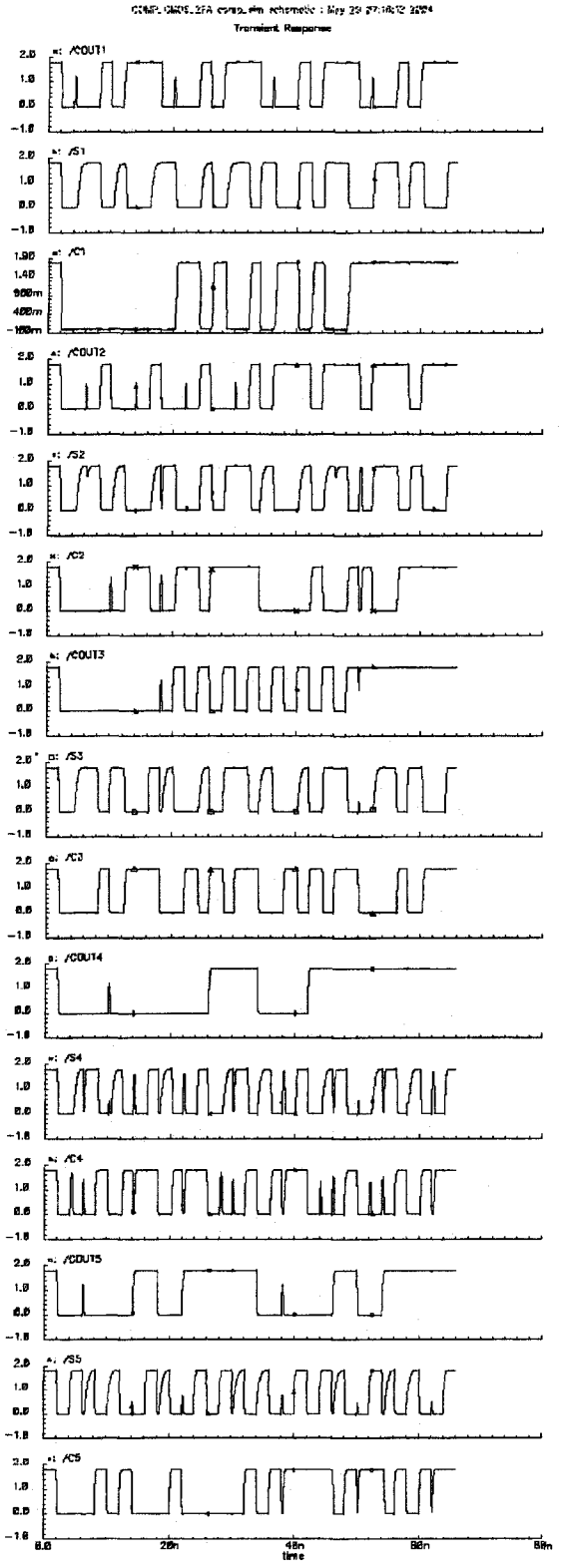


Figure B.8 HYBRID2 Output

B.2 Multiplier Waveforms

B.2.1 Input Waveforms

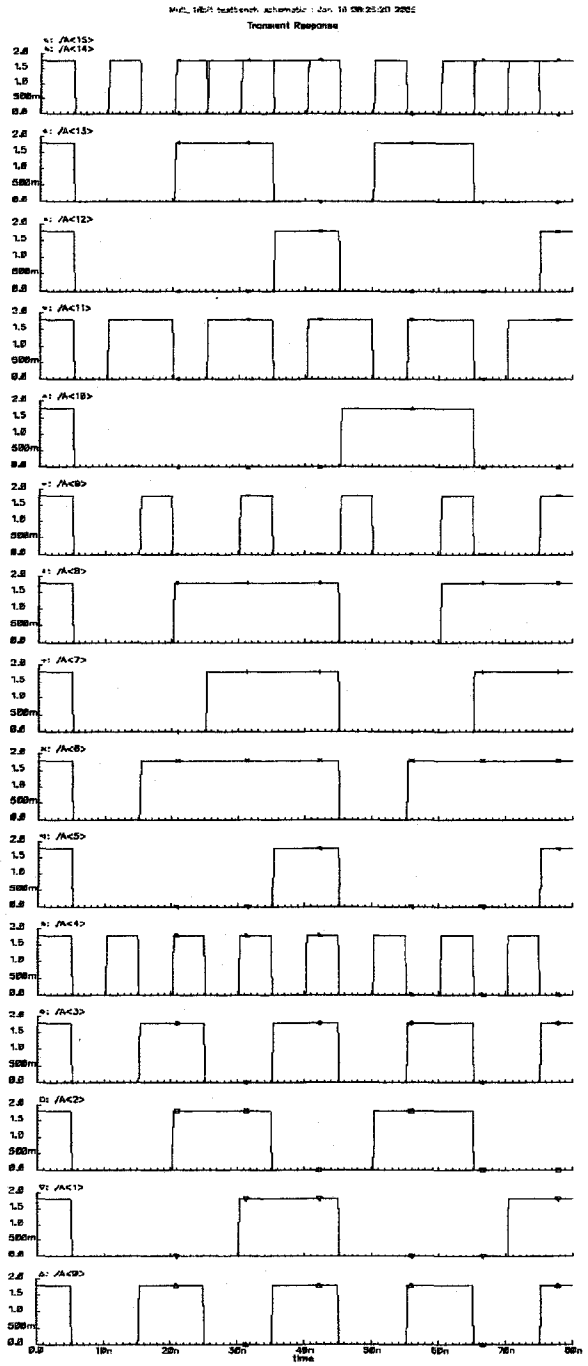


Figure B.9 Multiplicand A<15:0>

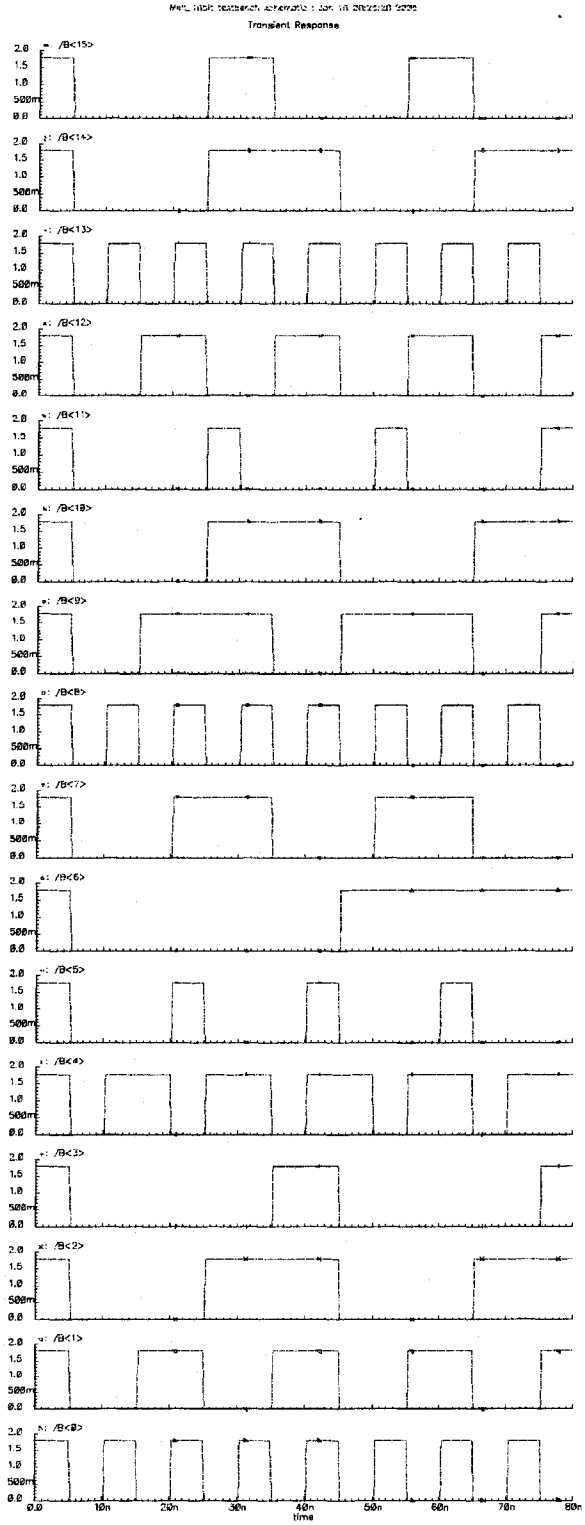


Figure B.10 Multiplier B<15:0>

B.2.2 Output Waveforms for CMOS1 Multiplier

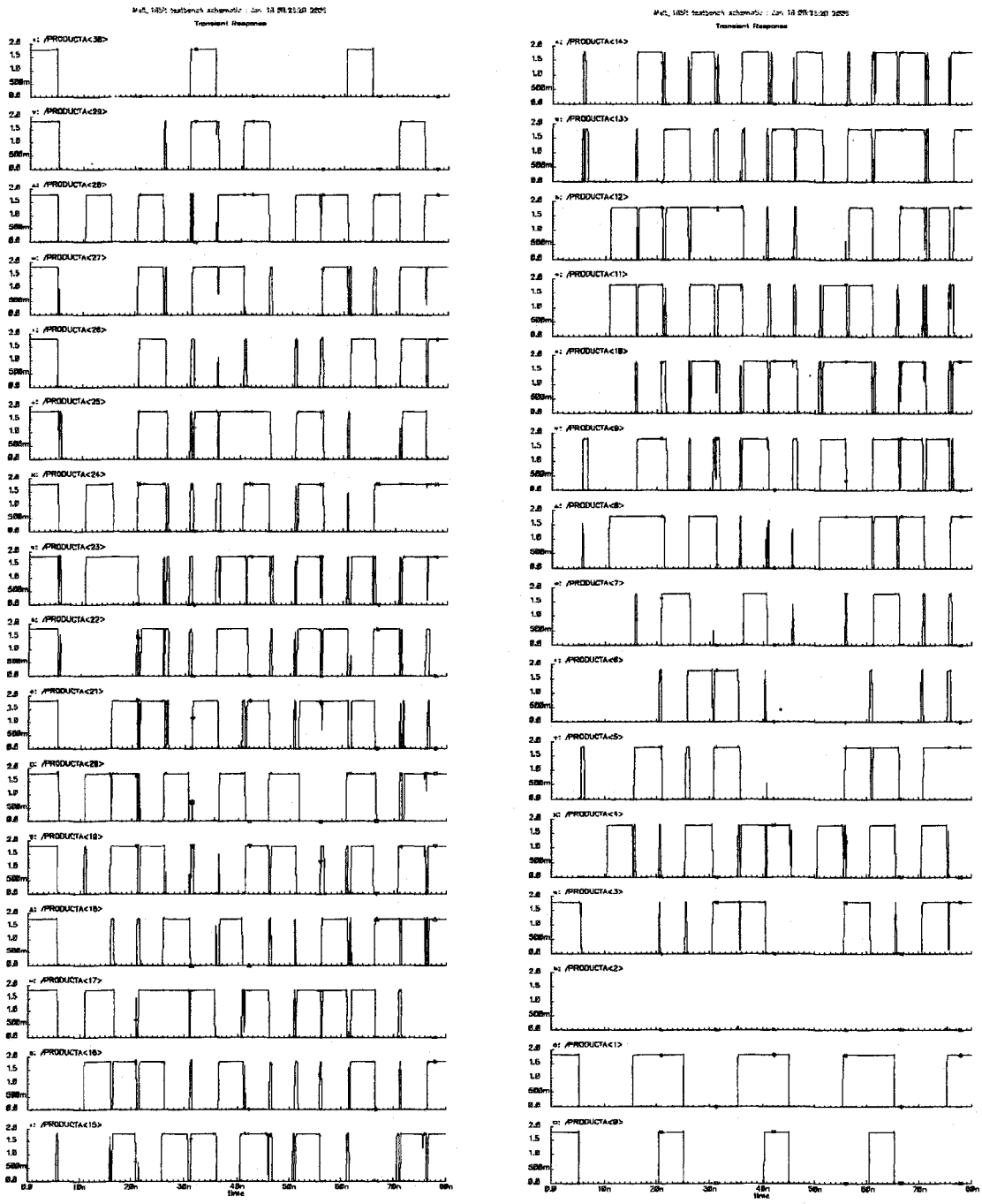


Figure B.11 31-Bit Partial Product A (a) PPA<30:15> (b) PPA<14:0>

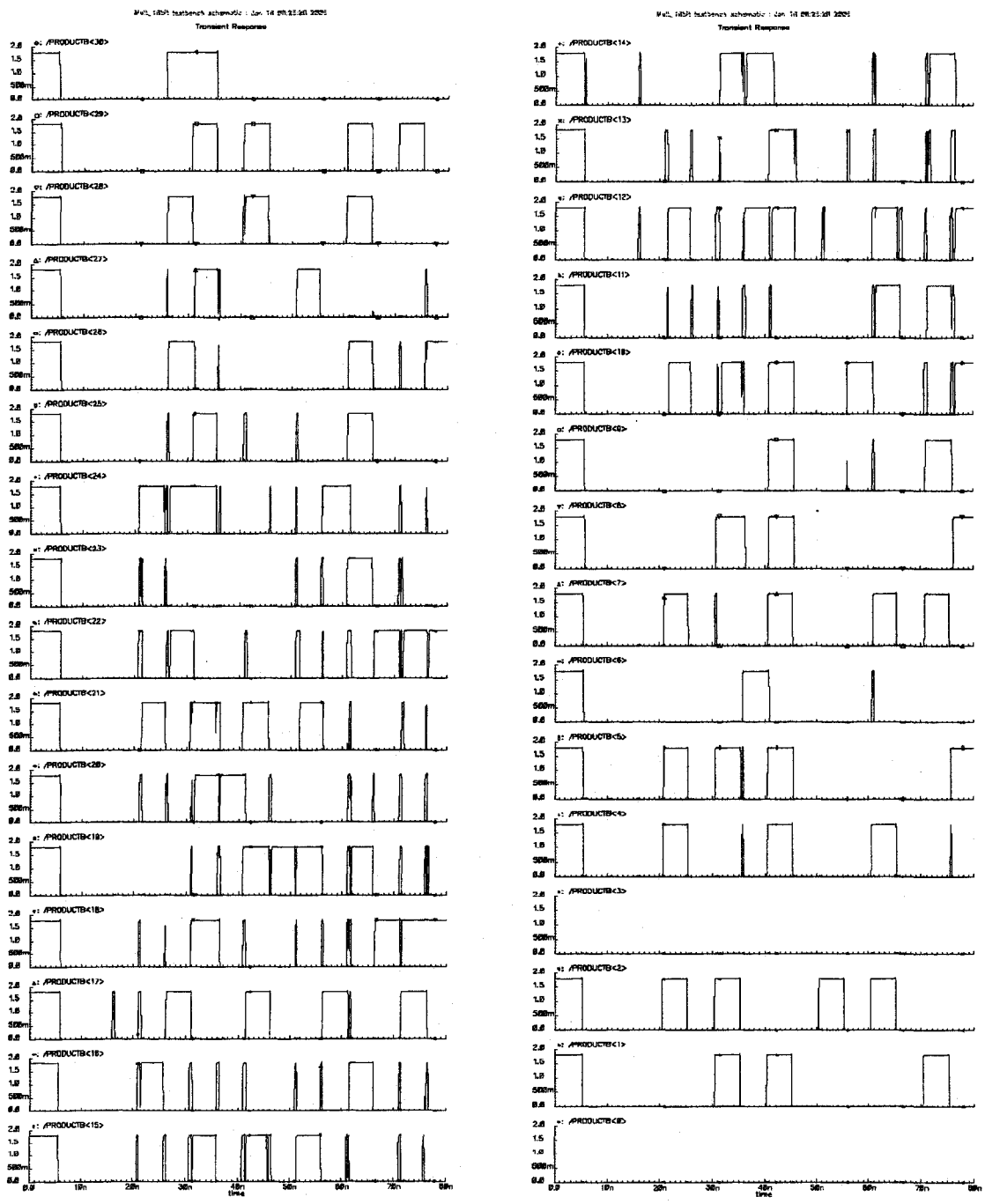


Figure B.12 31-Bit Partial Product B (a) PPB<30:15> (b) PPB<14:0>

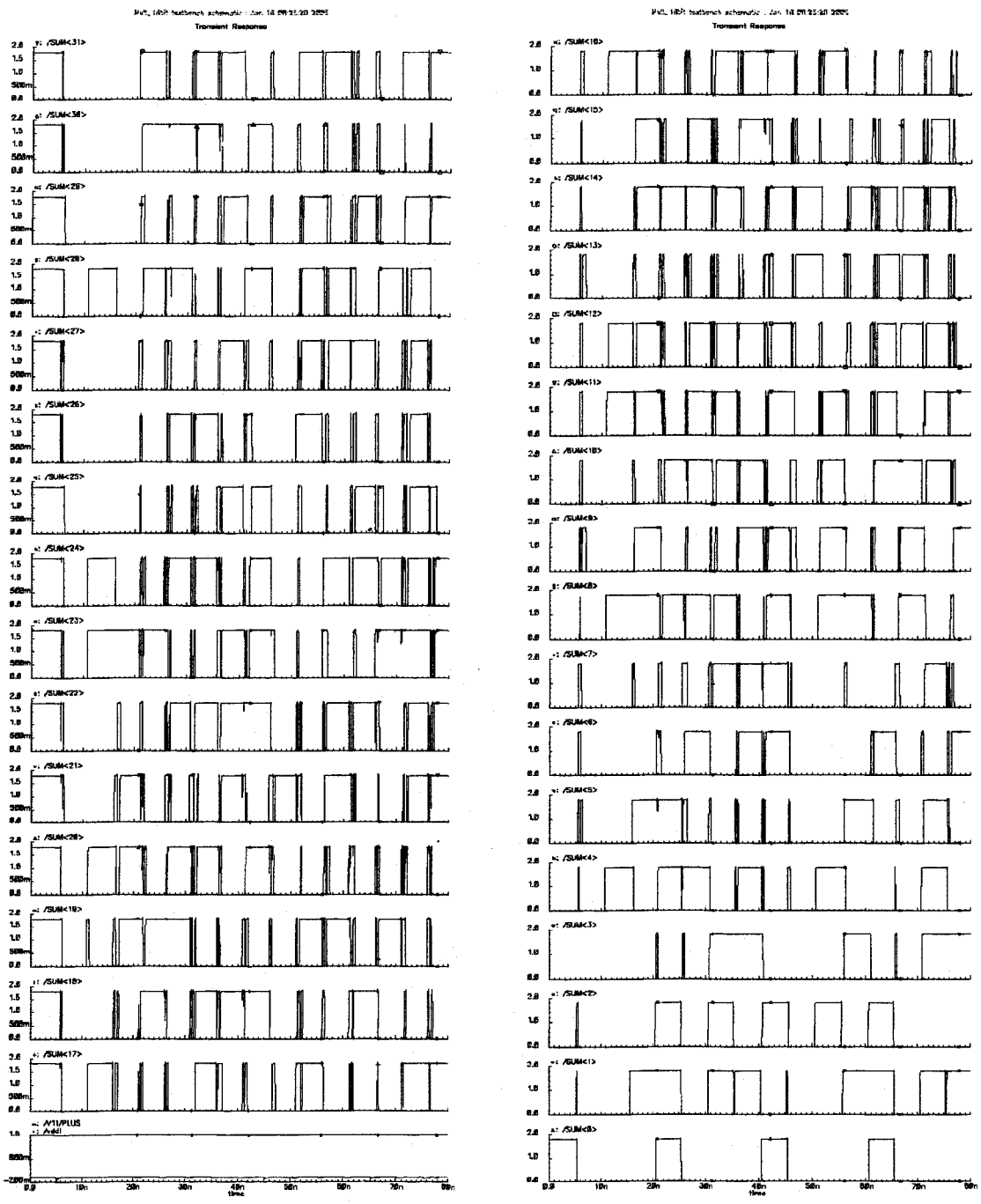


Figure B.13 CMOS1 Multiplier Product (a) Product <31:17> (b) Product <16:0>

B.2.3 Output Waveforms for PASS2 Multiplier

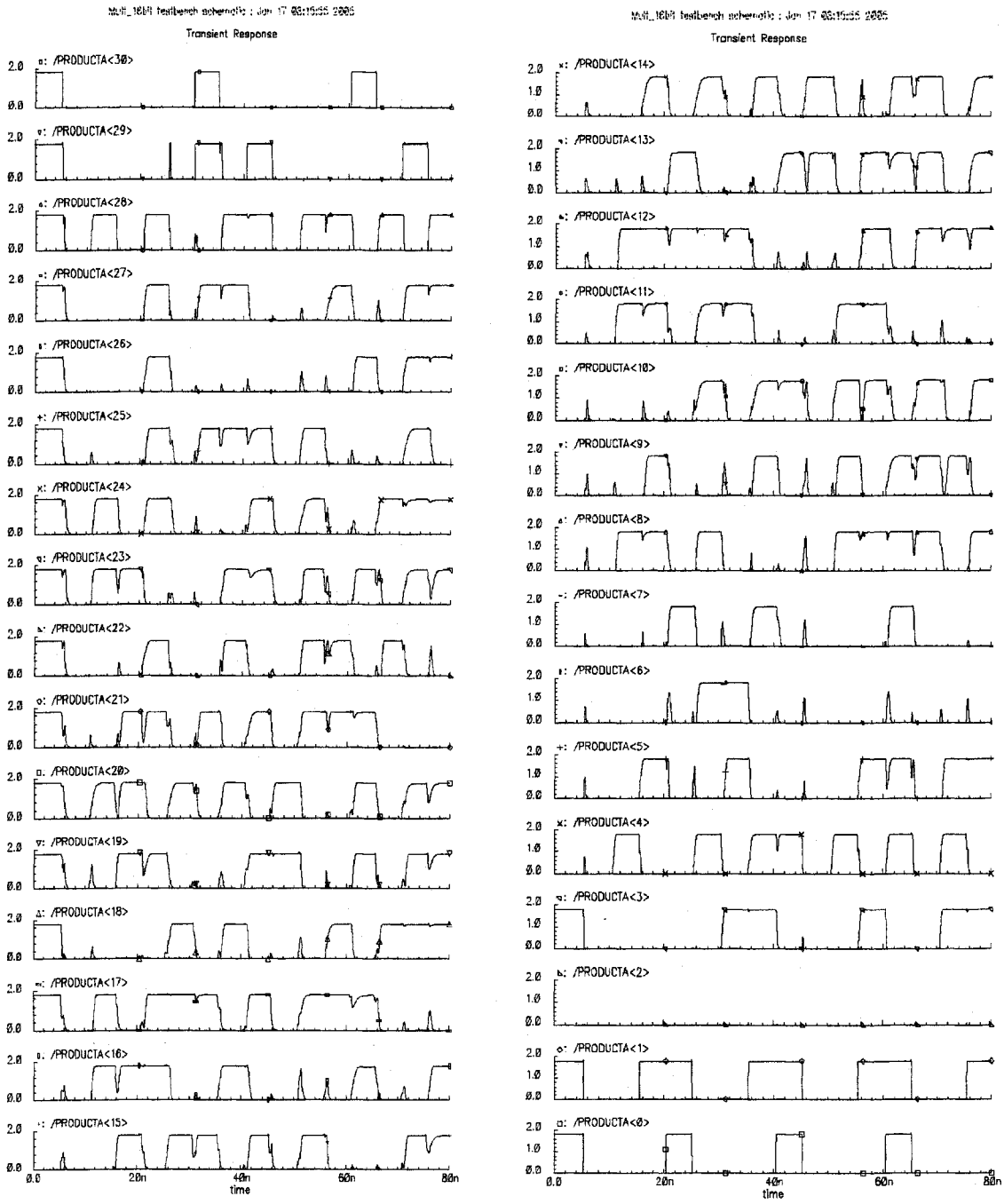


Figure B.14 31-Bit Partial Product A (a) PPA<30:15> (b) PPA<14:0>

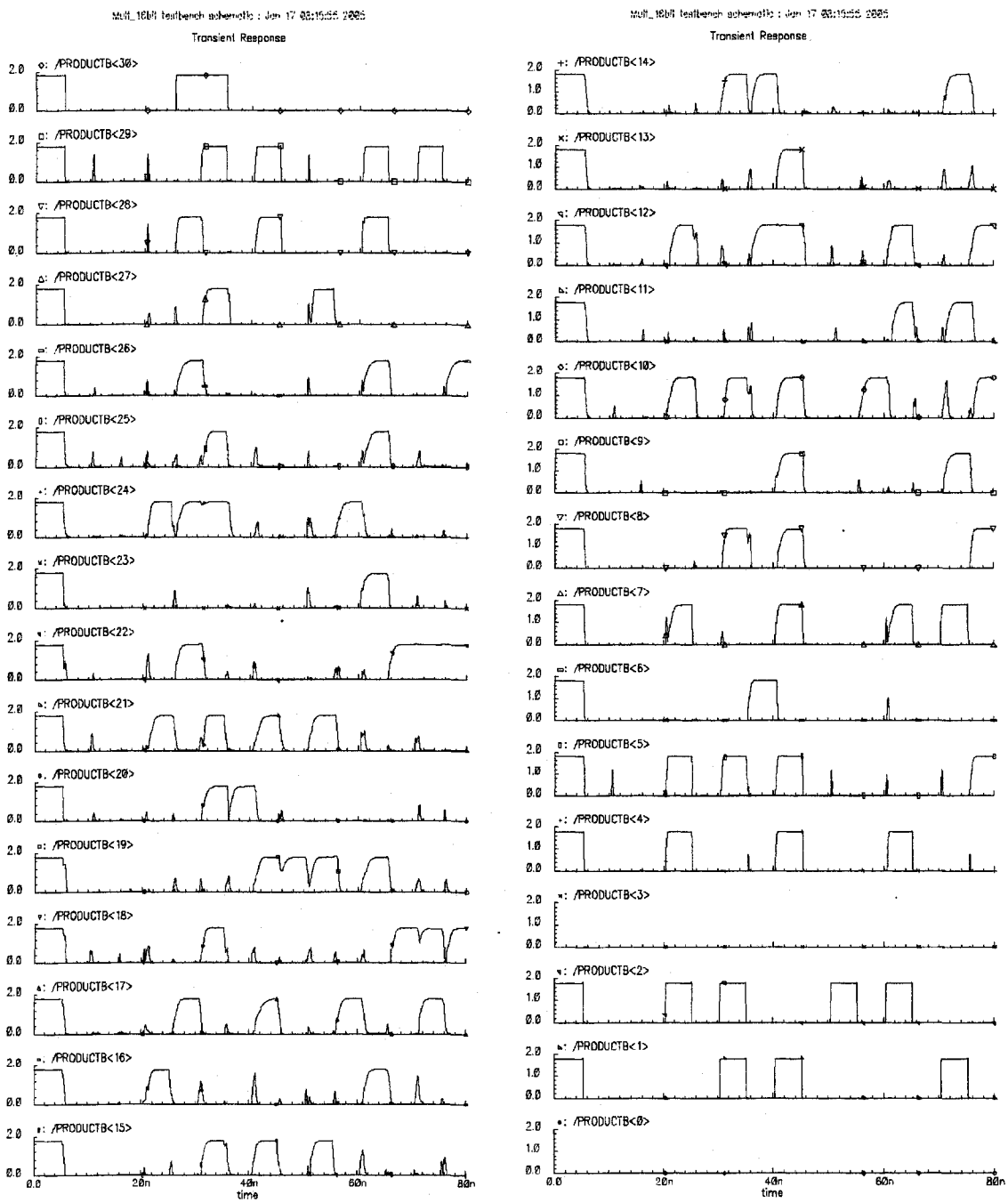


Figure B.15 31-Bit Partial Product B (a) PPB<30:15> (b) PPB<14:0>

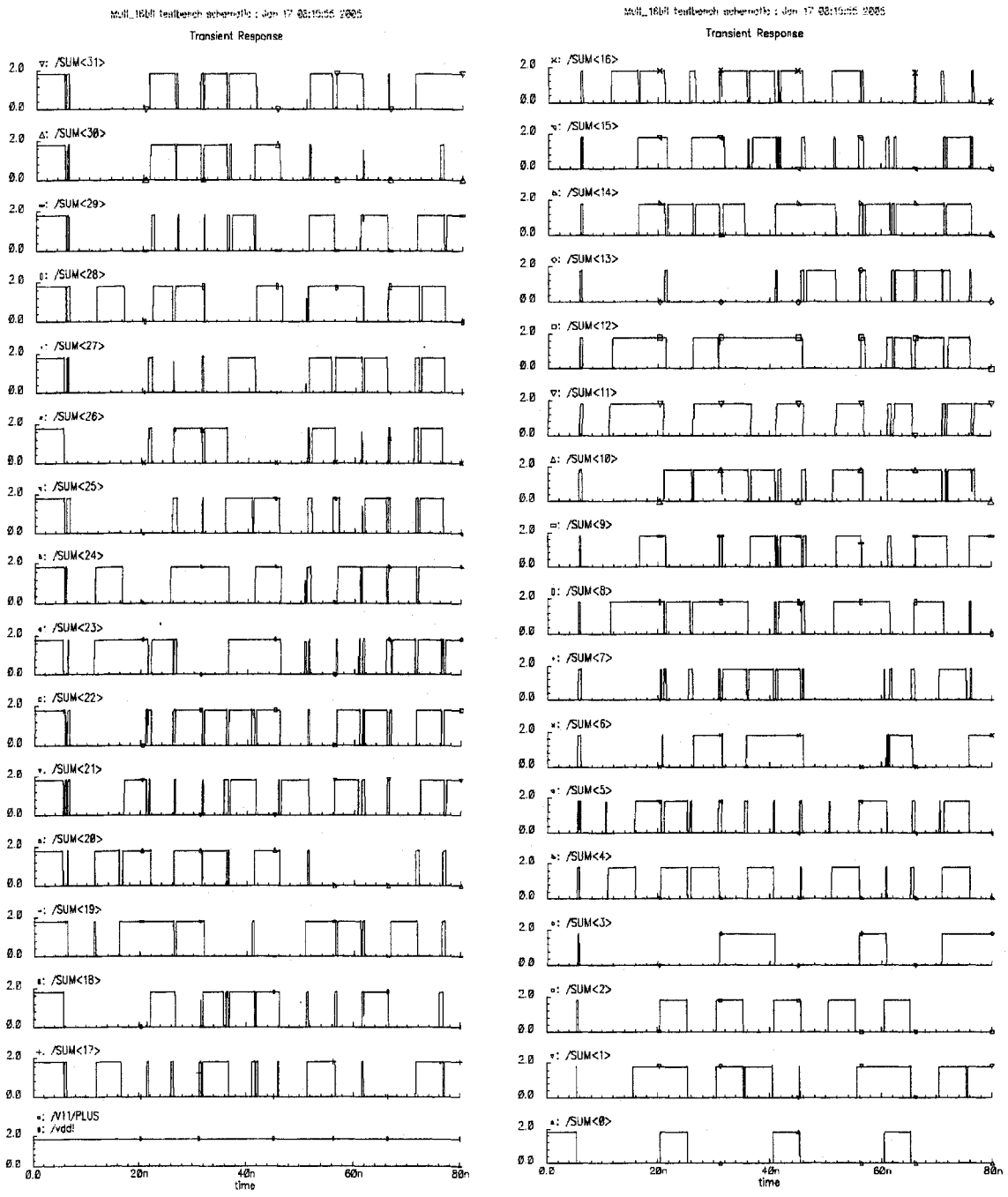


Figure B.16 PASS2 Multiplier Product (a) Product <31:17> (b) Product <16:0>

Appendix C

Hardware Test Code

C.1 IMS Screens Test Code

```
Set All #TXT
Rem ID Test Station ATS-200/128K V6.4d   Options: 1 Hard disk
Rem [1] ""
Rem [2] ""
Init All
Clk Internal,100ns
Event 0=User0,Off
Event 1=User1,Off
Event 2=STM-2,Off
Event 3=Error,On
Socket "DIP Auto", 80, "DIP Auto"
Config 0,"ATS Control","128K","Blazer System Controller"
Config 5,"ATS Data 200","128K","16 Bidirectional Channels"
Config 6,"ATS Data 200","128K","16 Bidirectional Channels"
Config 7,"ATS Data 200","128K","16 Bidirectional Channels"
Config 8,"ATS Timing","","System Timing, 1 Power Supply"
Config 16,"Slave Control","128K","Blazer Slave Controller"
Config 24,"ATS Timing","","System Timing, 1 Power Supply"
Config 31,"ATS PMU","","Analytical DC PMU"

Testconditions #TXT
 Start All
 Programmable Loads Enabled
 Last Vector Disable
 Branch Vector Maintain
 PMU Off
 Pins Tested All
 Vector PMU Maintain
 Stop PMU Disabled
 Autoranging Enabled
```

```
  Nominal Accuracy 0.1
Testconditions End

Resource INPUT=Force #TXT
  78, OS1, 7A2
  79, OS0, 7A0
  63, ADV, 5A0
  68, INB, 6A5
  69, INA, 6A3
Resource End
Radix INPUT=Bin
Polarity INPUT=Positive
Format INPUT=NRZ,Timing,11,Reset
Drive INPUT=0mV,3.00V,9

Calibrate Offset TDR INPUT #TXT
  78, 3.460ns, 0ns, 0ns
  79, 3.460ns, 0ns, 0ns
  63, 3.460ns, 0ns, 0ns
  68, 3.460ns, 0ns, 0ns
  69, 3.460ns, 0ns, 0ns
Calibrate End

Resource CLK=Force #TXT
  72, CLK1, 6A1
Resource End
Radix CLK=Bin
Polarity CLK=Positive
Format CLK=RZ,10.00ns,20.00ns,Timing,1,Reset
Drive CLK=0mV,3.00V,9

Calibrate Offset TDR CLK #TXT
  72, 3.460ns, 0ns, 0ns
Calibrate End

Resource OUTPUT=Compare #TXT
  2, O7, 7B6
  3, O6, 7B4
  8, O5, 7B1
  9, O4, 6B7
  11, 03, 6B4
  12, 02, 6B5
  18, O1, 7B7
  62, O0, 5A2
Resource End
Radix OUTPUT=Bin
Polarity OUTPUT=Positive
```

Format OUTPUT=Edge,50.00ns,Timing,0,Reset
Threshold OUTPUT=1.50V,1.50V
Terminate OUTPUT=Off
Active OUTPUT=Off,0nA,0nA,900mV
Rlleakage OUTPUT=Off

Calibrate Offset TDR OUTPUT #TXT
2, 3.460ns, 0ns, 0ns
3, 3.460ns, 0ns, 0ns
8, 3.460ns, 0ns, 0ns
9, 3.460ns, 0ns, 0ns
11, 3.460ns, 0ns, 0ns
12, 3.460ns, 0ns, 0ns
18, 3.460ns, 0ns, 0ns
62, 3.460ns, 0ns, 0ns
Calibrate End

Resource GND=Power,GND #TXT
1, CVSS1, 7B7
10, CVSS2, 6B6
19, RVSS1, 5B4
61, CVSS3, 5A3
73, RVSS2, 7A7
Resource End
Power GND=0V,,,GND

Resource RVDD=Power,VA #TXT
13, RVDD1, 6B3
70, RVDD2, 6A2
Resource End
Power RVDD=3.000V,200mA,0ms,LOZ

Equation Recomputation Manual
Equation End

Equation Recomputation Auto

Waveform Display #TXT
Force,INPUT,INA
Force,INPUT,OS0
Force,INPUT,OS1
Force,CLK,CLK1
Expect,OUTPUT,O7
Acquire,OUTPUT,O7
Expect,OUTPUT,O6

```
  Acquire,OUTPUT,O6
  Expect,OUTPUT,O5
  Acquire,OUTPUT,O5
  Expect,OUTPUT,O4
  Acquire,OUTPUT,O4
  Expect,OUTPUT,03
  Acquire,OUTPUT,03
  Expect,OUTPUT,02
  Acquire,OUTPUT,02
  Expect,OUTPUT,O1
  Acquire,OUTPUT,O1
  Expect,OUTPUT,O0
  Acquire,OUTPUT,O0
Waveform End
Waveform Magnify=10
Waveform Markers Sequence="Off","Off"

Show "All"
Display Compare="Expect and Acquire"
Fail 65536
Mask "11111111"
Column Error "11111111"
Acp Setup_Hold Binary #TXT
Acp Prop_Delay Binary #TXT
Acp End
Fail Acp Setup =0,Hold = 0,Prop = 0,Analysis = 0


PAR End
PAR GLT #TXT
  Connect=
  Open=
PAR End

DCP INPUT, None
DCC INPUT, None
DCResistance INPUT, 0m

DCP CLK, None
DCC CLK, None
DCResistance CLK, 0m

DCP OUTPUT, None
DCC OUTPUT, None
DCResistance OUTPUT, 0m
```

```
MASK DCP NONE
Mem 0 #TXT
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10000 1 00000000
10100 1 00000000
...
```

Several thousand more inputs follow...

```
...
10010 1 10101001:     Goto 1
Mem End
Set End
```

C.2 Test Results

The following are the least significant *8-bit* test results obtained for the first 34 input vectors:

```
Set SEQ #TXT
Rem ID Test Station ATS-200/128K V6.4d   Options: 1 Hard disk
Rem [1] ""
Rem [2] ""
Init Memory
Seq  0,All,"00000 1     00000000    00001000","  e"
Seq  1,All,"00000 1     00000000    00001000","  e"
Seq  2,All,"00000 1     00000000    00001000","  e"
Seq  3,All,"00000 1     00000000    00001000","  e"
Seq  4,All,"00000 1     00000000    00001000","  e"
Seq  5,All,"00000 1     00000000    00001000","  e"
Seq  6,All,"00000 1     00000000    00001000","  e"
Seq  7,All,"00000 1     00000000    00001000","  e"
Seq  8,All,"00000 1     00000000    00001000","  e"
Seq  9,All,"00000 1     00000000    00001000","  e"
Seq 10,All,"00000 1     00000000    00001000","  e"
Seq 11,All,"00000 1     00000000    00001000","  e"
Seq 12,All,"00000 1     00000000    00001000","  e"
Seq 13,All,"00000 1     00000000    00001000","  e"
Seq 14,All,"00000 1     00000000    00001000","  e"
Seq 15,All,"00100 1     00000000    00001000","  e"
Seq 16,All,"00011 1     00000000    00001000","  e"
Seq 17,All,"00010 1     00000000    00001000","  e"
Seq 18,All,"00000 1     00000000    00001000","  e"
Seq 19,All,"00001 1     00000000    00001000","  e"
Seq 20,All,"00010 1     00000000    00001000","  e"
Seq 21,All,"00010 1     00000000    00001000","  e"
Seq 22,All,"00001 1     00000000    00001000","  e"
Seq 23,All,"00011 1     00000000    00001000","  e"
Seq 24,All,"00000 1     00000000    00001000","  e"
Seq 25,All,"00000 1     00000000    00001000","  e"
Seq 26,All,"00000 1     00000000    00001000","  e"
Seq 27,All,"00000 1     00000000    00001000","  e"
Seq 28,All,"00000 1     00000000    00001000","  e"
Seq 29,All,"00000 1     00000000    00001000","  e"
Seq 30,All,"00000 1     00000000    00001000","  e"
Seq 31,All,"00100 1     10001011    00101000","  e"
Seq 32,All,"00000 1     10001011    00101000","  e"
Seq 33,All,"00011 1     10001011    00101000","  e"
```

Appendix D

Hardware Test Results

Expected output vectors				CHIP 1 Actual output vectors				CHIP 2 Actual output vectors			
00000000	00000000	00000000	00000000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000000	00000000	10001100	10001011	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000000	00000010	00110010	00101100	00000000	00000000	00101000	01000000	00000000	00000000	00001000	00001000
00000000	00000100	11110000	11100011	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000000	00001000	11001000	10110000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000000	00001101	10111001	10010011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000000	00010011	11000011	10001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000000	00011010	11100110	10011011	00000001	00000001	00101000	01z11010	00000001	00000001	00001000	00001000
00000000	00100011	00100010	11000000	00000000	00000000	00001000	00011000	00000000	00000000	00001000	00001000
00000000	00101100	01110111	11111011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00000000	00110110	11100110	01001100	00000001	00000001	00101000	00000000	00000001	00000001	00101000	00101000
00000000	01000010	01101101	10110011	00000000	00000000	00001000	01010010	00000000	00000000	00101000	00101000
00000000	01001111	00001110	00110000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000000	01011100	11000111	11000011	00001011	00001011	00101000	00101000	00001011	00001011	00101000	00101000
00000000	01101011	10011010	01101100	00001011	00001011	00101000	01000010	00001011	00001011	00101000	00101000
00000000	01111011	10000110	00101011	00001011	00001011	00101000	00101000	00001011	00001011	00001000	00001000
00000000	10001100	10001011	00000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000000	10011110	10101000	11101011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000000	10110001	11011111	11101100	00001011	00001011	00101000	00101000	00001011	00001011	00101000	00101000
00000000	11000110	00110000	00000011	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
00000000	11011011	10011001	00110000	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
00000000	11110010	00011011	01110011	00000011	00000011	00101000	00101000	00000011	00000011	00101000	00101000
00000001	00001001	10110110	11001100	00000011	00000011	00101000	00101010	00000000	0000001z	00101000	00101000
00000001	00100010	01101011	00111011	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00000001	00111100	00111000	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000001	01010111	00011111	01011011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000001	01110011	00011111	00001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000001	10010000	00110111	11010011	00001010	00001010	00001000	00001000	00001011	00001010	00001000	00001000
00000001	10101110	01101001	10110000	00001011	00001011	00001000	01011010	00001011	00001011	00101000	00101000
00000001	11001101	10110100	10100011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
00000001	11101110	00011000	10101100	00001000	00001011	00101000	01100000	00001000	00001000	00001000	00001000
00000010	00001111	10010101	11001011	00000001	00000001	00101000	01101000	00000001	00000001	00101000	00101000
00000010	00110010	00101100	00000000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000010	01010101	11011011	01001011	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
00000010	01111010	10100011	10101100	00000011	00000011	00101000	00101000	00000011	00000011	00001000	00001000
00000010	10100000	10000101	00100011	00000011	00000000	00001000	01010000	00000011	00000011	00101000	00101000
00000010	11000111	01111111	10110000	00000010	00000010	00001000	01000010	00000010	00000010	00001000	00001000
00000010	11101111	10010011	01010011	00001001	00001001	00101000	00001010	00001001	00001001	00001000	00001000
00000011	00011000	11000000	00001100	00001010	00001010	00101000	00101000	00001001	00001010	00001000	00001000
00000011	01000011	00000101	11011011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
00000011	01101110	01100100	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000011	10011010	11011100	10111011	00001000	00001000	00001000	00001010	00001000	00001000	00101000	00101000
00000011	11001000	01101101	11001100	00001000	00001000	00101000	00101000	00001001	00001000	00101000	00101000
00000011	11110111	00010111	11110011	00001001	00001001	00001000	01001000	00001001	00001001	00101000	00101000
00000100	00100110	11011011	00110000	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
00000100	01010111	10110111	10000011	00001001	00001001	00101000	01000000	00001001	00001001	00101000	00101000
00000100	10001001	10101100	11101100	00000000	00000000	00101000	00101010	00000000	00000000	00101000	00101000
00000100	10111100	10111011	01101011	00000000	00000000	00101000	01000010	00000000	00000000	00001000	00001000
00000100	11110000	11100011	00000000	00000000	00000000	00001000	0z000000*	00000000	00000000	00101000	00101000
00000101	00100110	00100011	10101011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000

00100011	00100010	11000000	00000000	00000010	00000010	00001000	00001000	00000010	00000011	00001000	00001000
00100011	10101111	11010111	10001011	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
00100100	00111110	00001000	00101100	00000010	00000010	00101000	00101000	00000010	00000010	00001000	00001000
00100100	11001101	01010001	11100011	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
00100101	01011101	10110100	10110000	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
00100101	11101111	00110000	10010011	00001001	00001001	00101000	01010010	00001010	00001010	00001000	00001000
00100110	10000001	11000101	10001100	00001010	00001010	00101000	00101000	00001010	00001010	00001000	00001000
00100111	00010101	01110011	10011011	00001011	00001011	00101000	00101010	00001011	00001010	00001000	00001000
00100111	10101010	00111010	11000000	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
00101000	01000000	00011010	11111011	00001001	00001001	00001000	01000010	00001001	00001001	00101000	00101000
00101000	11010111	00010100	01001100	00000000	00000000	00101000	01110000	00000001	00000000	00101000	00101000
00101001	01101111	00100110	10110011	00000001	00000001	00001000	01010010	00000001	00000001	00101000	00101000
00101010	00001000	01010010	00110000	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
00101010	10100010	10010110	11000011	00000000	00000000	00101000	00101010	00000000	00000000	00101000	00101000
00101011	00111101	11110100	01101100	00000000	00000000	00101000	00111010	00000000	00000000	00101000	00101000
00101011	11011010	01101011	00101011	00101011	00101011	00101000	00101000	00101011	00101011	00001000	00001000
00101100	01110111	11111011	00000000	00101010	00101010	00001000	01001000	00101010	00101010	00101000	00101000
00101101	00010110	10100011	11101011	00101000	00101000	00001000	00001000	00101000	00101000	00001000	00001000
00101101	10110110	01100101	11101100	00100011	00100011	00101000	00101000	00100011	00100011	00101000	00101000
00101110	01010111	01000001	00000011	00100000	00100000	00001000	01001000	00100000	00100000	00001000	00001000
00101110	11111001	00110101	00110000	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
00101111	10011100	01000010	01110011	00100011	00100011	00101000	01010000	00100011	00100011	00101000	00101000
00110000	01000000	01101000	11001100	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
00110000	11100101	10101000	00111011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
00110001	10001100	00000000	11000000	00101010	00101010	00001000	00001000	00000010	00101010	00101000	00101000
00110010	00110011	01110010	01011011	00101011	00101011	00001000	00001000	00101011	00101011	00001000	00001000
00110010	11011011	11111101	00001100	00101001	00101001	00101000	00101000	00101010	00101001	00001000	00001000
00110011	10000101	10100000	11010011	00000011	00000011	00001000	01011000	00000011	00000011	00001000	00001000
00110100	00110000	01011101	10110000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
00110100	11011100	00110011	10100011	00001001	00001001	00101000	01010010	00001001	00001001	00001000	00001000
00110101	10001001	00100010	10101100	00100001	00100001	00101000	01011000	00100001	00100001	00001000	00001000
00110110	00110111	00010010	11001011	00101001	00101001	00101000	00101000	00000011	00101001	00101000	00101000
00110110	11100110	01001100	00000000	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
00110111	10010110	10000110	01001011	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
00111000	01000111	11011001	10101100	00001010	00001010	00101000	00100000	00001011	00001010	00001000	00001000
00111000	11111010	01000110	00100011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
00111001	10101101	11001011	10110000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00111010	01100010	01101010	01010011	00000001	00000001	00101000	00111010	00000001	00000001	00001000	00001000
00111011	00011000	00100010	00001100	00000000	00000000	00101000	00101010	00000000	00000000	00001000	00001000
00111011	11001110	11110010	11010111	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00111100	10000110	11011100	11000000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00111101	00111111	11011111	10111011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00111101	11111001	11111011	11001100	00000011	00000011	00101000	00101000	00001000	00000011	00101000	00101000
00111110	10110101	00110000	11110011	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00111111	01110001	01111111	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01000000	00101110	11100110	10000011	00000000	00000000	00101000	00101000	00001001	00000000	00101000	00101000
01000000	11101101	01100110	11101100	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
01000001	10101101	00000000	01101011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
01000010	01101101	10110011	00000000	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
01000011	00101111	01111110	10101011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01000011	11110010	01100011	01101100	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
01000100	10110110	01100001	01000011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01000101	01111011	01111000	00110000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
01000110	01000001	10101000	00110011	00000011	00000011	00101000	00001010	00000000	00000011	00101000	00101000
01000111	00001000	11110001	01001100	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
01000111	11010001	01010011	01111011	00001001	00001001	00101000	00101000	00001001	00001001	00101000	00101000
01001000	10011010	11001110	11000000	00000000	00000000	00001000	0z000000	00000000	00000000	00101000	00101000
01001001	01100101	01100011	00011011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
01001010	00110001	00010000	10001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
01001010	11111101	11010111	00010011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
01001011	11001011	10110110	10110000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
01001100	10011010	10101111	01100011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
01001101	01101010	11000001	00101100	00000010	00000010	00101000	00101000	00000010	00000010	00001000	00001000
01001110	00111011	11101100	00001011	00000010	00000010	00101000	0z100010	00000010	00000010	00101000	00101000
01001111	00001110	00110000	00000000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01001111	11100001	10001101	00001011	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
01010000	10110110	00000011	00101100	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
01010001	10001011	10010010	01100011	00000010	00000010	00001000	01012000	00000010	00000010	00101000	00101000
01010010	01100010	00111010	10110000	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
01010011	00111001	11111100	00010011	00001001	00001001	00101000	01002010	00001001	00001001	00001000	00001000
01010100	00010010	11010110	10001100	00001010	00001010	00101000	00101000	00001010	00001010	00001000	00001000
01010100	11101100	11001010	00011011	00000010	00000010	00101000	01011010	00000010	00000010	00001000	00001000
01010101	11000111	11010110	11000000	00000001	00000001	00001000	00001000	00000000	00000000	00001000	00001000
01010110	10100011	11111100	01111011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
01010111	10000001	00111011	01001100	00001011	00001011	00101000	01101010	00001011	00001011	00101000	00101000
01011000	01011111	10010011	00110011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
01011001	00111111	00000100	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01011010	00011111	10001110	01000011	00000010	00000001	00101000	00101000	00000001	00000001	00101000	00101000

01011011	00000001	00110001	01101100	00001000	00001000	00101000	00101000	00001000	00001000	00101000	00101000
01011011	11100011	11101101	10101011	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
01011100	11000111	11000011	00000000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
01011101	10101100	10110001	01101011	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
01011110	10010010	10111000	11101100	00001001	00001001	00101000	01010000	00001001	00001001	00101000	00101000
01011111	01111001	11011001	10000011	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
01100000	01100010	00010011	00110000	00001000	00001000	00001000	01001010	00001000	00001000	00101000	00101000
01100001	01001011	01100101	11110011	00001010	00001010	00101000	00101010	00001010	00001010	00101000	00101000
01100010	00110101	11010001	11001100	00001000	00001000	00101000	00101000	00001000	00001000	00101000	00101000
01100011	00100001	01010110	10111011	00001001	00001001	00101000	00101000	00001001	00001001	00101000	00101000
01100100	00001101	11110100	11000000	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
01100100	11111011	10101011	11011011	00001000	00001000	00001000	00001000	00001001	00001001	00001000	00001000
01100101	11101010	01111100	00001100	00001001	00001001	00101000	01000010	00001010	00001001	00001000	00001000
01100110	11011010	01100101	01010011	00001010	00001010	00001000	00001010	00001010	00001010	00001000	00001000
01100111	11001011	01100111	10110000	00001001	00001001	00001000	00001000	00001000	00001001	00101000	00101000
01101000	10111101	10000011	00100011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
01101001	10110000	10110111	10101100	00001001	00001001	00101000	01010010	00001001	00001001	00001000	00001000
01101010	10100101	00000101	01001011	00001001	00001001	00101000	00101000	00001001	00001001	00101000	00101000
01101011	10011010	01101100	00000000	00001010	00001010	00001000	00001000	00001010	00001010	00001000	00001000
01101100	10010000	11101011	11001011	00001010	00001010	00001000	00001000	00001001	00001010	00101000	00101000
01101101	10001000	10000100	10101100	00001010	00001010	00101000	00010010	00001010	00001010	00001000	00001000
01101110	10000001	00110110	10100011	00001011	00001010	00001000	00001000	00001010	00001011	00101000	00101000
01101111	01111011	00000001	10110000	00001011	00001011	00001000	01011000	00001011	00001011	00001000	00001000
01110000	01110101	11100101	11010011	00001000	00001000	00101000	00101010	00001000	00001000	00001000	00001000
01110001	01110001	11100011	00001100	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
01110010	01101110	11111001	01011011	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
01110011	01101101	00101000	11000000	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
01110100	01101100	01110001	00111011	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
01110101	01101100	11010010	11001100	00001010	00001010	00101000	00101000	00001010	00001010	00101000	00101000
01110110	01101110	01001101	01110011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
01110111	01110000	11100001	00110000	00001001	00001001	00001000	000010z0	00001001	00001001	00001000	00001000
01111000	01110100	10001110	00000011	00100000	00100000	00101000	00101000	00100000	00100000	00101000	00101000
01111001	01111001	01010011	11101100	00100001	00100001	00101000	001z0010	00100001	00100001	00101000	00101000
01111010	01111111	00110010	11101011	00101001	00101001	00101000	00101000	00101001	00101001	00001000	00001000
01111011	10000110	00101011	00000000	00001000	00001000	00001000	00001000	00001011	00001011	00101000	00101000
01111100	10001110	00111100	00101011	00100000	00100000	00001000	00001000	00100000	00100000	00001000	00001000
01111101	10010111	01100110	01101100	00101011	00101011	00101000	00101010	00101011	00101011	00101000	00101000
01111110	10100001	10101001	11000011	00100000	00100000	00001000	00001000	00100000	00100000	00001000	00001000
01111111	10101101	00000110	00110000	00100010	00100010	00001000	00001000	00100011	00100011	00101000	00101000
10000000	10111001	01111011	10110011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
10000001	11000111	00001010	01001100	00101000	00101000	00101000	00101000	00101000	00101000	00101000	00101000
10000010	11010101	10110001	11111011	00101000	00101000	00101000	00101000	00101000	00101000	00101000	00101000
10000011	11100101	01110010	11000000	00101001	00101001	00001000	00001000	00101001	00101001	00101000	00101000
10000100	11110110	01001100	10011011	00101000	00101000	00001000	00001000	00101000	00101000	00001000	00001000
10000110	00001000	00111111	10001100	00101000	00101000	00101000	00101000	00101000	00101000	00001000	00001000
10000111	00011011	01001011	10010011	00101000	00101000	00001000	00001000	00101000	00101000	00001000	00001000
10001000	00101111	01110000	10110000	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
10001001	01000100	10101110	11100011	00100001	00100001	00101000	00101000	00100001	00100001	00001000	00001000
10001010	01011011	00000110	00101100	00100010	00100010	00101000	01000000	00100010	00100010	00001000	00001000
10001011	01110010	01110110	10001011	00100000	00100000	00101000	00101000	00100000	00100000	00101000	00101000
10001100	10001011	00000000	00000000	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
10001101	10100100	10100010	10001011	00100010	00100010	00001000	00001000	00100010	00100010	00101000	00101000
10001110	10111111	01011110	00101100	00000011	00000011	00101000	00101000	00100010	00000011	00001000	00001000
10001111	11011011	00110010	11100011	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
10010000	11111000	00100000	10110000	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
10010010	00010110	00100111	10010011	00100001	00100001	00101000	01010010	00100001	00100001	00001000	00001000
10010011	00110101	01000111	10001100	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
10010100	01010101	10000000	10011011	00101010	00001011	00101000	01000010	00001011	00001011	00001000	00001000
10010101	01110110	11010010	11000000	00000011	00000011	00001000	00001000	00000011	00000011	00001000	00001000
10010110	10011001	00111101	11111011	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
10010111	10111100	11000010	01001100	00100000	00100000	00101000	00101000	00100000	00100000	00101000	00101000
10011000	11100001	01011111	10110011	00101001	00101001	00001000	01001010	00001010	00101001	00001000	00001000
10011010	00000111	00010110	00110000	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
10011011	00101101	11100101	11000011	00101011	00101011	00101000	00101010	00101011	00101011	00101000	00101000
10011100	01010101	11001110	01101100	00101000	00001001	00101000	01000010	00001001	00001001	00101000	00101000
10011101	01111110	11010000	00101011	00101001	00101001	00101000	00101000	00101001	00101001	00001000	00001000
10011110	10101000	11101011	00000000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
10011111	11010100	00011110	11101011	00000011	00000011	00001000	00000000	00100010	00000011	00001000	00001000
10100001	00000000	01101011	11101100	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
10100010	00101101	11010010	00000011	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
10100011	01011100	01010001	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
10100100	10001011	11101001	01110011	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
10100101	10111100	10011010	11001100	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
10100110	11101110	01100101	00111011	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
10101000	00100001	01001000	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
10101001	01010101	01000101	01011011	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
10101010	10001010	01011011	00001100	00001011	00001011	00101000	01011000	00001000	00001011	00001000	00001000

10101011	11000000	10001001	11010011	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
10101100	11110111	11010001	10110000	00000000	00000000	00001000	00001010	00000000	00000000	00101000	00101000
10101110	00110000	00110010	10100011	00001010	00001010	00101000	01001010	00001010	00001010	00001000	00001000
10101111	01101001	10101100	10101100	00101010	00101010	00101000	00101000	00101010	00101010	00001000	00001000
10110000	10100100	00111111	11001011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
10110001	11011111	11101100	00000000	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
10110011	00011100	10110001	01001011	00101001	00101001	00001000	00001000	00101010	00101001	00001000	00101000
10110100	01011010	10001111	10101100	00101011	00101011	00101000	01010000	00101010	00101010	00001000	00001000
10110101	10011001	10000111	00100011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
10110110	11011001	10010111	10110000	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
10111000	00011010	11000001	01010011	00100001	00100001	00101000	00101010	00100001	00100001	00001000	00001000
10111001	01011101	00000100	00001100	00100011	00100011	00101000	00101010	00100011	00100011	00001000	00001000
10111010	10100000	01011111	11011011	00101011	00101011	00101000	00101000	00101000	00101000	00001000	00001000
10111011	11100100	11010100	11000000	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
10111101	00101010	01100010	10111011	00100001	00100001	00001000	00001010	00100001	00100001	00101000	00101000
10111110	01110001	00001001	11001100	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
10111111	10111000	11001001	11110011	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
11000001	00000001	10100011	00110000	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
11000010	01001011	10010101	10000011	00101010	00101010	00101000	01000000	00101010	00101010	00101000	00101000
11000011	10010110	10100000	11101100	00000011	00000011	00101000	01101010	00000011	00000011	00101000	00101000
11000100	11100010	11000101	01101011	00000001	00000001	00101000	0zz00010	00100000	00000001	00001000	00001000
11000110	00110000	00000011	00000000	00101000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
11000111	01111110	01011001	10101011	00000000	00000000	00001000	01000000	00001010	00001010	00001000	00001000
11001000	11001101	11001001	01101100	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
11001010	00011110	01010010	01000011	00000011	00000011	00001000	00001000	00000011	00000011	00001000	00001000
11001011	01101111	11110100	00110000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
11001100	11000010	10101111	00110011	00001010	00001010	00101000	00101000	00001010	00001010	00101000	00101000
11001110	00010110	10000011	01001100	00001011	00001011	00101000	00101000	00001000	00001011	00101000	00101000
11001111	01101011	01110000	01111011	00001000	00001000	00101000	00101000	00001001	00001000	00101000	00101000
11010000	11000001	01110110	11000000	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
11010010	00011000	10010110	00011011	00100011	00100011	00001000	00001000	00000010	00100011	00001000	00001000
11010011	01110000	11001110	10001100	00001000	00001000	00101000	01010010	00100001	00001000	00001000	00001000
11010100	11001010	00100000	00010011	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
11010110	00100100	10001010	10110000	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
11010111	10000000	00001110	01100011	00101001	00101001	00101000	01000010	00101001	00101001	00001000	00001000
11011000	11011100	10101011	00101100	00000010	00000010	00101000	01001000	00100011	00000010	00001000	00001000
11011010	00111010	01100001	00001011	00100011	00100011	00101000	00101000	00100011	00100011	00101000	00101000
11011011	10011001	00110000	00000000	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
11011100	11111001	00011000	00001011	00101000	00101000	00001000	00001000	00101000	00101000	00101000	00101000
11011110	01011010	00011001	00101100	00101010	00101010	00101000	00101000	00101010	00101010	00001000	00001000
11011111	10111100	00110011	01100011	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
11100001	00011111	01100110	10110000	00101000	00101000	00001000	00001000	00101000	00101000	00001000	00001000
11100010	10000011	10110011	00010011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
11100011	11101001	00011000	10001100	00101011	00101011	00101000	00101000	0010z011	00000000	00001000	00001000
00000000	10101010	10010111	00011011	00000001	00000001	00101000	0z100010	00000001	00000001	00001000	00001000
00000001	01011111	00101110	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000010	00010100	11011111	01111011	00000000	00000000	00001000	00001010	00000000	00000000	00101000	00101000
00000010	11001011	10101001	01001100	00000000	00000000	00101000	00100010	00000000	00000000	00101000	00101000
00000011	10000011	10001100	00110011	00000001	00000001	00001000	01010000	00000001	00000001	00101000	00101000
00000100	00111100	10001000	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000100	11110110	10011101	01000011	00001001	00001001	00101000	00000000	00001001	00001001	00101000	00101000
00000101	10110001	11001011	01101100	00000000	00000000	00101000	01010000	00000000	00000000	00101000	00101000
00000110	01101110	00010010	10101011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000111	00101011	01110011	00000000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00000111	11101001	11101100	01101011	00000000	00000000	00001000	00001000	00000001	00000000	00001000	00001000
00001000	10101001	01111110	11101100	00000000	00000000	00101000	01010000	00000000	00000000	00101000	00101000
00001001	01101010	00101010	10000011	00000001	00000001	00001000	00001000	00001000	00000001	00001000	00001000
00001010	00101011	11101111	00110000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
00001010	11101110	11001100	11110011	00001010	00001010	00101000	00101000	00001010	00001010	00101000	00101000
00001011	10110010	11000011	11001100	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
00001100	01110111	11010011	10111011	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
00001101	00111101	11111100	11000000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00001110	00000101	00111110	11011011	00000011	00000011	00001000	00001000	00000011	00000011	00001000	00001000
00001110	11001101	10011010	00001100	00000001	00000001	00101000	00101010	00000001	00000001	00001000	00001000
00001111	10010111	00001110	01010011	00001001	00001001	00001000	01001010	00001001	00001001	00001000	00001000
00010000	01100001	10011011	10110000	00001001	00001000	00001000	00001000	00001000	00001001	00101000	00101000
00010001	00101101	01000010	00100011	00001001	00001001	00101000	00101000	00001001	00001001	00001000	00001000
00010001	11111010	00000001	10101100	00001010	00001010	00101000	00101010	00001010	00001010	00001000	00001000
00010010	11000111	11011010	01001011	00001010	00001010	00101000	01010000	00001010	00001010	00101000	00101000
00010011	10010110	11001100	00000000	00001010	00001010	00001000	00001000	00001010	00001010	00001000	00001000
00010100	01100110	11010110	11001011	00100011	00100011	00001000	00001000	00100010	00100011	00101000	00101000
00010101	00110111	11111010	10101100	00100011	00100011	00101000	00000010	00100011	00100011	00001000	00001000
00010110	00001010	00110111	10100011	00001011	00001011	00001000	00001000	00001011	00001011	00101000	00101000
00010110	11011101	10001101	10110000	00100000	00100000	00001000	00001000	00100000	00100000	00001000	00001000
00010111	10110001	11111100	11010011	00001011	00001011	00101000	01001010	00001011	00001011	00001000	00001000
00011000	10000111	10000101	00001100	00001010	00001010	00101000	01010000	00001010	00001010	00001000	00001000
00011001	01011110	00100110	01011011	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000

00011010	00110101	11100000	11000000	00101011	00101011	00001000	00001000	00101011	00101011	00001000	00001000
00011011	00001110	10110100	00111011	00100011	00100011	00001000	00001000	00100011	00100011	00101000	00101000
00011011	11101000	10100000	11001100	00001011	00001011	00101000	00101000	00001011	00001011	00101000	00101000
00011100	11000011	10100110	01110011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
00011101	10011111	11000101	00110000	00100011	00100011	00001000	00001000	00100010	00100011	00001000	00001000
00011110	01111100	11111101	00000011	00100000	00100000	00101000	00101010	00100000	00100000	00101000	00101000
00011111	01011011	01001101	11101100	00101010	00101010	00101000	00101000	00101010	00101010	00101000	00101000
00000000	00010011	10110111	11101011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000000	00101011	00111011	00000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000000	01000011	11010111	00101011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000000	01011101	10001100	01101100	00000001	00000001	00101000	00101010	00000001	00000001	00101000	00101000
00000000	01111000	01011010	11000011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000000	10010100	01000010	00110000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000000	10110001	01000010	10110011	00000000	00000000	00101000	01000000	00000000	00000000	00101000	00101000
00000000	11001111	01011100	01001100	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
00000000	11101110	10001110	11111011	00000001	00000001	00101000	01111000	00000001	00000001	00101000	00101000
00000001	00001110	11011010	11000000	00000010	00000010	00001000	01001000	00000010	00000010	00101000	00101000
00000001	00110000	00111111	10011011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00000001	01010010	10111101	10001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00000001	01110110	01010100	10010011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00000001	10011011	00000100	10110000	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
00000001	11000000	11001101	11100011	00001010	00001010	00101000	00100000	00001010	00001010	00001000	00001000
00000001	11100111	10110000	00101100	00001011	00001011	00101000	01111000	00001011	00001011	00001000	00001000
00000010	00001111	10101011	10001011	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
00000010	00111000	11000000	00000000	00001010	00001010	00001000	00001000	00001010	00001010	00001000	00001000
00000010	01100010	11101101	10001011	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
00000010	10001110	00110100	00101100	00001001	00001001	00101000	00101000	00001001	00001001	00001000	00001000
00000010	10111010	10010011	11100011	00000001	00000001	00001000	00001010	00000001	00000001	00101000	00101000
00000010	11101000	00001100	10110000	00001011	00001011	00001000	0z001000	00001011	00001011	00001000	00001000
00000011	00010110	10011110	10010011	00000000	00000000	00101000	00100010	00000000	00000000	00001000	00001000
00000011	01000110	01001001	10001100	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
00000011	01110111	00001101	10011011	00001001	00001001	00101000	00101000	00001000	00001000	00001000	00001000
00000011	10101000	11101010	11000000	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
00000011	11011011	11100000	11111011	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
00000100	00001111	11110000	01001100	00000000	00000000	00101000	01111000	00000001	00000000	00101000	00101000
00000100	01000101	00011000	10110011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
00000100	01111011	01011010	00110000	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
00000100	10110010	10110100	11000011	00000000	00000000	00101000	01011010	00000000	00000000	00101000	00101000
00000100	11101011	00101000	01101100	00001010	00001010	00101000"	00111010	00001010	00001010	00101000"	00101000
00000101	00100100	10110101	00101011	00001011	00001011	00101000	00101000	00001011	00001011	00001000	00001000
00000101	01011111	01011011	00000000	00001001	00000011	00001000	00001000	00000011	00000011	00101000	00101000
00000101	10011011	00011001	11101011	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
00000101	11010111	11110001	11101100	00001000	00001000	00101000	00101000	00100001	00001000	00101000	00101000
00000110	00010101	11100011	00000011	00000011	00000011	00001000	00001000	00000011	00000011	00001000	00001000
00000110	01010100	11101101	00110000	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
00000110	10010101	00010000	01110011	00000011	00000011	00101000	00101000	00000011	00000011	00101000	00101000
00000110	11010110	01001100	11001100	00001001	00001001	00101000	00101000	00001001	00001001	00101000	00101000
00000111	00011000	10100010	00111011	00101011	00101011	00101000	01000000	00101011	00101011	00101000	00101000
00000111	01011100	00010000	11000000	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
00000111	10100000	10011000	01011011	00001001	00001001	00001000	00001000	00001010	00001010	00001000	00001000
00000111	11100110	00111001	00001100	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
00001000	00101100	11110010	11010011	00001001	00001001	00001000	01011000	00001001	00001001	00001000	00001000
00001000	01110100	11000101	10110000	00100000	00100000	00001000	00001000	00100000	00100000	00101000	00101000
00001000	10111101	10110001	10100011	00100000	00100000	00101000	00101000	00100000	00100000	00001000	00001000
00001001	00000111	10110110	10101100	00101001	00101001	00101000	00101000	00101001	00101001	00001000	00001000
00001001	01010010	11010100	11001011	00101010	00101010	00101000	00101000	00101001	00101010	00101000	00101000
00001001	10011111	00001100	00000000	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
00001001	11101100	01011100	01001011	00101011	00101011	00001000	00001000	00101011	00101011	00101000	00101000
00001010	00111010	11000101	10101100	00100000	00100000	00101000	00101000	00100000	00100000	00001000	00001000
00001010	10001010	01001000	00100011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
00001010	11011010	11100011	10110000	00100010	00100010	00001000	00001010	00100010	00100010	00001000	00001000
00001011	00101100	10011000	01010011	00101010	00101010	00101000	00101010	00101010	00101010	00001000	00001000
00001011	01111111	01100110	00001100	00101011	00101011	00101000	01000010	00101011	00101011	00001000	00001000
00001011	11010011	01001100	11011011	00100001	00100001	00101000	00001000	00100001	00100001	00001000	00001000
00001100	00101000	01001100	11000000	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
00001100	01111110	01100101	10111011	00101001	00101001	00001000	01010010	00101001	00101001	00101000	00101000
00001100	11010101	10010111	11001100	00100000	00100000	00101000	00101000	00100000	00100000	00101000	00101000
00001101	00101101	11100010	11110011	00101011	00101011	00001000	000z1000	00000011	00101011	00101000	00101000
00001101	10000111	01000111	00110000	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
00001101	11100001	11000100	10000011	00101011	00101011	00101000	00101000	00101011	00101011	00101000	00101000
00001110	00111101	01011010	11101100	00100000	00100000	00101000	00101000	00100000	00100000	00101000	00101000
00001110	10011010	00001010	01101011	00101000	00101001	00101000	00101010	00101001	00101001	00001000	00001000
00001110	11110111	11010011	00000000	00101001	00101001	00001000	00001000	00101001	00101001	00101000	00101000
00001111	01010110	10110100	10101011	00101011	00101011	00001000	00001000	00101011	00101011	00001000	00001000
00001111	10110110	10101111	01101100	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00010000	00010111	11000011	01000011	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
00010000	01111001	11110000	00110000	00101001	00101001	00001000	00001000	00101001	00101001	00101000	00101000

00010000	11011101	00110110	00110011	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
00010001	01000001	10010101	01001100	00001000	00001000	00101000	00101000	00001000	00001000	00101000	00101000
00010001	10100111	00001101	01111011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
00010010	00001101	10011110	11000000	00101011	00101010	00001000	00001000	00101010	00101011	00101000	00101000
00010010	01110101	01001001	00011011	0000001z	00000011	00001000	00001000	00000011	00000011	00001000	00001000
00010010	11011110	00001100	10001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00010011	01000111	11101001	00010011	00000010	00000010	00001000	00001010	00000010	00000010	00001000	00001000
00010011	10110010	11011110	10110000	00100010	00100010	00001000	00001000	00100010	00100010	00101000	00101000
00010100	00011110	11101101	01100011	00001010	00001010	00101000	00101000	00001010	00001010	00001000	00001000
00010100	10001100	00010101	00101100	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
00010100	11111010	01010110	00001011	00001010	00000001	00101000	01011010	00000001	00001010	00101000	00101000
00010101	01101001	10110000	00000000	00000000	00000000	00001000	01001000	00000000	00000000	00001000	00001000
00010101	11011010	00100011	00001011	00000011	00000011	00001000	01010010	00000011	00000011	00101000	00101000
00010110	01001011	10101111	00101100	00000010	00000010	00101000	01001000	00000010	00000010	00001000	00001000
00010110	10111110	01010100	01100011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
00010111	00110010	00010010	10110000	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
00010111	10100110	11101010	00010011	00001001	00001001	00101000	00101000	00001001	00001001	00001000	00001000
00011000	00011100	11011010	10001100	00001010	00001010	00101000	00101000	00001010	00001010	00001000	00001000
00011000	10010011	11100100	00011011	00100011	00100011	00101000	00101000	00100011	00100011	00001000	00001000
00011001	00001100	00000110	11000000	00001010	00001010	00001000	01011000	00001010	00001010	00001000	00001000
00011001	10000101	01000010	01111011	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
00011001	11111111	10010111	01001100	00100001	00100001	00101000	00101000	00100010	00100001	00101000	00101000
00011010	01111011	00000101	00110011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
00011010	11110111	10001100	00110000	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
00011011	01110101	00101100	01000011	00100010	00100010	00101000	01001000	00100010	00100010	00101000	00101000
00011011	11110011	11100101	01101100	00101011	00101011	00101000	01011000	00101011	00101011	00101000	00101000
00011100	01110011	10110111	10101011	00101000	00101000	00101000	0z100000	00101000	00101000	00001000	00001000
00011100	11110100	10100011	00000000	00100000	00100000	00001000	00001000	00100000	00100000	00101000	00101000
00011101	01110110	10100111	01101011	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
00011101	11111001	11000100	11101100	00101000	00101000	00101000	01000000	00101000	00101000	00101000	00101000
00011110	01111101	11111011	10000011	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
00011111	00000011	01001011	00110000	00100001	00100001	00001000	01011010	00100001	00100001	00101000	00101000
00011111	10001001	10110011	11110011	00101011	00101011	00101000	00101000	00101011	00101011	00101000	00101000
00100000	00010001	00110101	11001100	00101011	00101011	00101000	0101z010	00101011	00101011	00101000	00101000
00100000	10011001	11010000	10111011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
00100001	00100011	10000100	11000000	00100010	00100010	00001000	00001000	00100010	00100010	00101000	00101000
00100001	10101110	01010001	11011011	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
00100010	00111010	00111000	00001100	00100011	00100011	00101000	00101000	00100011	00100011	00001000	00001000
00100010	*11000111*	*00110111*	*01010011*	*00101010*	*00101010*	*00001000*	*00001000*	00101010	00101010	00001000	00001000
00100011	01010101	01001111	10110000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00100011	11100100	10000001	00100011	00101000	00101000	00101000	00101000	00101000	00101000	00001000	00001000
00100100	01110100	11001011	10101100	00001011	00001011	00101000	01100010	00101010	00001011	00001000	00001000
00100101	00000110	00101111	01001011	00101010	00101010	00101000	00101000	00101010	00101010	00101000	00101000
00100101	10011000	10101100	00000000	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
00100110	00101100	01000001	11001011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
00100110	11000000	11110000	10101100	00001011	00001011	00101000	00101010	00001011	00001011	00001000	00001000
00100111	01010110	10111000	10100011	00001000	00001000	00001000	00001000	00001011	00001000	00101000	00101000
00100111	11101101	10011001	10110000	00001010	00001010	00001000	01010000	00001010	00001010	00001000	00001000
00101000	10000101	10010011	11010011	00000010	00000010	00101000	01011010	00000010	00000010	00001000	00001000
00101001	00011110	10100111	00001100	00001010	00001010	00101000	01001000	00000001	00001010	00001000	00001000
00101001	10111000	11010011	01011011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
00101010	01010100	00011000	11000000	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
00101010	11110000	01110111	00111011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00101011	10001101	11101110	11001100	00000000	00000000	00101000	00101000	00000001	00000000	00101000	00101000
00101100	00101100	01111111	01110011	00000000	00000000	00001000	01001010	00000000	00000000	00101000	00101000
00101100	11001100	00101001	00110000	00000000	00000000	00001000	01001010	00000000	00000000	00001000	00001000
00101101	01101100	11101100	00000011	00000000	00000000	00101000	01011010	00000000	00000000	00101000	00101000
00101110	00001110	11000111	11101100	00000001	00000001	00101000	01001010	00000001	00000001	00101000	00101000
00101110	10110001	10111100	11101011	00000001	00000001	00101000	00101000	00000001	00000001	00001000	00001000
00101111	01010101	11001011	00000000	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
00101111	11111010	11110010	00101011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00110000	10100001	00110010	01101100	00000011	00000011	00101000	01000010	00000011	00000011	00101000	00101000
00110001	01001000	10001011	11000011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00110001	11110000	11111110	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
00110010	10011010	10001001	10110011	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00110011	01000101	00101110	01001100	00001010	00001010	00101000	00101000	00001011	00001010	00101000	00101000
00110011	11110000	11101011	11111011	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00110100	10011101	11000010	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00110101	01001011	10110010	10011011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
00110101	11111010	10111011	10001100	00000011	00000011	00101000	00z00010	00000011	00000011	00001000	00001000
00110110	10101010	11011101	10010011	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
00110111	01011100	00011000	10110000	00001000	00001000	00001000	01000010	00001000	00001000	00101000	00101000
00111000	00001110	01101100	11100011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
00111000	11000001	11011010	00101100	00001010	00001010	00101000	01100000	00001001	00001010	00001000	00001000
00111001	01110110	01100000	10001011	00001001	00001001	00101000	01000010	00001001	00001001	00101000	00101000
00111010	00101100	00000000	00000000	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
00111010	11100010	10111000	10001011	0000101z	00001010	00001000	00001000	00001011	00001011	00101000	00101000

00111011	10011010	10001010	00101100	00001010	00001010	00101000	00101000	00001010	00001010	00001000	00001000
00111100	01010011	01110100	11100011	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
00111101	00001101	01111000	10110000	00001010	00001010	00001000	00001000	00001010	00001010	00001000	00001000
00111101	11001000	10010101	10010011	00001011	00001011	00101000	00101000	00100000	00001011	00001000	00001000
00111110	10000100	11001011	10001100	00001011	00001011	00101000	00101000	00001011	00001011	00001000	00001000
00111111	01000010	00011010	10011011	00100000	00100000	00101000	00101010	00100000	00100000	00001000	00001000
01000000	00000000	10000010	11000000	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
01000000	11000000	00000011	11111011	00001011	00001011	00001000	00001000	00001011	00001011	00101000	00101000
01000001	10000000	10011110	01001100	00001000	00001000	00101000	00100000	00001000	00001000	00101000	00101000
01000010	01000010	01010001	10110011	00101001	00101001	00001000	01011010	00101001	00101001	00101000	00101000
01000011	00000101	00011110	00110000	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
01000011	11001001	00000011	11000011	00001011	00001011	00101000	01111010	00001000	00001011	00101000	00101000
01000100	10001110	00000010	01101100	00001001	00001001	00101000	00101000	00001001	00001001	00101000	00101000
01000101	01010100	00011010	00101011	00100000	00100000	00101000	00101000	00100000	00100000	00001000	00001000
01000110	00011011	01001011	00000000	00101011	00101011	00001000	00001000	00101011	00101011	00101000	00101000
01000110	11100011	10010100	11101011	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
01000111	10101100	11110111	11101100	00101001	00101001	00101000	0z101000	00101001	00101001	00101000	00101000
01001000	01110111	01110100	00000011	00101011	00101010	00001000	00001000	00101010	00101010	00001000	00001000
01001001	01000011	00001001	00110000	00101010	00101010	00001000	00001000	00101011	00101011	00101000	00101000
01001010	00001111	10110111	01110011	00101001	00101001	00101000	00101000	00101001	00101001	00101000	00101000
01001010	11011101	01111110	11001100	00101000	00101000	00101000	01011010	00101001	00101000	00101000	00101000
01001011	10101100	01011111	00111011	00101001	00101001	00101000	01010000	00101001	00101001	00101000	00101000
01001100	01111100	01011000	11000000	00101010	00101011	00001000	00001000	00101011	00101011	00101000	00101000
01001101	01001101	01101011	01011011	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
01001110	00011111	10010111	00001100	00101000	00101000	00101000	00101000	00101000	00101000	00001000	00001000
01001110	11110010	11011011	11010011	00100001	00100001	00001000	00001000	00000010	00000010	00001000	00001000
01001111	11000111	00111001	10110000	00000010	00000010	00001000	00001010	00100001	00000010	00101000	00101000
01010000	10011100	10110000	10100011	00101010	00101010	00101000	00101000	00101010	00101010	00001000	00001000
01010001	01110011	01000000	10101100	00101010	00101010	00101000	00011000	00101010	00101010	00001000	00001000
01010010	01001010	11101001	11001011	00001011	00001011	00101000	00101000	00001011	0000101L	00101000	00101000
01010011	00100011	10101100	00000000	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
01010011	11111101	10000111	01001011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
01010100	11011000	01111011	10101100	00101010	00101010	00101000	00100000	00101010	00101010	00001000	00001000
01010101	10110100	10001001	00100011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
01010110	10010001	10101111	10110000	00000011	00000011	00001000	00001010	00000011	00000011	00001000	00001000
01010111	01101111	11101111	01010011	00000011	00000011	00101000	01001010	00000011	00000011	00001000	00001000
01011000	01001111	01001000	00001100	00001011	00001011	00101000	00101000	00001011	00001011	00001000	00001000
01011001	00101111	10111001	11011011	00000000	00000000	00101000	01000000	00000000	00000000	00001000	00001000
01011010	00010001	01000100	11000000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01011010	11110011	11101000	10111011	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
01011011	11010111	10100101	11001100	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
01011100	10111100	01111011	11110011	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
01011101	10100010	01101011	00110000	00000001	00000001	00001000	00001000	00000001	00000001	00001000	00001000
01011110	10001001	01110011	10000011	00000011	00000011	00101000	00101000	00000011	00000011	00101000	00101000
01011111	01110001	10010100	11101100	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
01100000	01011010	11001111	01101011	00000011	00000011	00101000	01010010	00000011	00000011	00001000	00001000
01100001	01000101	00100011	00000000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
01100010	00110000	10001111	10101011	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
01100011	00011101	00010101	01101100	00000010	00000010	00101000	00101000	00000010	00000010	00101000	00101000
01100100	00001010	10110100	01000011	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
01100100	*11111001*	*01101100*	*00110000*	*00000001*	*00000001*	*00001000*	*01001010*	*00000001*	*00000001*	*00101000*	*00101000*
01100101	11101001	00111101	00110011	00000001	00000001	00101000	00101010	00000010	00000001	00101000	00101000
01100110	11011010	00100111	01001100	00000000	00000000	00101000	011z1010	00000000	00000000	00101000	00101000
01100111	11001100	00101010	01111011	00000011	00000010	00101000	00101000	00000011	00000011	00101000	00101000
01101000	10111111	01000110	11000000	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
01101001	10110011	01111100	00011011	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
01101010	10101000	11001010	10001100	00001000	00001000	00101000	00101000	00001000	00001000	00001000	00001000
01101011	10011111	00110010	00010011	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000
01101100	10010110	10110010	10110000	00001010	00001001	00001000	00000000	00001001	00001001	00101000	00101000
01101101	10001111	01001100	01100011	00000000	00000000	00101000	00101000	00000001	00000001	00001000	00001000
01101110	10001000	11111111	00101100	00000001	00000001	00101000	01010000	00000001	00000001	00001000	00001000
01101111	10000011	11001011	00001011	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
01110000	01111111	10110000	00000000	00000000	00000000	00001000	00001000	00000000	00000000	00001000	00001000
01110001	01111100	10101110	00001011	00000011	00000011	00001000	00001000	00000011	00000011	00101000	00101000
01110010	01111010	11000101	00101100	00001010	00001010	00101000	01010000	00001010	00001010	00001000	00001000
01110011	01111001	11110101	01100011	00000001	00000000	00001000	00000000	00000001	00000001	00101000	00101000
01110100	01111010	00111110	10110000	00001000	00001000	00001000	00011010	00001000	00001000	00001000	00001000
01110101	01111011	10100001	00010011	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
01110110	01111110	00011100	10001100	00000000	00000000	00101000	00101000	00000000	00000000	00001000	00001000
01110111	10000001	10110001	00011011	00000001	00000001	00101000	01000010	00000001	00000001	00001000	00001000
01111000	10000110	01011110	11000000	00000010	00000010	00001000	00001000	00000010	00000010	00001000	00001000
01111001	10001100	00100101	01111011	00100001	00100001	00001000	00001010	00100001	00100001	00101000	00101000
01111010	10010011	00000101	01001100	00001001	00001001	00101000	00101000	00001010	00001001	00101000	00101000
01111011	10011010	11111110	00110011	00001000	00001000	00001000	00001000	00001000	00001000	00101000	00101000
01111100	10100100	00010000	00110000	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
01111101	10101110	00111011	01000011	00000010	00001010	00101000	00101000	00001010	00001010	00101000	00101000
01111110	10111001	01111111	01101100	00101010	00101010	00101000	00101000	00101010	00101010	00101000	00101000

01111111	11000101	11011100	10101011	00000000	00000000	00101000	01z01000	00000000	00000000	00001000	00001000
10000000	11010011	01010011	00000000	00000001	00000001	00001000	00001000	00000001	00000001	00101000	00101000
10000001	11100001	11100010	01101011	00001010	00001010	00001000	00001000	00001010	00001010	00001000	00001000
10000010	11110001	10001010	11101100	00001011	00001011	00101000	01010000	00001011	00001011	00101000	00101000
10000100	00000010	01001100	10000011	00101000	00101000	00001000	01011010	00101000	00101000	00001000	00001000
10000101	00010100	00100111	00110000	00101001	00101001	00001000	00001000	00101001	00101001	00101000	00101000
10000110	00100111	00011010	11110011	00101001	00101001	00101000	00101000	00101010	00101001	00101000	00101000
10000111	00111011	00100111	11001100	00100010	00100010	00101000	00101000	00100010	00100010	00101000	00101000
10001000	01010000	01001101	10111011	00101001	00101001	00101000	00100000	00101001	00101001	00101000	00101000
10001001	01100110	10001100	11000000	00101000	00101000	00001000	00001000	00101000	00101000	00101000	00101000
10001010	01111101	11100100	11011011	00100001	00100001	00001000	00001000	00100001	00100001	00001000	00001000
10001011	10010110	01010110	00001100	00101010	00101010	00101000	00101000	00101010	00101010	00001000	00001000
10001100	10101111	11100000	01010011	00101011	00101011	00001000	00001010	00101011	00101011	00001000	00001000
10001101	11001010	10000011	10110000	00000000	00100011	00001000	00001000	00100011	00000000	00101000	00101000
10001110	11100110	01000000	00100011	00100000	00100000	00101000	00101000	00100000	00100000	00001000	00001000
10010000	00000011	00010101	10101100	00000010	00000010	00101000	00101000	00100001	00000010	00001000	00001000
10010001	00100001	00000100	01001011	00100010	00100010	00101000	00101000	00100011	00100010	00101000	00101000
10010010	01000000	00001100	00000000	00100011	00100011	00001000	00001000	00100011	00100011	00001000	00001000
10010011	01100000	00101100	11001011	00000010	00000010	00001000	00001000	00000010	00000010	00101000	00101000
10010100	10000001	01100110	10101100	00000000	00000000	00101000	00100010	00000000	00000000	00001000	00001000
10010101	10100011	10111001	10100011	00001011	00001011	00001000	00001000	00001011	00001011	00101000	00101000
10010110	11000111	00100101	10110000	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
10010111	11101011	10101010	11010011	00000001	00000001	00101000	01011010	00000001	00000001	00001000	00001000
10011001	00010001	01001001	00001100	00100010	00100010	00101000	00101000	00100010	00100010	00001000	00001000
10011010	00111000	00000000	01011011	00100001	00100001	00101000	00101000	00100001	00100001	00001000	00001000
10011011	01011111	11010000	11000000	00100010	00100010	00001000	0101z000	00100010	00100010	00001000	00001000
10011100	10001000	10111010	00111011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
10011101	10110010	10111100	11001100	00001000	00001000	00101000	00101000	00001000	00001000	00101000	00101000
10011110	11011101	11011000	01110011	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
10100000	00001010	00001101	00110000	00001011	00001011	00001000	00001000	00001011	00001011	00001000	00001000
10100001	00110111	01011011	00000011	00000000	00000000	00101000	01000010	00000000	00000000	00101000	00101000
10100010	01100101	11000001	11101100	00001010	00001010	00101000	01000010	00001010	00001010	00101000	00101000
10100011	10010101	01000001	11101011	00001001	00001001	00101000	00001010	00001001	00001001	00001000	00001000
10100100	11000101	11011011	00000000	00001001	00001001	00001000	00001000	00001001	00001001	00101000	00101000
10100101	11110111	10001101	00101011	00100000	00100000	00001000	00001000	00100010	00100010	00001000	00001000
10100111	00101010	01011000	01101100	00001001	00001001	00101000	00100010	00001001	00000011	00101000	00101000
10101000	01011110	00111100	11000011	00000010	00000010	00001000	00001000	00000001	00000001	00001000	00001000
10101001	10010011	00111010	00110000	00001010	00001010	00001000	00001000	00001010	00001010	00101000	00101000
10101010	11001001	01010000	10110011	00000011	00000011	00101000	00101000	00000011	00000011	00101000	00101000
10101100	00000000	10000000	01001100	00101000	00101000	00101000	00101000	00101000	00101000	00101000	00101000
10101101	00111000	11001000	11111011	00101010	00101010	00101000	00101000	00101010	00101010	00101000	00101000
10101110	01110010	00101010	11000000	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
10101111	10101100	10100101	10011011	00101010	00101010	00001000	00001000	00101010	00101010	00001000	00001000
10110000	11101000	00111001	10001100	00100000	00100000	00101000	00101000	00100000	00100000	00001000	00001000
10110010	00100100	11100110	10010011	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
10110011	01100010	10101100	10110000	00101010	00101010	00001000	01010010	00101010	00101010	00101000	00101000
10110100	10100001	10001011	11100011	00101011	00101011	00101000	01100000	00101011	00101011	00001000	00001000
10110101	11100001	10000100	00101100	00101001	00101001	00101000	01100000	00101001	00101001	00001000	00001000
10110111	00100010	10010101	10001011	00101000	00101000	00101000	00101000	00000000	00000000	00101000	00101000
10111000	01100100	11000000	00000000	00101001	00101001	00001000	00001000	00101001	00101001	00001000	00001000
10111001	10101000	00000011	10001011	00100001	00100001	00001000	00001000	00100001	00100001	00101000	00101000
10111010	11101100	01100000	00101100	00100001	00100001	00101000	01010000	00100001	00100001	00001000	00001000
10111100	00110001	11010101	11100011	00100011	00100011	00001000	00001010	00100011	00100011	00101000	00101000
10111101	01111000	01100100	10110000	00100010	00100010	00001000	00001000	00100010	00100010	00001000	00001000
10111110	11000000	00001100	10010011	00100011	00100011	00101000	00101010	00100011	00100011	00001000	00001000
11000000	00001000	11001101	10001100	00101000	00101000	00101000	01z00000	00101000	00101000	00001000	00001000
11000001	01010010	10100111	10011011	00000011	00100010	00101000	01011010	00100010	00100010	00001000	00001000
11000010	10011101	10011010	11000000	00100011	00100011	00001000	00001000	00100011	00100011	00001000	00001000
11000011	11101001	10100110	11111011	00101001	00101001	00001000	00001000	00101001	00101001	00101000	00101000
11000101	00110110	11001100	01001100	00100010	00100010	00101000	00101000	00100010	00100010	00101000	00101000
11000110	10000101	00001010	10110011	00001011	00001011	00001000	00001000	00001011	00001011	00101000	00101000
11000111	11010100	01100010	00110000	00001001	00001001	00001000	01010010	00001001	00001001	00001000	00001000
00000000	10001101	11010010	11000011	00000000	00000000	00101000	00101010	00000000	00000000	00101000	00101000
00000001	00101100	01011100	01101100	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00000001	11001011	11111111	00101011	00000000	00000000	00101000	01001000	00000000	00000000	00001000	00001000
00000010	01101100	10111011	00000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000011	00001110	10001111	11101011	00000001	00000001	00001000	00001000	00000000	00000001	00001000	00001000
00000011	10110001	01111101	11101100	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00000100	01010101	10000101	00000011	00000010	00000010	00001000	01011000	00000010	00000010	00001000	00001000
00000100	11111010	10100101	00110000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00000101	10100000	11011110	01110011	00000001	00000001	00101000	00101000	00000001	00000001	00101000	00101000
00000110	01001000	00110000	11001100	00000011	00000011	00101000	00101000	00000011	00000011	00101000	00101000
00000110	11110000	10011100	00111011	00000000	00000000	00101000	00101000	00000000	00000000	00101000	00101000
00000111	10011010	00100000	11000000	00000000	00000000	00001000	00001000	00000000	00000000	00101000	00101000
00001000	01000100	10111110	01011011	00001000	00001000	00001000	00001000	00001000	00001000	00001000	00001000
00001000	11110000	01110101	00001100	00001001	00001001	00101000	00101000	00001001	00001001	00001000	00001000
00001001	10011101	01000100	11010011	00001001	00001001	00001000	00001000	00001001	00001001	00001000	00001000

Vita Auctoris

Mike Howard, born 15 August, 1977 in Windsor Ontario, Canada. He received a B.A.Sc. in Electrical Engineering from the University of Windsor in 2001. Mike has gained invaluable professional and life experience through his enrolment in the co-op program. His professional employment includes positions at Ford Motor Company, Daimler Chrysler Corporation, JDS Uniphase and Nortel Networks.

In Fall 2002, Mike began the pursuit of a M.A.Sc. degree in Electrical Engineering under the supervision of Dr. Majid Ahmadi at the University of Windsor. During this time he published three papers at IEEE endorsed conference proceedings, and was an active member of the Research Center for Integrated Microsystems.

Currently Mike is pursuing a career in industry.

Vita Auctoris

Mike Howard, born 15 August, 1977 in Windsor Ontario, Canada. He received a B.A.Sc. in Electrical Engineering from the University of Windsor in 2001. Mike has gained invaluable professional and life experience through his enrolment in the co-op program. His professional employment includes positions at Ford Motor Company, Daimler Chrysler Corporation, JDS Uniphase and Nortel Networks.

In Fall 2002, Mike began the pursuit of a M.A.Sc. degree in Electrical Engineering under the supervision of Dr. Majid Ahmadi at the University of Windsor. During this time he published three papers at IEEE endorsed conference proceedings, and was an active member of the Research Center for Integrated Microsystems.

Currently Mike is pursuing a career in industry.