Design of An IEEE Double Precision Floating-Point Adder/Subtractor in GaAs Technology

Qiong Liu

A Thesis Submitted for The Degree of Master of

Engineering Science

In

the Department of Electrical and Electronic Engineering

The University of Adelaide

Adelaide, South Australia.

November, 1995

Table of Contents

Abstract

This project aims to produce a 64-bit floating-point double precision adder/subtractor of a Solid Modelling accelerator in Gallium Arsenide technology, which is used to reduce computational time and to increase accuracy of algorithms.

Addition is the most fundamental and the simplest operation in any computer arithmetic operation. According to the IEEE 754 Standard Format, the resulting architecture based on the addition/subtraction algorithm is mainly divided into two portions - the exponent and the mantissa.

In logic design, there are three major difficult circuits including a mantissa shifter, a mantissa adder, and a normalizer, all of which affect the speed of the addition process. Design of a high speed operation not only relies on speed property of a gate, an efficient way to accelerate the process but also is on selection and design of the fastest feasible circuits, and an optimal placement to reduce interconnections. A rapid barrel shifter has been used as an alignment shifter and a normalization shifter, and an adder combined a carry-select adder and a binary carry-look ahead adder has been developed for adding the two mantissas. In normalization, a novel approach has been adopted while designing the encoder, which omits the 6-bit incrementer normally required in this process.

As a result of the idiosyncrasies of GaAs technology, the design is much more difficult than CMOS. For simplicity of layout, a multi-bit-input circuit has been broken into several segments, then connected together to achieve the desired function. Furthermore, some examples of PLA implementation are given in the priority detector and the encoder.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and to the best of my knowledge and belief, contains no material previously published or written by any other person, except where due reference in made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

SIGNED: DATE: 11th, April 1996

Acknowledgement

I would like to express my gratitude to my supervisor Mr. Michael Liebelt for his guidance and valuable help throughout the course of this project.

In particular, I am grateful to Mrs. Pat Brooks, my beloved "Fang Dong", for her strong support, good advice and constant encouragement through difficult times in many ways. I also wish to express my deepest gratitude to my family continuous support and concern.

I would also like to express my gratitude to dear Paul Rohan and Charlie for reading a rough draft and correcting my English with patience. I would like to give thanks to my good friend Xiaoming and my colleagues Michael (MeGeeve), Andrews, Nozar, Kim, Eric, Thong, Song and all of who have given me any help. Special thanks go to my good friend Ali (Moini) due to giving me confidence and encouragement.

Once again, thanks to all who made this thesis possible.

Abbreviation

AC:	*alternating current*
ADD:	*adder*
A_E:	*the exponent of data A*
A_M:	*the mantissa of data A*
BCLA:	*binary carry look-ahead*
BS:	*barrel shifter*
CAD:	*computer aide design*
CAM:	*computer aide manufacturing*
CLA:	*carry look-ahead*
CMOS:	*complementary metal-oxide-semiconductor*
D-mode:	*deplete-type field effect mode*
D:	*dimension*
DC:	*direct current*
DCFE-mode:	*enhancement-type field effect mode*
XOR:	*exclusive or*
Exp:	*exponent*
GND:	*ground*
GaAs:	*gallium arsenide*
IC:	*integrated circuit*
L:	*length of transistor*
LSB:	*the least significant bit*
LSM:	*the least significant bit of mantissa*
MESFET:	*metal semiconductor field effect transistor*
mm^2:	*square micrometers*
MOSFET:	*metal oxide semiconductor field effect transistor*
MSB:	*the most significant bit*
MSM:	*the most significant bit of mantissa*
MUX:	*multiplex*
mV:	*millivoltages*
mW:	*miliwatt*

ns: *nanosecond*

PLA: *programmable logic array*

ps: *picosecond*

pu: *pull up*

RC: *ripple carry*

R_E: *the result of exponent difference*

R_M: *the result of mantissa*

SBFL: *super buffer field effect transistor logic*

SDCFL: *source follower direct couple field effect transistor logic*

SUB: *subtractor*

VLSI: *very large-scale integration*

W: *width of transistor*

List of Figures

List of Tables

Chapter 1 Introduction

Abstract: *This thesis describes the design of a high-speed* ***double-precision floating-point adder/subtractor,*** *as an arithmetic processing unit of a hardware accelerator for* ***Solid Modelling*** *algorithms. In this* ***VLSI*** *digital design, we investigate high speed digital* ***GaAs*** *technology to achieve the required fast computer arithmetic. In order to achieve high precision arithmetic operations in the solid modelling algorithms, a 64-bit double precision floating-point standard format, consisting of two major parts: the exponent and the mantissa, is used. To satisfy the design requirements of high-speed, high-density, low power and high reliability, we need to select and create optimal and feasible circuits for each subsection of the whole processor. This chapter surveys the scope of the thesis.*

1.1 The Aim of This Project

CAD/CAM technology, supporting many computerized activities in design and manufacturing, is now playing an increasingly important role in production industries. When using CAD for design, it is frequently necessary to both represent and manipulate a three-dimensional (3-D) shape in a computer. This process is called "Solid Modelling" which has application in many fields, such as computer graphics and visualization, architecture and construction, computer vision, integrated circuits, electronic packaging, and CAD/CAM computer systems.

Solid Modelling, which is used to represent three-dimensional shapes [Chi88] in hardware and/or software, comprises the theories, techniques, and systems representations of solids. A solid model has a clear and accurate computer representation of a physical solid object. The primary representation schemes for solid modelling can be divided into two broad categories: *contractive solid geometry* (CSG) and *boundary representation* (B-Rep). Solid objects in the CSG system are represented as a combination of some primitive objects i.e., cubes, spheres, cores and so on, which are then combined using regularized Boolean set operators. Its algorithms are simple and fast. The B-Rep system uses a graph data structure to describe an object in terms of its surface boundary elements such as vertices, edges, faces etc. Such a representation has high accuracy and can express a very wide class of objects with great flexibility, however, the relevant algorithms are complex and slow.

The project focuses on B-Rep algorithms using merged GaAs/CMOS/BiCMOS technologies to implement a Solid Modelling accelerator [Moo93]. This is a part of a project to produce a Solid Modelling accelerator. Solid Modelling systems demand reducing computational time (high speed) and to increase accuracy of algorithms (high precision) requiring the use of high speed double precision 64-bit floating-point calculations. In order to increase the throughout of a processor usually two approaches are used: parallelism or technological improvement like GaAs. In any case, optimization of logic circuits is required.

The specific objective of the project aims to design an efficient and possible rapid double precision floating-point adder/subtractor for a vector processor in the Solid Modelling accelerator.

1.1 The Aim of This Project

CAD/CAM technology, supporting many computerized activities in design and manufacturing, is now playing an increasingly important role in production industries. When using CAD for design, it is frequently necessary to both represent and manipulate a three-dimensional (3-D) shape in a computer. This process is called "Solid Modelling" which has application in many fields, such as computer graphics and visualization, architecture and construction, computer vision, integrated circuits, electronic packaging, and CAD/CAM computer systems.

Solid Modelling, which is used to represent three-dimensional shapes [Chi88] in hardware and/or software, comprises the theories, techniques, and systems representations of solids. A solid model has a clear and accurate computer representation of a physical solid object. The primary representation schemes for solid modelling can be divided into two broad categories: *contractive solid geometry* (CSG) and *boundary representation* (B-Rep). Solid objects in the CSG system are represented as a combination of some primitive objects i.e., cubes, spheres, cores and so on, which are then combined using regularized Boolean set operators. Its algorithms are simple and fast. The B-Rep system uses a graph data structure to describe an object in terms of its surface boundary elements such as vertices, edges, faces etc. Such a representation has high accuracy and can express a very wide class of objects with great flexibility, however, the relevant algorithms are complex and slow.

The project focuses on B-Rep algorithms using merged GaAs/CMOS/BiCMOS technologies to implement a Solid Modelling accelerator [Moo93]. This is a part of a project to produce a Solid Modelling accelerator. Solid Modelling systems demand reducing computational time (high speed) and to increase accuracy of algorithms (high precision) requiring the use of high speed double precision 64-bit floating-point calculations. In order to increase the throughout of a processor usually two approaches are used: parallelism or technological improvement like GaAs. In any case, optimization of logic circuits is required.

The specific objective of the project aims to design an efficient and possible rapid double precision floating-point adder/subtractor for a vector processor in the Solid Modelling accelerator.

1.2 Choice of Technology

Realization of the floating-point adder/subtractor has to be based on an appropriate technology. At present, there are two common technologies for custom digital circuit designs: gallium arsenide (GaAs) and silicon. Although silicon is a very mature technology and has extensively been used in computer arithmetic design and many applications, GaAs is also beginning to achieve maturity, and its application is spreading out in many fields. Results reported so far indicate that acceptable reliability levels in GaAs can be reached, and are comparable with those of silicon ICs [Mil86].

The growing popularity of GaAs is due to its capability of improving speed significantly. GaAs succeeds not only as a result of high speed, but also low power at high frequency.

Among the many types of GaAs devices, digital VLSI design based on MESFET is now a mature and commercially available technology, and also increases allowing high density circuit implementation [Roc90] [Tie94]. To avoid V_{SS} (negative voltage supply) body effect and an extra voltage supply influences, only E/D-MESFET GaAs, especially three static logic families (DCFL, SDCFL and SBFL), are treated in the project. The DCFL logic family plays a dominant role among these three families, as it is used for realizing the logic functions. SDCFL and SBFL are used as buffers for driving large loads.

Performance merits, such as speed, area, noise immunity, and power consumption, rely primarily on gate properties which are in turn determined by the transistor dimensions used in the gate. Of course, they are also dependent upon the logic circuit configuration, interconnections, design styles and so forth. In the following chapter, we will clarify these parameters and how they affect performance at the gate logic level.

A 0.8 micron E/D-MESFET process with three layers of metal has been used in the VLSI implementation of the floating-point adder/subtractor. A 2 volts supply voltage has been used.

1.3 Floating-Point Addition/Subtraction

High speed and high precision computations are required not only for the Solid Modelling accelerator but also for many other applications like signal processing, image processing, computer graphics, model simulations, and CAD/CAM. During the last two decades many floating-point processors have been reported in detail. A 64-bit double precision floating-point adder/subtractor in GaAs technology, however, has received little investigation.

Our design adheres to the ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic [IEEE754]. By definition, a binary floating-point number is represented by a bit string characterized by three components: a sign of the mantissa, a signed exponent, and a mantissa. In contrast to fixed-point arithmetic, the floating-point arithmetic increases either the range, or the precision of the number's representation for the same number of bits.

Floating-point addition and subtraction are more complex than floating-point multiplication and division. A floating-point adder/subtractor needs several steps to complete: compare the two exponents, shift the smaller mantissa to right with regard to the difference between the two exponents, add/subtract the mantissas, normalize the mantissa, and finally adjust the exponent. However, a floating-point multiplier/divisor can operate exponents and mantissas respectively.

In developing a VLSI architecture for the floating-point adder/subtractor the designer is faced with a number of important choices. These include: a) combinational vs. sequential processing; and b) what kind of shifter, adder and normalization circuits to select in practical circuit designs. The correct architecture will depend heavily on the application and its constraints on speed performance, power, reliability, and of course cost. The primary criteria considered in the design of this adder/subtractor application are the speed and feasibility of its implementation.

Implementation of operations within the adder/subtractor (such as normalization) using combinational logic circuitry, in general, provides minimum execution delay of each operation, and is also more straightforward to design. Thus, combinational logic is used where practical. Also all internal registers are edge-triggered with

a single clock.

Pipelines in sequential logic are generally used to improve performance in a processor. In [Mil86$_1$] the pipeline studies, however, refer to microprocessor design, where data and instructions are stored in external memories. In this case the pipeline equilibration is difficult because on chip-off chip communications delay is high compared to internal delays. Of course, this is not directly applicable to arithmetic coprocessors which are not microprogrammed. Especially in a GaAs environment, pipeline implementation is much more difficult than in silicon [Mil86$_1$]. Although the combinational logic design may require much more space, it removes the need of complicated control parts and avoids clock distribution problems. In addition, we can use only the static GaAs logic families to build the adder and the subtractor.

Three of the most critically important circuits in the adder/subtractor are the 56-bit shifter, the 56-bit mantissa adder and the 56-bit priority detector. Their speed largely determines the speed of the whole processor. We use a rapid barrel shifter to accomplish the alignment shifter and the normalization shifter functions. A novel combinational adder consisting of a binary look-ahead and a carry select adder is demonstrated and shown to yield good simulation results. In the normalization part, a novel approach of omitting a 6-bit incrementer is introduced.

Due to the characteristics of GaAs technology, such as the lack of complementary p-FET devices, Schottky Diode OR gates, pass transistors, NAND gate, and the limitation of fan-in and fan-out's, the resulting design tends to be more complex. As a result, we should take into account the feasibility of optimal logic circuits in GaAs layout design.

1.4 Overview of The Thesis

Figure 1-1 shows the structure of this thesis. We begin in Chapter 1 with an introduction to the thesis. This chapter describes briefly the purpose of the project, the technology choice and the floating-point addition/subtraction methodologies.

Chapter 2 reviews high-speed ratioed GaAs technology principles and its advantages and disadvantages compared with the CMOS technology. We emphasize the

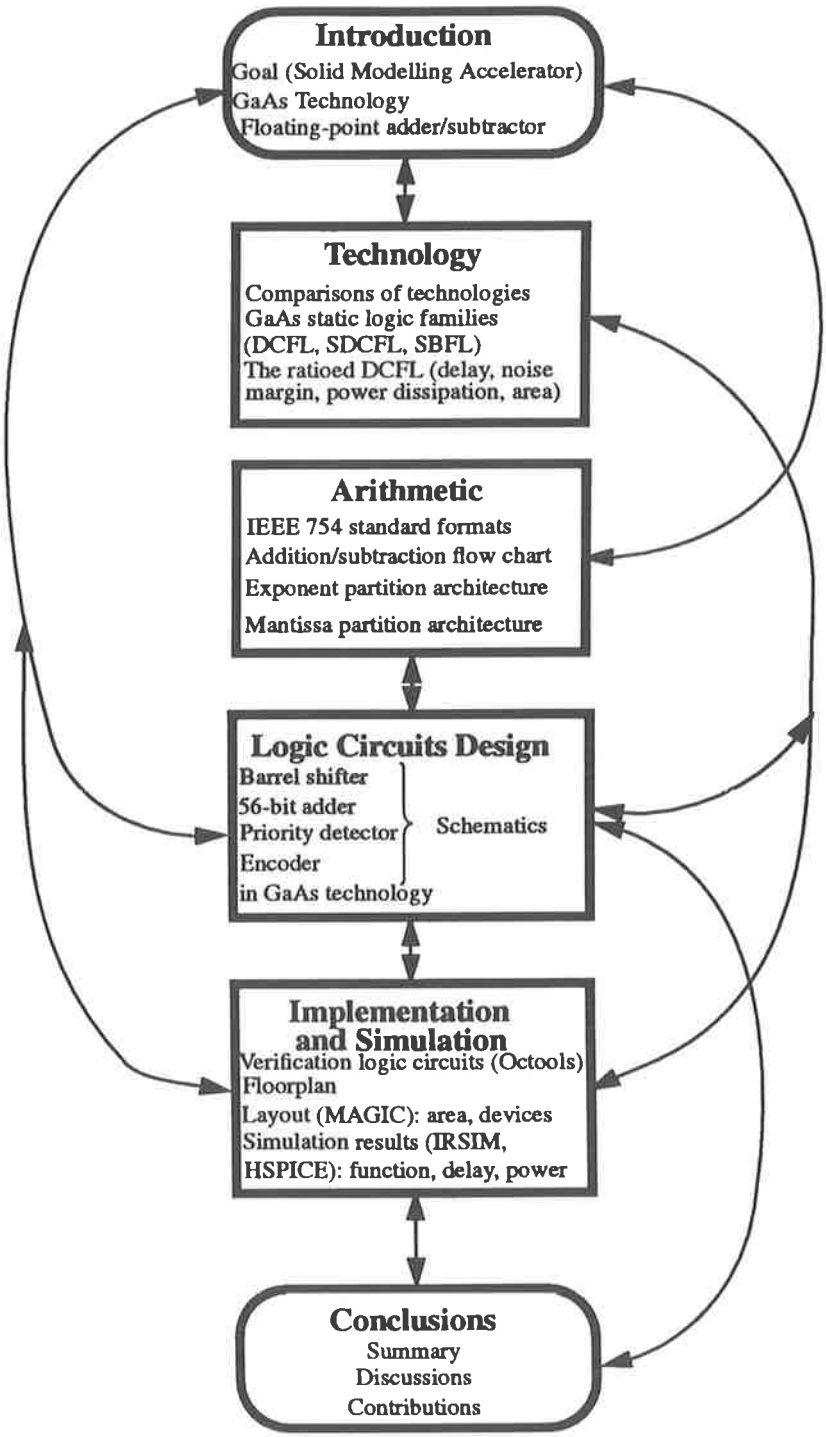


FIGURE 1-1. A relative structure of thesis among chapters.

search for the fastest three terminal device from the simplest direct-coupled logic (DCFL) to powerful buffers: source follower logic (SDCFL) and superbuffer logic (SBFL). Computer simulations are used throughout the third subsection to show the expected performances for different ratios and dimensions, study the effects of parameter variations at the gate level, and at last conclude the trade-offs among all parameters.

The 3rd chapter describes some fundamental floating-point definitions and the single precision and double precision IEEE 754 standard formats. The algorithm for the addition/subtraction operation is presented. We partition the floating-point adder/subtractor architecture into the exponent and the mantissa parts to deal with them individually.

Circuit analysis and logic circuit design begins in Chapter 4. This chapter contains an overview of the different approaches to the design of VLSI digital circuits for a specific function, explaining the advantages and disadvantages of each approach. As indicated, an efficient and acceptable way to speed up the whole process is to design applicable circuits in GaAs. The structure of the mantissa adder is explained in detail as a particular GaAs adder example. Also in this chapter, we emphasize how to normalize the mantissa result by means of combinational logic, and evaluate, develop and optimize an encoder to achieve the normalization requirement so that a 6-bit incrementer can be omitted.

Chapter 5 deals with VLSI circuit design techniques, implementations and simulations by means of the CAD tools **MAGIC**, **IRSIM** and **HSPICE**. In this chapter, we mention many practical problems in layout due to GaAs's limitations. Some special examples are: how to deal with a number of fan-in and fan-out, how to minimize noise effect, how to reduce heat generation, and how to reduce long wire delays. Also some design methodologies, such as how to regularize multi-bit functions (use PLA and etc.) to complete the layout easily and practically, and how to place each part and merge them together properly to attain minimum interconnection delay and area, are introduced. Experimental results demonstrate that each function has reasonable trade-offs between speed, area, and power dissipation. Finally, the performance of the overall system is discussed.

Chapter 6 contains the discussions and conclusions of this work. We examine the worst case delay of the mantissa adder in the simulation, several practical and optimal circuits, and some realistic approaches in the implementation.

In Figure 1-1, the topics discussed in each chapter are illustrated. We can search an objective with descriptions. The figure is intended to make it easier for interested readers to refer directly to the chapter describing the topics of interest.

Chapter 2 Ultra High Speed Gallium Arsenide Technology

Abstract: *Gallium Arsenide (**GaAs**) **MESFET** technology has been widely used in various fields, as **GaAs** technology has two significant advantages over silicon: higher electron mobility (high speed), and lower parasitic capacitance. There are, however, some disadvantages, like limited fan-in and fan-out capabilities, and low noise margins in some applications. Furthermore, fabrication costs are higher than those of **CMOS**. This chapter presents and analyses three typical static **GaAs** logic families: **DCFL, SDCFL** and **SBFL**. The influences of the size and ratio of transistors used in the **DCFL** gate with regard to speed, power dissipation, area, and noise margin, are also discussed.*

2.1 Introduction

Gallium Arsenide (GaAs) as a compound semiconductor has been studied and researched since the late 1960's [Lon90]. With many years of promoting the superiority of GaAs ICs over silicon ICs in speed, power, temperature range, and radiation hardness, many people began to think of GaAs ICs as exotic devices in a world of their own. In the late 1980's, U.S manufacturers of digital gallium arsenide circuitry, like Convex Computer Corp., Vitesse, TriQuint Semiconductor Inc., finally carried out the reality of a GaAs world [Cat90].

Within this decade, GaAs has emerged as the starting material for integrated circuits with one million or more transistors per chip. The technology today is firmly in the domain of high-performance, very large-scale integration, with chip clock rates reaching 500 MHz and above. More recently, Vitesse has announced that direct coupled FET logic for GaAs VLSI design has reached speeds of up to 800 MHz using a 0.5 micron, 5 level metal interconnects process [ZEN95] for a digital IC design. Importantly, the manufacturing cost is decreasing.

Using the high speed capability of GaAs for implementing digital integrated circuits, supercomputers and specialized high speed microprocessors have been successfully developed [Ras86][Hel89][Fox86].

Veljko Milutinovic in GaAs Microprocessor Technology [Mil86] cited one company's products as a particular example of a comparison between two 32-bit microprocessors implemented in both DCFL E/D-MESFET technology and CMOS technology. The result demonstrated that the 32-bit microprocessor which used GaAs was much faster than used CMOS in speed performance. Since we require a high speed processing for the Solid Modelling accelerator, not achievable using current CMOS technologies, we choose GaAs technology to implement the floating-point arithmetic. There has been little research reported on the implementation of floating-point arithmetic in GaAs. It makes the study more interesting.

There are some problems in GaAs VLSI design, such as fan-in and fan-out limitations, sensitivity to noise and so forth. To address these defects, before implementing the logic circuits, it is imperative to optimize geometrical parameters of the transistors at the gate level. Transistor sizing directly affects the overall performance

of the circuits and the system.

In this project the speed of operation is of critical importance. However, for proper operation of the circuits, problems associated with digital noise and the effects of temperature should also be addressed. These factors are traded off against each other, making the optimization process of the transistor sizing more difficult and critical.

Another technology-dependent choice made for this design is to choose which logic families to be employed. DCFL logic, due to its low power consumption and small size, is considered to be the most suitable family for large-scale integration among the many kinds of GaAs logic families [Tie94], even though it has low fan-in and fan-out capabilities. Many experiments have verified that the optimal design choice is the GaAs domino DCFL logic family [Cha95] [Lar86] [Roc90]. There are other logic families including SDCFL and SBFL that have high fan-out and fan-in capacities, but they exhibit a high power dissipation, complexity and large chip area requirement compared to DCFL.

This chapter is devoted to a preliminary discussion of GaAs technology developments, namely: contrasting advantages and disadvantages to CMOS, introducing three static ratioed logic families, compensating for transistor performances, and finally a summary.

2.2 Advantages And Disadvantages

Historically, by 1980 a technical and economic conflict appeared quite evident between GaAs and silicon technologies [Roc90]. The most effective application of GaAs technology is to produce VLSI performance which is not possible nor easy to attain with Silicon, namely its inherent high speed and low power capability. In contrast, CMOS is a very mature and predominant technology, which is applied widely in many fields due mainly to its low fabrication price, low power at low speed, and flexible design methodology. Compared to CMOS, GaAs DCFL technology has several considerable advantages:

- High speed. Due principally to a semi-insulating substrate structure in GaAs, it makes GaAs with the higher electron mobility, high peak electron velocity, and a

lower parasitic capacitance. For example, in 0.5 micron technology, the shortest gate delay in CMOS is about 150ps as opposed to 70ps for GaAs [Dey95].

- Higher radiation hardness.
- Higher tolerance of temperature variations.
- Uses fewer transistor. For instance, a 2-input NOR (NOR2) gate uses three transistors, but it uses four in CMOS.
- Normally, lower power supply voltage at 1.5V or 2V.
- No dynamic power consumption, and less power consumption in high speed environments. Figure 2-1 shows a trend of power and frequency of GaAs and CMOS. It can be seen that GaAs maintains a constant power at different frequencies. CMOS power, however, increases as frequency increases.

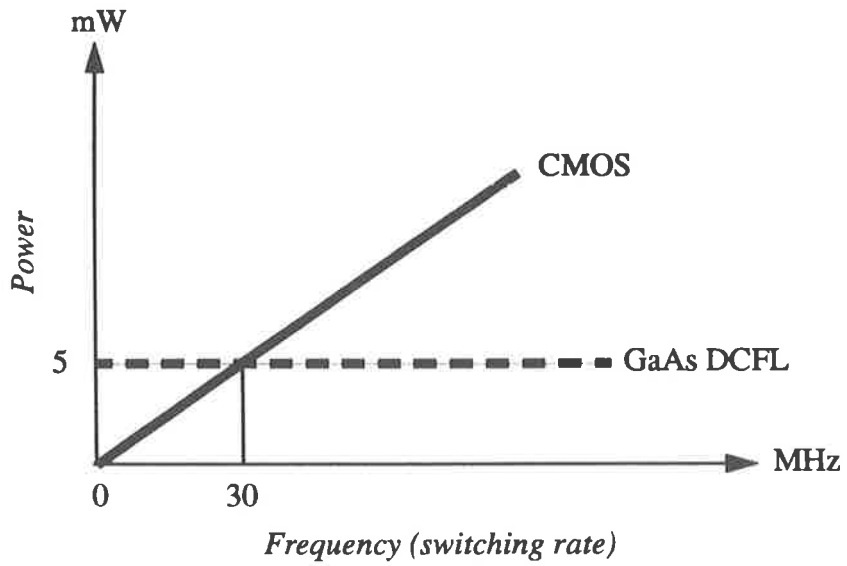


FIGURE 2-1. Power dissipation variation with frequency for two different technologies.

Obviously, GaAs technology, in contrast to CMOS, has some basic disadvantages [Win90] [Hel89]:

- Lower thermal conductivity.
- Restricted fan-out of about three gates and poor fan-in.
- Inverter, 2 and 3 input NOR gates are the only gates normally used in the design,

causing more complexity and inflexibility. NAND gates can not be used in GaAs [Hel89] because of a very low noise margin at the gate level.

- Higher noise sensitivity. The high voltage is typically 0.6V and the low voltage is 0.1V resulting in a low voltage swing. This drastically reduces the noise margin in DCFL gates. To deal with this problem, we must use techniques to minimize noise problems. This techniques often tend to increase the integration density.
- Higher cost. Due mainly to the scarcity of gallium and its inferior quality, and difficulty in manufacturing the gallium arsenide compound.
- Low yield.

Therefore, GaAs technology is expensive and has certain limitions which cause the design to be more critical. To achieve the high speed requirements in many applications, however, it is of value to employ this technology [Scl88].

2.3 The Static Logic Families of GaAs

The static and dynamic FET logics are two common types of options in design. The dynamic FET logics consume much less power than the static logics. However, main disadvantages of dynamic logic are reliability (noise effect) and clock requirements (2 phases clock). In addition, dynamic logic families require more transistors and power supplies. For example, in a two phase dynamic FET logic (TDFL) a negative voltage supply and two more transistors in an inverter are required [Lon92]. Although sometimes transistor count is higher, the transistor sizes are smaller than in static logic.

There are several different static logic gate configurations: buffered E/D logic, buffered FET logic, Schottky Diode FET Logic, differential pass-transistor logic and the simple E/D DCFL logic, all of which are now widely used in high-speed GaAs logic circuits. As buffered FET logic, Schottky Diode FET logic, and differential pass-transistor are based on the depletion MESFET, which requires a $-2V$ negative gate-to-source voltage (V_{ss}) to turn-off, they must have two power supplies. As a result of the negative voltage, the subtrate voltage may be changed (In the process that we have used $V_{sub} = 0.6 + Vss$), which leads to an increase in the transistor threshold voltage. These result in a cumbersome chip design, chip layout, system integration, and larger gate delays. For these reasons, we select static normally-off logic

families to design the combinational logic adder/subtractor, avoiding the need for a second power supply and negative voltage.

For convenience, static logic families such as buffered E/D logic including SDCFL and SBFL, and DCFL logic, are adopted. Here we shall talk briefly about these three static logic families.

2.3.1 DCFL

Direct-coupled FET logic (DCFL) [Roc90] uses both enhancement-mode and depletion-mode transistors in a circuit as shown in Figure 2-2. In practice, the DCFL

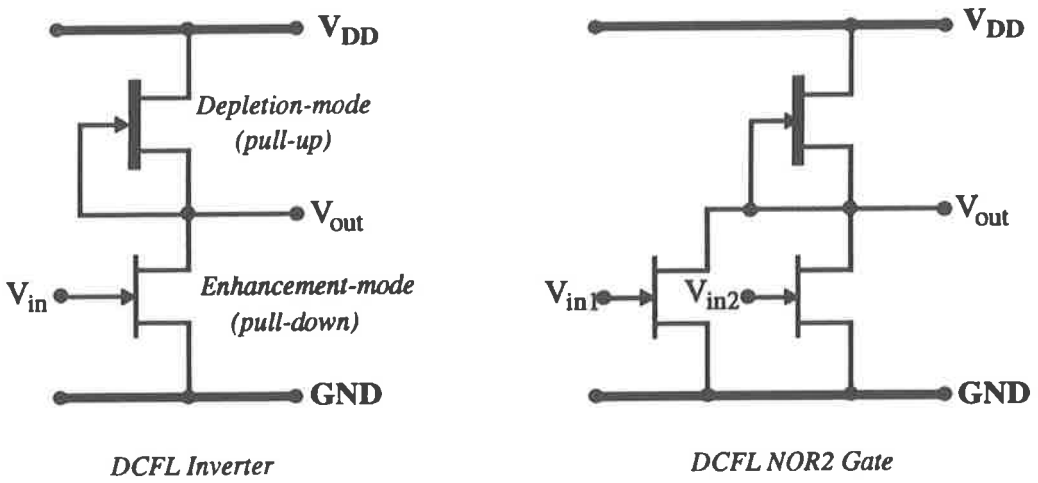


FIGURE 2-2. DCFL inverter and NOR gates.

circuit design is similar to that in a silicon NMOS process. The E-MESFET is normally-off, so there is no need for an additional power supply and level shifting.

In comparison with other logic families discussed here, i.e. SDCFL and SBFL, DCFL is simpler, has a higher density and a significantly lower power dissipation. In Section **2.4** the transistor level design of DCFL is described.

In Figure 2-2, two basic DCFL logic circuits, an inverter and a two-input NOR gate, are illustrated. Each gate consists of a depletion-mode load device (pull-up), and one or two enhancement-mode pull-down transistors. NOR gates with more than two

inputs can be constructed by adding more enhancement-mode transistors in parallel. However, there is a limitation in the number of inputs of a gate. The limiting factors are the increasing transistor leakage current, and the increasing gate delay. The number of fan-outs is also limited by the current driving capability, and the gate noise margin. Fan-in and fan-out affect both the speed and noise margin significantly [Scl88], however, the two buffered logic functions described below overcome these problems.

2.3.2 SDCFL

A Source Follower Direct Coupled FET Logic (SDCFL) [Esh91] can act as a buffer to improve the noise margin, and the fan-in and fan-out capabilities when driving a large fan-in and fan-out [Win90]. On the other hand, to control the delay caused by a large fan-out, suitable transistor sizes and ratios should be carefully selected. Referring to Eric Chu suggestion [Chu94] and our simulating results, a SDCFL inverter with optimized transistor sizes are illustrated in Figure 2-3. The sizes (W: width, L: length) of the transistors are in microns.

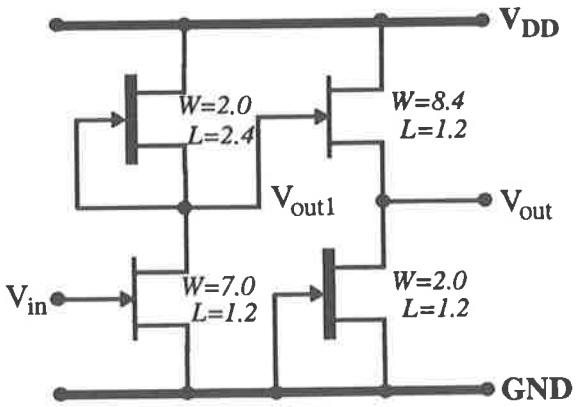


FIGURE 2-3. Logic gate level of an optimized SDCFL inverter indicating transistors sizes.

This improvement in the fan-in and fan-out phases comes at the cost of increased circuit complexity, and larger area. The SDCFL consists of two stages: the first stage is a DCFL inverter whose output is V_{out1}, and the second stage is the source follower gate whose logic output voltage V_{out} has the same state as V_{out1}.

Multiple input SDCFL NOR gates can easily be formed by adding parallel E-mode transistors in multiple input DCFL NOR gates.

There are two phenomena observed in SDCFL applications: the first one is that as the line capacitance C_A increases (driving long wire), V_{OH} and V_{OL} in the first inverter stage, which is similar to a DCFL gate but with different transistor sizes, remain constant. The V_{OH} in this stage is twice as high as typical high voltage (0.6V) so as to turn on a load and the source follower enhancement mode completely when the input V_{in} is low as shown in Figure 2-4. These results were obtained by simulation

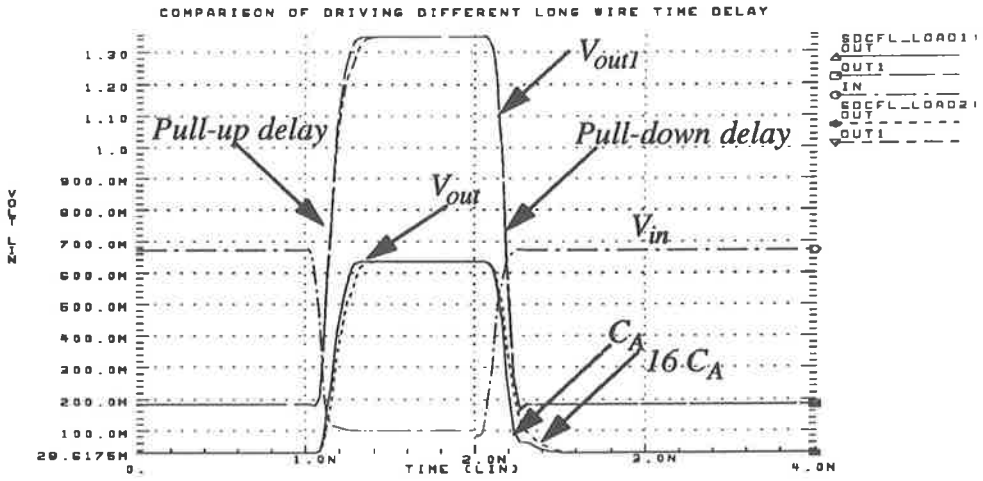


FIGURE 2-4. Capacitances load effect on speed in SDCFL.

of the circuit in Figure 2-3 using **HSPICE**. At the output of the source follower stage (V_{out}), V_{OL} is lower than the defined normal low voltage of 0.1V typical in DCFL gates, so there is a larger low noise margin in this stage than in a DCFL gate. The low noise margin is a very important factor to ensure that the next transistor turns off completely when the input is high.

The size of the pull-down transistor gate length is often fixed to the minimum gate length. Then transistor sizes are selected to make V_{OL} as low as possible to increase noise immunity.

Figure 2-5 shows the second phenomenon, which indicates that the SDCFL has

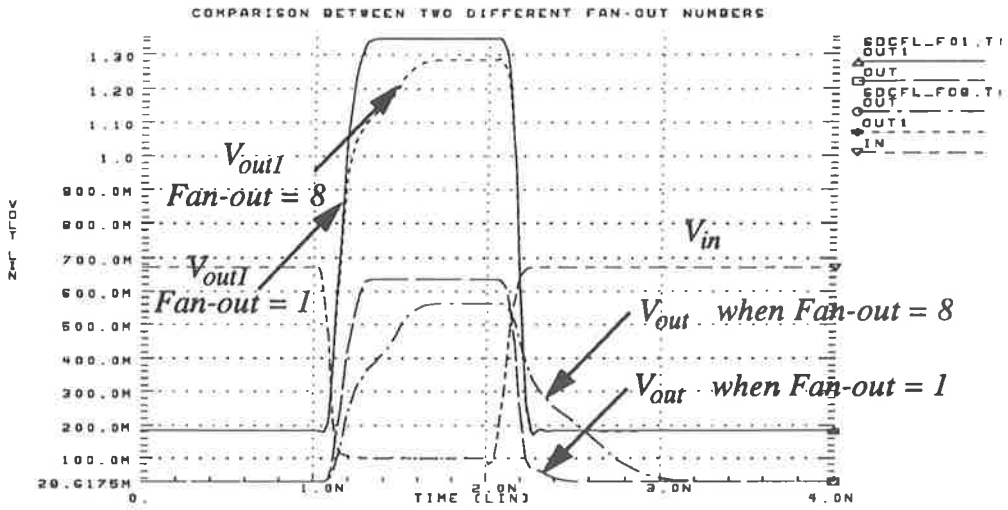


FIGURE 2-5. Fan-out effect in SDCFL.

good fan-out driving capability. With a fan-out of 8, the V_{OL} is sufficiently low (lower than 0.1V) and the output waveform is not distorted severely in the second stage. Even though the V_{OH} in both stages is lower than the case of a fan-out, and the second stage delay increases, the second stage V_{OL} remains invariant and is very low.

The SDCFL inverter consumes about 9 times more power than a normal DCFL inverter when using our selected transistor sizes of the normal DCFL (see **2.4.4**), and it also occupies a larger area. Therefore, in practical design, the SDCFL logic family is employed only as buffers to drive high fan-out and fan-in.

2.3.3 SBFL

A Super Buffer FET logic (SBFL) inverter has two stages: the input stage consisting of an E/D ratioed inverter and the output stage consisting of two non-ratioed enhancement transistors as shown in Figure 2-6. SBFL's speed is slightly slower than SDCFL, so that the inverter stage's pull-down width is decreased for increasing speed. The major part of the area of a SBFL inverter is occupied by the second stage non-ratioed and non-power consuming transistors. Driving a large fan-out, however, requires a large current ($I \propto W_{pu}$) in the second stage. Also its W_{pd} should be equal

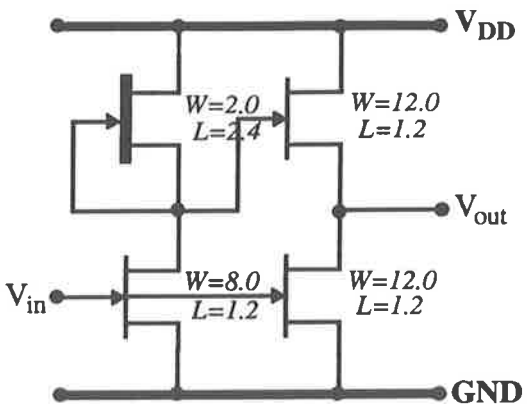


FIGURE 2-6. Schematic of optimized SBFL inverter.

to W_{pu} making the pull-up and the pull-down delays symmetric.

The output enhancement transistors are controlled by the input and its complement so that when one input goes high, the other goes low; that is one of them will be less conductive when the other becomes highly conductive. As a result, SBFL, like SDCFL, has very good low-logic noise margin. Also its high-logic noise margin is better than that of SDCFL. Nevertheless, the SBFL family incurs a longer delay than the SDCFL family when driving the same number of fan-outs. However, if the buffer is loaded with a large fan-out, the pull-up delay (the worst case delay) is almost independent of fan-out. The simulation results in Figure 2-7 with the same load indicate that compared with the labelled parameters in Figure 2-3 and Figure 2-6, a SDCFL inverter consumes about 5-6 times more power than a SBFL inverter as shown in Figure 2-7.

The SBFL inverter occupies approximately one and half times the area of the SDCFL inverter. As well as it is impossible to build a SBFL NOR gate by simply adding an E-mode transistor as for SDCFL NOR gates, since each input must drive two transistors which the area requirement is considerably larger than that of a SDCFL NOR gate. In addition, the two transistors present a capacitive load to the driving stage about three times as large as that presented by a SDCFL NOR gate. For these

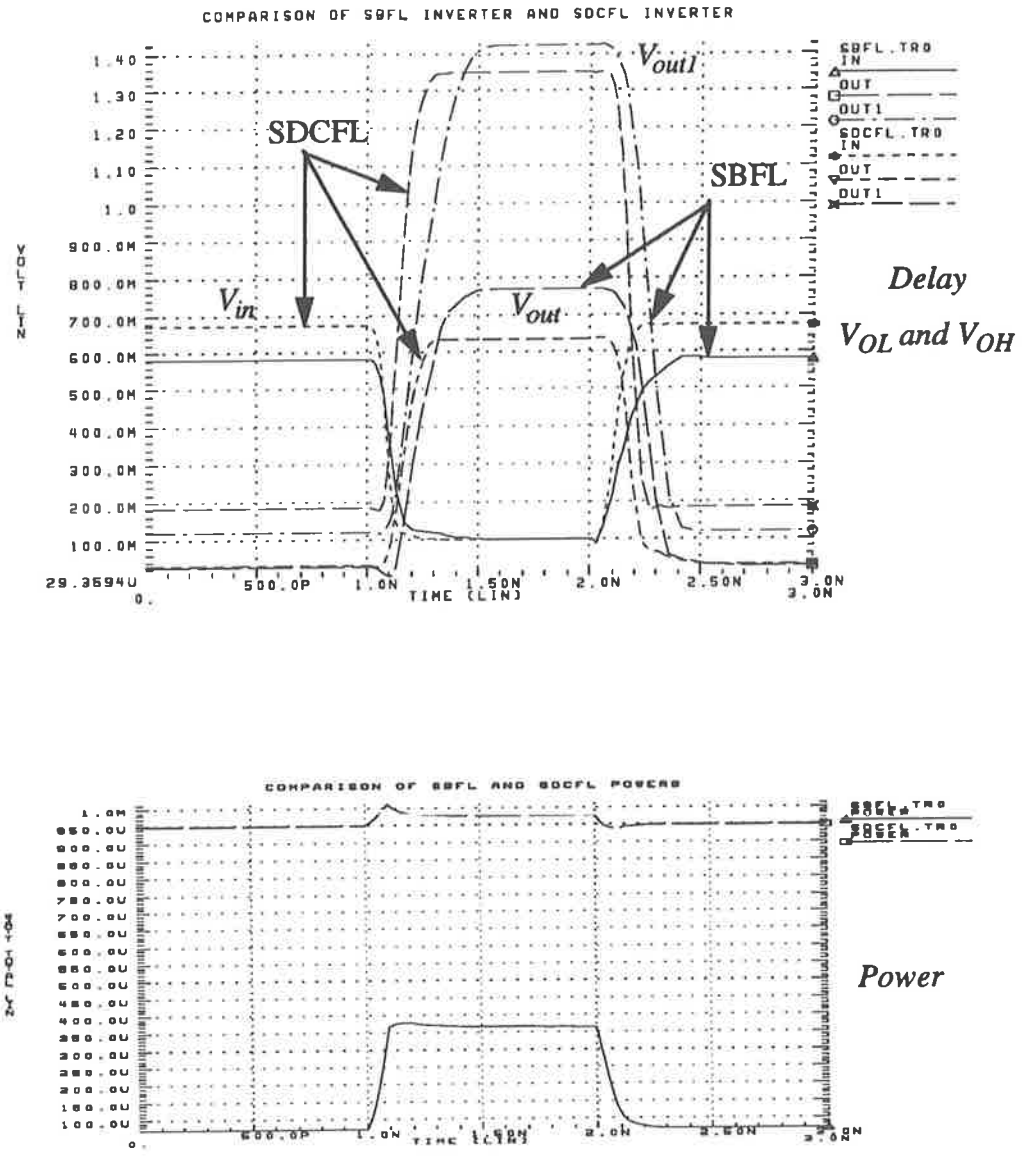


FIGURE 2-7. Comparisons of the SDCFL inverter and the SBFL inverter.

reasons, the SBFL logic family is normally used only for buffers and not for basic logic elements. Figure 2-6 shows an "inverting" SBFL buffer. If the inputs of two E modes in the second stage are swapped, the logic can accomplish a "non-inverting" buffer function.

The schematics and descriptions of the three logic families have been

explained. In Chapter 5, we will employ these three typical static logic families to implement the floating-point adder/subtractor.

2.4 The Selection of Transistor Dimensions And Ratio in DCFL

GaAs is a ratioed technology, meaning signal slopes and logic voltage levels vary with gate dimensions. For best performance, the transistor size and ratio should be carefully chosen before performing any layout design. Performance parameters that are of importance are the speed, noise margin, and power consumption. This subsection aims to discuss how different ratios and transistor dimensions influence performance in a DCFL gate.

The ratio of the D-mode and the E-mode transistors of an inverter or a NOR gate is defined as follows,

$$\beta = \frac{(L/W)_{pu}}{(L/W)_{pd}} \tag{2-1}$$

where $(L/W)_{pu}$ is the D-mode channel length to width ratio: $(L/W)_{pd}$ is the ratio for the E-mode. In DCFL logic, the size of the E-mode is such that it can conduct all the current provided by the D-mode (pull-up) at a drain-to-source voltage of approximately 0.1V, which is regarded as a logic-low state. In this way a valid logic-low state is generated since the threshold voltage of the enhancement device (transistor threshold voltage) is around 0.24V. Once the gate input exceeds this transistor threshold voltage, the E-mode is turned on. This is why the low output voltage V_{OL} must be as low as possible to minimize the probability of noise effecting the correct output voltage.

A logic-high voltage, on the contrary, is set automatically since the driving gate generates a current that forward-biases the gate-to-source junction of the driven enhancement device. Therefore, the typical forward voltage drop of a GaAs MESFET gate is 0.6V, this becomes the maximum value for a logic-high state.

Certainly, transistor performances are dominated by the device's dimensions. In this study, the pull-down channel length is chosen to be the minimum feature size of

$1.2\mu m$, to make the transistor threshold voltage maximum and maintain the high gate ratio β (see formula 2-1) to increase noise immunity. The width of the pull-up transistor is chosen at the minimum size ($2.0\mu m$) to gain a high transconductance β, a small size, and low power consumption.

2.4.1 Delay

Digital circuits use three-terminal devices used to switch the on and off voltage between two distinct levels. This operation requires the transfer of a certain amount of electric charge along connecting lines. Because of intrinsic characteristics of real devices, the input capacitance induces switching delays including a rise time delay (pull-up delay) and a fall time delay (pull-down delay). If we assume that ΔV is the voltage swing (typically 0.6V to 1V swings in GaAs), C_T the total load capacitance (due to fan-in, the pull-down transistor gate W_{pd} value and interconnection wires), and I_{max} is the maximum source current determined by the $(L/W)_{pu}$ and other transistor parameters, and also by the input resistance, then the minimum switching time (the gate delay) is:

$$T_{min} = C_T \times (\Delta V) / I_{max}. \tag{2-2}$$

From this formula, the design of integrated circuits for the fastest switching performance will tend to reduce all geometrical dimensions to minimize C_T, as well as use devices to be able to control large currents to charge the capacitance in a very short time with small voltage swings. Referring to formula 2-2, not only is the gate delay influenced by DC (the device characteristics ΔV and I_{max}), but also by AC characteristics (C_T).

Figure 2-8 shows four cases of delay for different ratios. The lowest ratio gate indicated by 'A' is associated with the smallest delay but has the highest V_{OL}. In order to obtain a large gate ratio to improve the noise margin, normally W_{pd} takes a much larger value than W_{pu}, which causes the rise time to take longer than the fall time. Thus, the rise time is regarded as the worse case delay time in the measurement.

The overall delay in a circuit, in fact, is produced by two major aspects: gate delay and interconnection line delay. The gate delay comes from the intrinsic delay, the load delay, parasitic capacitance, input capacitance of the driven circuits, resist-

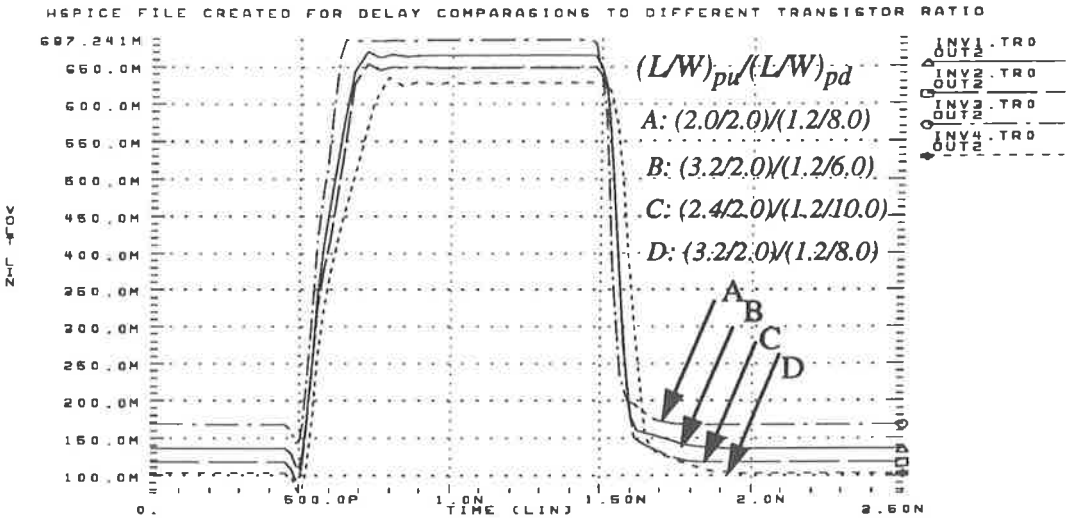


FIGURE 2-8. Delays under different ratios in DCFL.

ance and so on [Dey95]. The interconnection delay will be discussed in Chapter 5.

The simulation results in Figure 2-9 represent the fan-out = 3 delays for different ratios. A lower ratio transistor produces less delay when the fan-out increases,

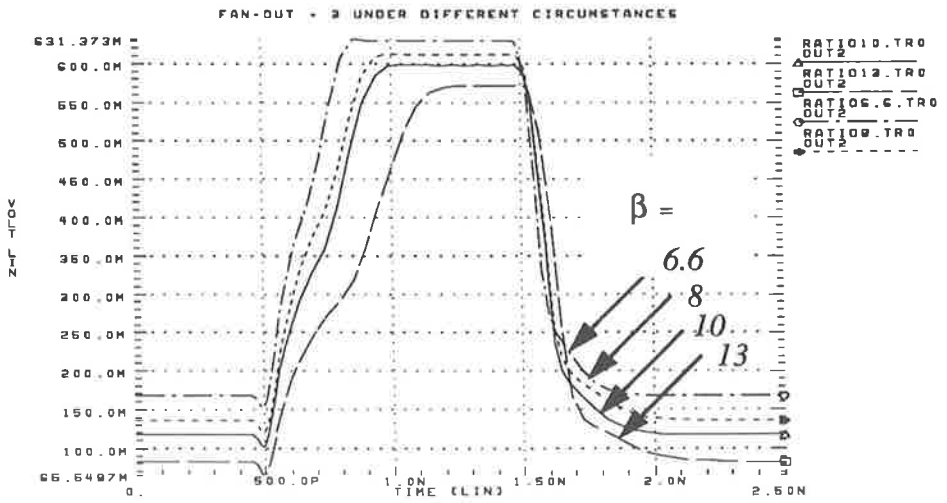


FIGURE 2-9. Effects with different ratios when the fan-out = 3 in the DCFL.

while the low noise margin decreases and the high noise margin increases. This is because we change the pull-up channel length to increase the ratio. If we maintain the same pull-down dimensions, the current with a higher ratio will be less than with a lower ratio, which causes both V_{OL} and V_{OH} to decrease simultaneously. Hence, high speed and good noise margin are conflicting requirements.

The minimum size gates in DCFL, SDCFL and SBFL logic families are inefficient at driving long capacitive lines. When speed is the objective, larger transistors must be used to restrict the interconnection-delay rise. Ordinarily, we enlarge W_{pu} and W_{pd} in the same ratio to obtain large currents ($I \propto W_{pu}$) to reduce the delay. Simultaneously, the power dissipation per gate increases so as to considerably limit the level of integration on a chip.

2.4.2 Power Consumption

Regardless of any technology used, the power consumption in a digital circuit is represented by [Her93]:

$$P_{dissipation} = P_{static} + P_{dynamic} + P_{leakage} \tag{2-3}$$

where P_{static} is static power; $P_{dynamic}$ is the dynamic term occurring due to the switching transient current and the charging and discharging of the load capacitance; $P_{leakage}$ is produced by the leakage currents, which is much smaller than the other two.

Generally, most of the power dissipation comes from static currents in the DCFL GaAs technology [Tie94], so that only the static power, which relies on V_{DD} and L/W_{pu} (DC effect), requires consideration. The following formula represents the static power per gate, which is in direct proportion to the pull-up maximum currents and supply voltage:

$$P_{static} \approx V_{supply} \times I_{pumax} \tag{2-4}$$

To reduce the power dissipation, the $(W/L)_{pu}$ ratio must be reduced. However, from (2-2) the gate delay will then increase. There is an obvious trade-off between speed and power at the gate level. Since the power consumption relies on the

static current and supply voltage, the gate power dissipation is fixed automatically after choosing the pull-up parameters. For a combinational circuit, the total power is approximately equal to the sum of the power dissipation of all individual gates.

The fourth chapter will describe some strategies to simplify the logic circuits aiming at reducing the number of gates. In our layout design, we have tried to use fewer gates to accomplish the same logic functions. The multi-bit multiplexer implementation is a particular example.

Another alternative for reducing the power dissipation is using dynamic GaAs circuits, but it is not currently a common practice. As already mentioned, due to the need of negative voltage and low noise immunity [Lon92], the practical circuits demonstrated so far have had very low integration levels in fabrication.

2.4.3 Noise Margin

The noise margin (or noise immunity) represents an allowance by which noise voltage on the input of a gate will not affect the output of the gate. This is a vital performance in a design. It can only be measured at a gate level.

The design yield is determined by the noise margin of a basic gate, as this ensures a correct operation of a logic gate in a given circuit environment. In general, the noise margin should be large enough to prevent defective operation due to static (resistive) and dynamic (capacitive inductive) noise as well as variation of the process parameters (process noise). Obtaining a large enough DC noise margin needs a good logic swing defined by the V_{OL} and the V_{OH} output levels.

One method of estimating the noise margins is given by low noise margin (NML) and high noise margin (NMH), which can be simplistic [Roc90] [Lon90]:

$$\begin{aligned} NMH &= V_{OH} - V_{IH} \\ NML &= V_{IL} - V_{OL} \end{aligned} \tag{2-5}$$

where V_{IH} is the high input threshold voltage, and V_{IL} is low input threshold voltage, both values of which are desirably equal midway in the logic swing. The NMH and the NML are desired to be equal as well.

The logic swing of a digital technology is imposed by the active device and logic gate structure. Hence, the noise margin magnitude at the gate level is determined by the transistor ratio, the fan-in, and fan-out. The effect of the ratio on V_{OH} and V_{OL} is represented in Figure 2-10. Measuring conditions are illustrated in Figure 2-11.

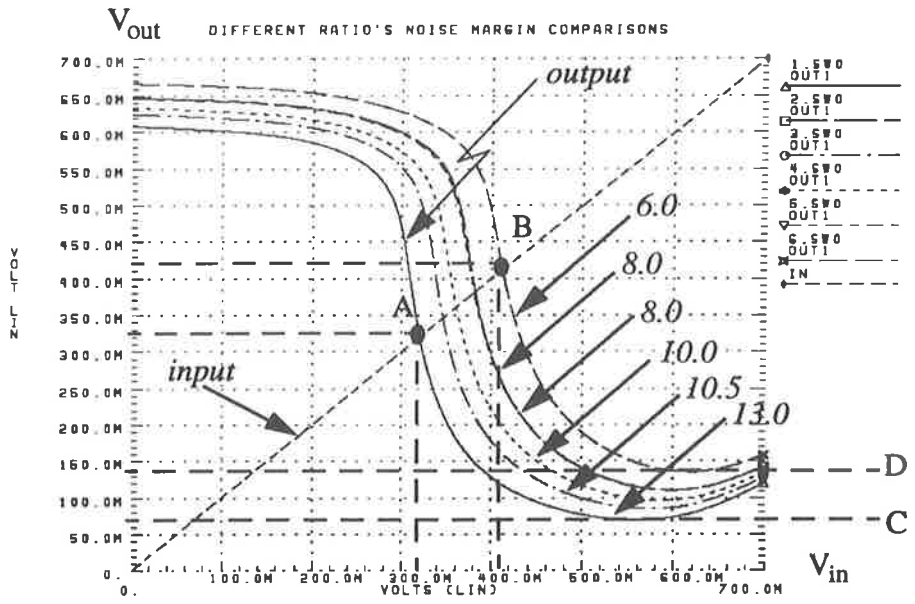


FIGURE 2-10. Schematic of low and high outputs under different ratios.

The diagram shows that regardless of the parameters of the pull-up and the pull-down, both V_{OH} and V_{OL} maintain consistency with the same ratios (for example, two ratios all are equal to 8 but they have different transistor parameters). From Figure 2-10, we can see as well that the logic threshold voltage increases as the ratio rises (from point A moving to point B). The gate with a higher ratio has the same symmetry as a lower ratio between the low noise margin and the high noise margin. The output of the highest ratio is with the lowest low noise point C compared to point D which has the lowest ratio. Even so, considering speed and area factors, it is not possible to choose very high ratio transistors for design of high speed arithmetic operation. For ensuring a value of noise margin, normally, there are two measuring methods: slope and square [Roc90] [Wes93] [Lon90]. For an accurate measurement,

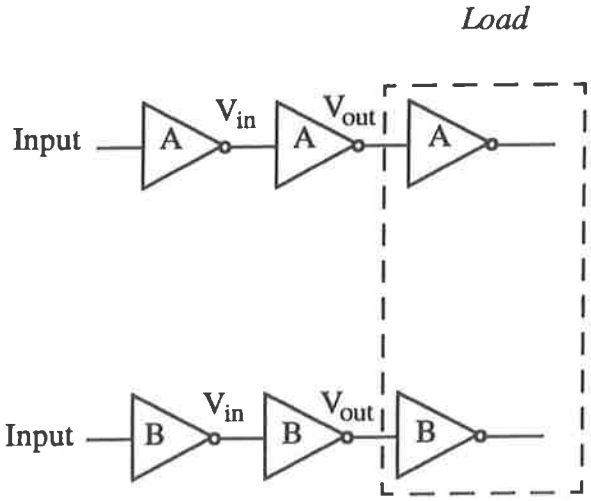


FIGURE 2-11. Measuring noise margin conditions.

slope method is used in CMOS technology as the static transfer curve of CMOS is very straight, but square method is used in GaAs technology.

To obtain a low V_{OL}, we can increase the width of the E-mode to attain a high ratio, but it will use a larger area in the layout. Figure 2-12 and 2-13 show two different cases: (1) to change the pull-up parameters with the same pull-down. Both V_{OH} and V_{OL} vary with the different pull-up ratios. (2) to change the pull-down dimensions under the same pull-up dimensions. The pull-down variation affects V_{OL} exclusively.

On the other hand, other factors that affect the noise margin are the fan-in and fan-out. In fact, the fan-out only affects V_{OH}, but not V_{OL}. If a NOR4 gate (fan-in = 4) drives only an inverter or a NOR gate, the effect of high fan-in on the noise margin is very slight, which will be demonstrated in **5.4.2**.

In general, if V_{OL} is sufficiently low, V_{OH} will be high enough to cause a gate current to flow in the second stage. In the selected GaAs process, the simulations have shown that when an inverter input voltage, with one fan-out, is higher than 0.35V, the transistor will be able to turn on (see Figure 2-10). As a result, the GaAs device is

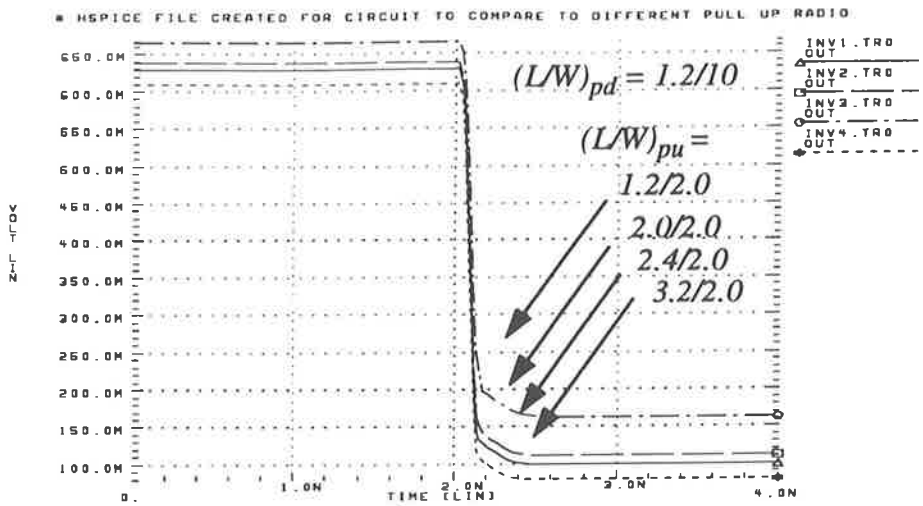


FIGURE 2-12. The effect to the output voltage with the constant pull-down parameters.

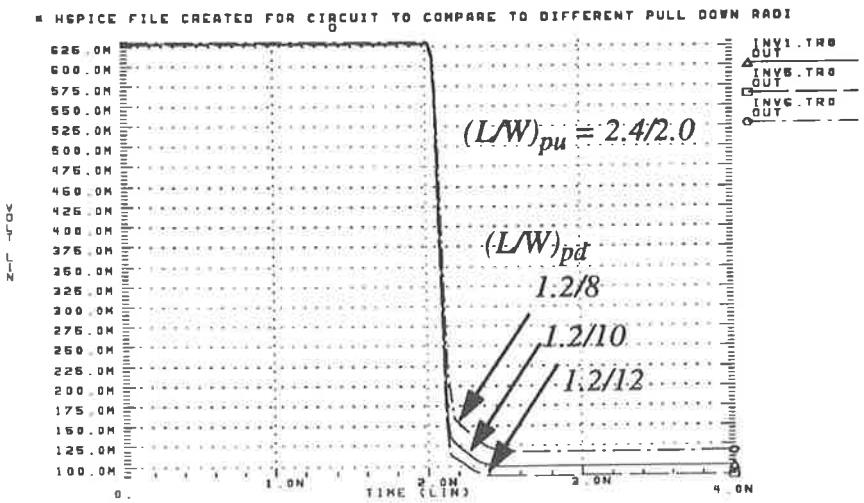


FIGURE 2-13. The effect to V_{OL} with the constant pull-up parameters.

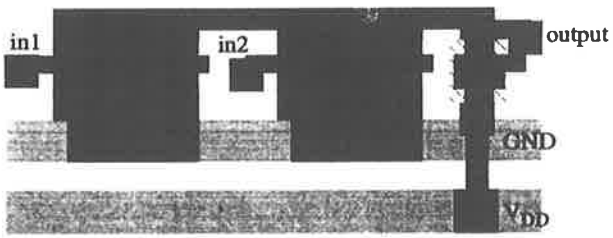
easy to turn on.

The simulation result from Figure 2-4 demonstrates that when driving different wire lengths, the output high and low voltages are maintained equal, i.e. the noise margin is not influenced by any AC effect.

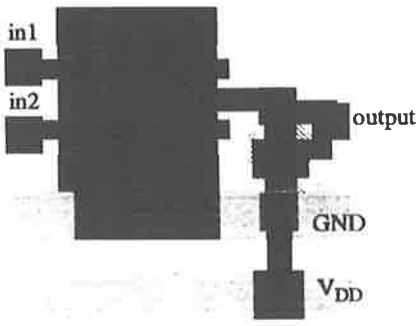
2.4.4 Area

To a large extent, the VLSI area, which is immediately related to cost, can be traded off against production yield. The area of a gate depends on the transistor size, normally the pull-down width size, since the pull-up size is usually smaller. However, if the pull-down width is too small the noise margin will be poor and yield will be poor as well. Meanwhile, influence of power supply and ground noise should be kept in mind.

Once the ratio has been fixed, there are several types of design styles in terms of ring-notation[Esh91] methodology which are easy to structure and improve the reliability of mask layout. Two mask layout styles are illustrated in Figure 2-14.



(a)



(b)

FIGURE 2-14. Two styles of DCFL NOR2 in mask layout with the same gate ratio.

Figure 2-14 (b) saves about 25% in area compared with Figure 2-14 (a). However, since the metal connected to GND and the source device overlaps active devices in Figure 2-14 (b), the GND line noise, whose magnitude can easily exceed the signal voltages, may increase to unacceptable levels. To ensure reliable operation, we use the mask layout of Figure 2-14 (a) in the design.

After considering the trade-off discussed above, the process parameters are selected as $(L/W)_{pu} = 2.4/2.0$ and $(L/W)_{pd} = 1.2/10.0$, giving $\beta = 10$. The length of the pull-up is expanded from the minimum size of 1.2 to 2.4 to obtain a higher ratio and less power, but at some cost in speed. The measured results of an inverter show that the worst delay is about 90 ps, the area is 15.8 by 18.9 square μm, and the average power dissipation is 0.085 mW.

2.5 Summary

The major advantage of GaAs technology is that it has high speed performance, even though there are some difficulties in implementations.

In this chapter, the pros and cons of the commonly used gate arrangements were considered in terms of design constraints, such as power dissipation, noise margin, speed, and area at the gate level. In Chapter 5, we will use the selected gates to implement the required logic functions, and then simulate the performance of the overall circuits.

Different transistor ratios cause different performance. Three compatible and ratioed, normally-off classes of logic, DCFL, SDCFL, and SBFL have been optimized, taking into account the effect of process spread, and will be employed in the present work. To minimize heat generation and cost, the implementation will predominantly use the DCFL logic family. Although logic circuits based on the E/D DCFL family are simple to design, and efficient in area and power, they have low fan-in and low fan-out capabilities, which affect both speed and noise margin. SDCFL and SBFL will be used as buffers to improve these capabilities. We can use a low ratio gate to increase the speed as well. On the other hand, to deal with the delays caused by long connecting lines, the gates of all three logic families can be enlarged in accordance with the same ratio.

An interesting property of the DCFL is that the power dissipation of the logic gate is independent of logic state or frequency of operation. Once the D-mode parameters are fixed, the current on the power supply buses is constant, so the power consumption is constant. The overall power in a chip is approximately directly proportional to the gate count. Much work will be described in later chapters in order to reduce the number of gates to minimize the power consumption.

The ratio of the pull-up and the pull-down plays a major role in determining the noise margin. Simulation results show that the gate with a high ratio provides a good noise margin, resistant to the various types of noise: such as (1) process noise (random variations of the electrical and geometrical process parameters), (2) static logic noise (DC voltage drops in the ground and power buses, as well as along interconnect lines), and (3) dynamic logic noise (from the inductive behaviour of the ground and power buses as well as the capacitive coupling between interconnection lines). In addition, the fan-in and fan-out numbers alter the noise margin slightly. To minimize the effects of noise, the ring notation methodology is used.

Optimum ratio choice will result in minimum delay, low power and high noise margins. There are two approaches to increase the circuit reliability: expand the pull-up length, or extend the pull-down width. Both result in a speed reduction.

In order to maintain the pull-up dimensions and extend the width of the pull-down to improve low noise margin, the area will increase and the speed will reduce. On the other hand, when shrinking a transistor size (choose small sizes for L_{pu} and W_{pd}), the gate will occupy a smaller area and the delay will be smaller. However, the noise margin will be poorer and the power dissipation will be increased.

Chapter 3 Floating-Point Fundamentals

Abstract: *There are two different types of real number representations which are commonly used in computer arithmetic: fixed-point and floating point. In applications requiring a representation of numbers with high precision and a large range, floating-point representation is more suitable, although arithmetic operations can be more difficult. This chapter presents some essential floating-point principles as well as describing the* ***IEEE 754*** *Standard Formats. Moreover, a floating-point addition/subtraction operation procedure, and the architecture of* ***exponent and mantissa*** *processors needed to implement it, are introduced.*

3.1 Floating-Point Principles

In computer arithmetic operation, such as addition, subtraction, multiplication and division, there exists a large number of datatypes that may be represented. At the same time, one number can be represented in many various ways in different types of machines. This section discusses the most common data representations of non-integers and the IEEE 754 Standard Formats used in computer arithmetic.

3.1.1 Data Representations

A representation of numbers may be classified as either fixed-point or floating-point. On many occasions the range of numbers expressed may be very large. For example, the speed of light is 300, 000, 000 meter/second (in decimal), which we generally write as 3×10^8 meter/second; the permittivity of free space is 0.0, 000, 000, 000, 000, 885 Farad/cm, which can been written 8.85×10^{-14} Farad/cm.

A fixed-point is to use integer arithmetic and simply imagine the binary point somewhere other than just to the right of the least significant digit. Adding two such numbers can be done with an integer addition. A floating-point representation, in contrast to a fixed-point representation, increases the range of numbers, since it can represent both very large numbers and very small fractions [Sta93] [Sto80]. If the data is A, it can be represented by the formula,

$$A = \pm f \times X^{\pm e} \tag{3-1}$$

where X is any radix like 2, 8, 10, 16...; f is the fraction or the mantissa; and e is the exponent.

In the floating-point representation the range is increased considerably over the fixed-point representation. Nevertheless, the expansion in the range may incur a penalty of reduced accuracy. This is because the exponent uses 11 bits in IEEE 754 double precision format which could otherwise be used for increased precision, if the number of bits is fixed. Therefore, there is a trade-off between precision and range [Pat94]. To solve this problem and provide both a wide range and high accuracy con-

currently, more bits are sometimes required. Another advantage of a floating-point computer is its ease of handling a scale factor [Hill73] [Fuj92]. At input, regardless of floating-point or fixed-point, a scale factor must be assigned to each number to convert it to a fraction. To the number 753, there will be a scale factor of 1000 assigned and the number will enter the machine as 0.753. A floating-point computer is comparatively more complicated in hardware design than a fixed-point one and the execution of computations may take longer [Man88] because there are two parts, the exponent and the mantissa to be taken into account. Additional cost is also incurred [Hill73].

Conventionally, an integer may be expressed by several bytes in computers, each byte containing 8 bits. The mantissa and the exponent of a floating-point number can be represented by several binary bytes as well [Bar92] [Mur90]. The most common integer representations in computer arithmetic are:

- **Signed-magnitude:** The sign bit is the leftmost bit and it denotes the sign of the number (0_2 for positive numbers, 1_2 for negative numbers). The remaining bits denote the magnitude of a number. For example: $7_{10} = (0111)_2$; $-7_{10} = (1111)_2$, with the leftmost bit in the 4 bit binary number being the sign bit. Notice, however, that the positive value of 0 is not equal to the negative value of 0. In IEEE 754 Standard Formats, the mantissa expression uses this representation.

- **One's complement:** The leftmost bit indicates the sign of the number just as in the signed-magnitude representation; 0_2 is a positive number and 1_2 is a negative one. If the sign is 0_2, the n-bit number can be represented by the binary magnitude; if the sign is 1_2, the magnitude of the number is equal to the value of the positive n-bit number obtained by inverting each bit. Generally speaking, one's complement in binary notation is implied by changing 1 into 0 and 0 into 1, bit by bit. For example, if the number is 11001, its one's complement is 00110. It is worth mentioning that, again in one's complement representation, the positive value of 0 does not have the same representation as the negative value of 0.

- **Two's complement:** Here a positive number has the same representation as the

corresponding signed-magnitude and one's complementary positive number. However, the negative binary two's complementary number results from complementing the bits and adding 1, that is the one's complement operation + 1. The leftmost bit in the two's complement representation is simply a sign bit. In 4 bit representation, the number -6_{10} can be obtained from $+6_{10}$ $(0110)_2$ by inverting $(0110)_2$, to yield $(1001)_2$, and adding 1 to finally yield $(1010)_2$ in two's complement representation. As another example, suppose we want to represent -0_{10} in two's complement. We first take $+0_{10} = (0000)_2$, complement the bits $(1111)_2$ and add 1 to gain $(0000)_2$. It is clear that in two's complement -0 and +0 have the same representation. As well the MSB does not represent a sign, so there is no distinction between the negative number and the positive number in two's complementary representation. Because of these advantages and the advantage in implementation of arithmetic, most computers use the two's complement arithmetic form. When adding two numbers with different signs, the two numbers can be added directly, and it is not necessary to consider the subtraction operation in the design. For convenience, when adding/subtracting two mantissas (or fraction) in floating-point operations, we will convert the signed-magnitude mantissas into two's complement representation before adding them.

- **Bias (Excess-n):** A biased number is also called an excess number [Man88] [Bar92]. It is often used for an exponent representation in a floating-point computer [Sto80]. If K is a binary number, the excess-n representation is the unsigned representation of $n + K$, where n is the bias. The number $n + K$ is called *a biased number*. Table 3-1 illustrates the 4-bit excess-3 numbers. $(11)_2$ that is decimal 3 represented as 0_{10} in a biased representation. For an 11-bit exponent in double precision floating-point format, according to IEEE 754 Standard, the bias is $(1023)_{10}$. Assuming that the real exponent is the minimum exponent $-\left(2^{10}-1\right) = -1023$, its excess-1023 representation is 0. Hence after adding the excess number, all exponents become positive. The biased exponent representation has two significant advantages: one is that the biased numbers are only positive numbers, consequently it is easier to compare the magnitude of two numbers without signs. Accordingly, the smallest feasible exponent, which is the most negative exponent, is zero with a biased representation [Man88].

Table 3-1: 4-bit binary number representation of a decimal number.

Decimal	Signed Magnitude	One's Complement	Two's Complement	Excess-3
12	----	----	----	1111
11	----	----	----	1110
10	----	----	----	1101
9	----	----	----	1100
8	----	----	----	1011
7	0111	0111	0111	1010
6	0110	0110	0110	1001
5	0101	0101	0101	1000
4	0100	0100	0100	0111
3	0011	0011	0011	0110
2	0010	0010	0010	0101
1	0001	0001	0001	0100
0	0000	0000	0000	0011
-0	1000	1111	----	----
-1	1001	1110	1111	0010
-2	1010	1101	1110	0001
-3	1011	1100	1101	0000
-4	1100	1011	1100	----
-5	1101	1010	1011	----
-6	1110	1001	1010	----
-7	1111	1000	1001	----
-8	----	----	1000	----

3.1.2 IEEE 754 Standard Formats

The above subsection has described common number representations in computer arithmetic. In practical applications, a floating-point number may be expressed in many forms. For example the decimal number 3.2, can be represented as

0.0032×10^{3}, 0.032×10^{2}, 32×10^{-1}, 320×10^{-2} and in other forms. These are called non-normalized numbers. Normalized numbers require the decimal point of the floating-point number to have a fixed position, that is the decimal point is to the left of the most significant digit. In this case, the fraction satisfies: $1 > F \geq 1/X$, where X is a radix. In this example, the number 3.2 in normalized form is then represented by 0.32×10^{1}. It can be seen that the fraction must be between 0.1 and 1. If numbers are smaller than the smallest normalized number, then we call these numbers denormalized numbers. The notation of binary normalized number is different from the decimal. In a binary representation, the form of the normalized numbers is

$$1.xxxxxxxx_2 \times 2^{yyyy}. \tag{3-2}$$

Also, more than twenty years ago, many computer manufacturers had their own format in a floating-point computer, such as in the CDC CYBER 170 which used a 60-bit word consisting of a 48-bit mantissa, an 11-bit exponent (bias is 1024), 2 as base and a bit sign of the mantissa. In the IBM Corporation System/370, the 32-bit single precision floating-point number had a 24-bit mantissa, a base of 16, a 7-bit exponent (excess-1024) and a one bit mantissa sign and so forth [Tan90] [Bar92].

In order to unify the various representations and forms of floating-point computers, the IEEE published "The IEEE 754 Standard Formats" in 1985 [IEEE754] [Sta93] [Pat94]. In these documents a 32-bit single-precision and a 64-bit double-precision are standardized. Figure 3-1 shows these standard formats: they are an 8-bit exponent (single precision) and an 11-bit exponent (double precision) with the biased representation; 2 is the exponent base which is implied; a 23-bit mantissa and a 52-bit mantissa; the most significant bit is the sign bit of the mantissa, if $MSB = 0$, the mantissa is positive; if $MSB = 1$, the mantissa will be negative. In the IEEE 754 formats, the mantissas are expressed in signed-magnitude. There is a hidden bit at the leftmost position of the real mantissa, this bit is always 1 and is implicit.

Excess-1023 is used as a bias in the double precision floating-point exponent field [Man88] [Pat94]. The real exponent range is from -1022_{10} (minimum exponent) through to $+1023_{10}$ (maximum exponent). Consequently, the biased exponent range is $1 < e < 2046$. Normally $e = E + 1023$, where E is the actual exponent value.

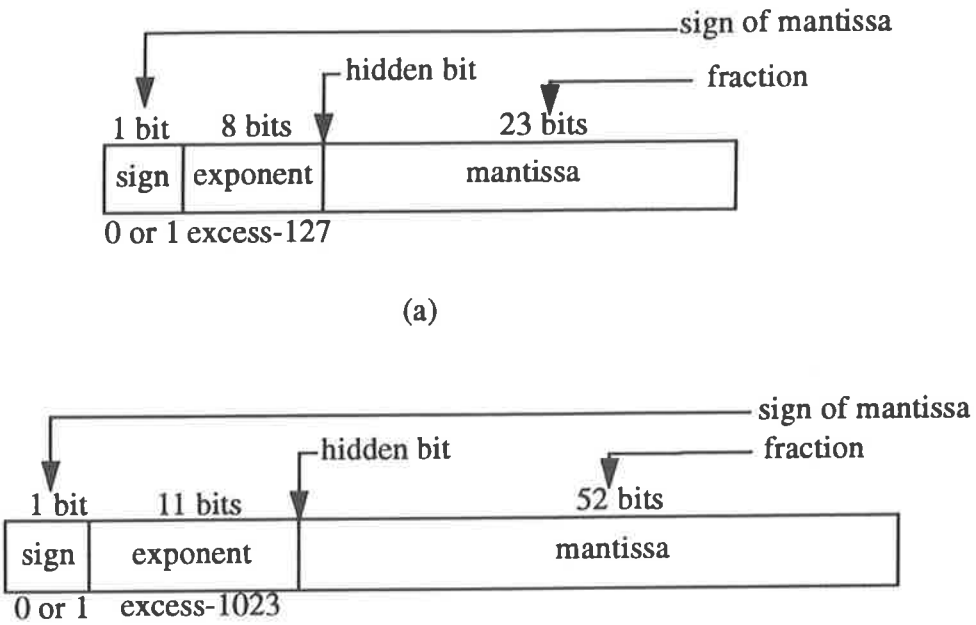


FIGURE 3-1. IEEE floating-point formats. (a) single precision. (b) double precision.

Let us consider four special cases of this representation:

1. If $e = 2047$ and $f = 0$, the number denotes plus or minus infinity. The sign bit indicates whether number is positive or negative.
2. If $e = 2047$ and $f \neq 0$, the representation is called NaN (not a number) and it is without any sign.
3. If $e = 0$ and $f = 0$, the number is plus zero or minus zero, depending upon the sign.
4. If $e = 0$, and $f \neq 0$, the number is a denormalized number which is less than the minimum normalized value in IEEE formats. Table 3-2 shows the numerical characteristics of the IEEE floating-point numbers.

We have described before that the hidden bit is implied as being 1 inserted left-

most of the mantissa. Therefore, the real range of the mantissa is from 1 to 2 (excluding 2). The hidden bit expands the one bit mantissas precision [Sta93] [Sto80], but

Table 3-2: The numerical characteristics of the IEEE floating-point.

Item	Single precision	Double precision
Bits in sign	1	1
Bits in exponent	8	11
Bits in fraction	23	52
Bits, total	32	64
Exponent system	Excess 127	Excess 1023
Exponent range	-126 to +127	-1022 to +1023
Smallest, normalized	2^{-126}	2^{-1022}
Largest, normalized	approx. 2^{+128}	approx. 2^{+1024}
Decimal range	approx. 10^{-38} to 10^{+38}	approx. 10^{-308} to 10^{+308}
Smallest, denormalized	approx. 10^{-45}	approx. 10^{-324}

this bit does not occupy a position in the computer arithmetic operation. Since the leading bit of the fractional mantissa is always 1 for a normalized number, there is no need to store it in the computer [Cle85].

A double precision mantissa number is represented in 52-bits in the computer. However, arithmetic must be performed in at least 53 bits to include the hidden bit. Table 3-3 shows some examples of mantissa to decimal conversion.

Table 3-3: Fraction field values and the corresponding significant examples.

Fraction (f)	Significand (1.f)	Decimal Equivalent
100.....0	1.100.....0	1.50
010.....0	1.010.....0	1.25
000.....0	1.000.....0	1.00

In significand, the hidden bit is represented. When adding/subtracting two num-

bers, if the numbers are represented in a normalized form, the result can have greater accuracy [Sto80]. The following decimal example illustrates the advantage of normalization in the addition of two 6-digit floating-point numbers:

Un-normalized: 0.000500×10^{0} and 0.000030×10^{-2}, in adding these numbers, first of all we align the number with the smaller exponent and then add them:

$$\begin{array}{r} 0.000500 \text{ x } 10^{0} \\ +0.000000 \text{ x } 10^{0} \\ \hline 0.000500 \text{ x } 10^{0} \end{array}$$

Normalized: In order to normalize a number, the first fractional position must be a nonzero digit like 0.500000×10^{-3} and 0.300000×10^{-6}. Assuming that we are to add the two data, first normalize, then align the numbers and add them:

$$\begin{array}{r} 0.500000 \text{ x } 10^{-3} \\ +0.000300 \text{ x } 10^{-3} \\ \hline 0.500300 \text{ x } 10^{-3} \end{array}$$

To compare the un-normalized case with the normalized one, the result of adding the two normalized numbers has more significant digits. There are other advantages in using normalized numbers: it simplifies exchange of data that includes floating-point numbers, and it simplifies floating-point arithmetic algorithms which can assume that numbers will always be in this format. The following expression gives the value of a binary number from a double precision representation,

$$\left(-1^{s}\right)\times(1\cdot f)\times2^{E} = \left(-1^{s}\right)\times(1\cdot f)\times2^{e-1023} \qquad (3\text{-}3)$$

where s is the sign (0 or 1),

f is the fraction, that is a 52-bit mantissa, the range is $0 \le f < 1$,

E is the real exponent, and

e is the biased exponent, $e = E + 1023$.

3.2 Descriptions of An Architecture for Addition/ Subtraction

We have described how data is represented in floating-point computers. Now we shall discuss the floating-point number addition/subtraction procedure.

3.2.1 Additions/Subtraction Arithmetic Operation Principles

In floating-point arithmetic, addition and subtraction are more complex than multiplication and division [Sta93]. Unlike a fixed-point addition/subtraction, a floating-point addition/subtraction must consider both the exponent part and the mantissa part. Not only does a floating-point addition/subtraction treat the exponent and the mantissa separately, like in a floating-point multiplication and division, but also the exponent difference of a floating-point addition/subtraction influences the mantissa addition.

Assume that it is required to add or subtract two decimal floating-point data $A = 548 \times 10^1$, and $B = 427 \times 10^{-2}$ by hand. The mantissas of A and B, quite clearly can not be added/subtracted immediately because of the different exponents. If adding/subtracting them, the first thing to do is to make their exponents equal. The difference between these two exponents is $1 - (-2) = 3$. Thus the B mantissa with the smaller exponent must first be shifted by 3 bits from right to left to make the exponents be equal, then they are added, as: $548 \times 10^1 + 0.427 \times 10^1 = 548.427 \times 10^1$.

Notice that in order to make the exponents equal, the mantissa having the smaller exponent is always shifted [Cle85] [Sta93] [Pat94], as shifting the mantissa with the larger exponent causes significant digits to be lost from the shifted number. For example, there are two data with a 6 bit mantissa, $C = 0.101010 \times 2^1$ and $D = 0.101010 \times 2^{-2}$. If adding them, we shift C having the larger exponent 3 bits from right to left, after shifting to equalize D's exponent, $C = 101.101000 \times 2^{-2}$. Unless we allocate more bits to the representation of C, C would become 0.101000×2^{-2}, and the highest 3 bits are lost. In shifting D with the smaller expo-

nent 3 bits from left to right, D is converted to 0.000101 . The lowest 3 bits will be lost. The mantissa having the smaller exponent is shifted to the right thus increasing the exponent to that of the other number. This results in a minimum loss of precision since only the least significant bits of the mantissa of one of the operand are lost. If the mantissa of the higher exponent is shifted left to decrease it to the value of the other number, the leftmost bits of this number may be lost, resulting in error.

To achieve addition/subtraction arithmetic, there are five essential algorithm steps [Pat94] to follow:

- compare the exponents;
- align the mantissas;
- add or subtract the mantissas;
- normalize the result; and
- round-off the result.

The flow chart for binary floating-point addition/subtraction is shown in Figure 3-2. Steps 1 and 2 align the mantissas to provide a proper set up for step 3 for the addition/subtraction of the mantissas. After the fourth step normalization occurs, and the exponent must be tested to see whether overflow (the value of a result is too large in relation to the representable range) or underflow (the computational result is less than the small value of the range) occurs or not. Once overflow or underflow occurs, the procedure will be interrupted.

The mantissas result may not be non-representable because of the finite number of bits. Rounding and truncation are two methods for handling the precision problem [Goo89]. For simplicity, truncation can be accepted. As a result, the rounding method is a compound of three truncation conditions. The precision is increased by the rounding method [Goo89], therefore, this method is preferred in the design. With three extra bits of precision two bits are used for ensuring correct rounding and one bit is called the *sticky bit* which is the least significant bit [Bur94] [Zyn88]. These extra 3 bits are generally located in the lowest position. The further details can be referred to Zyner thesis [Zyn88] and Burgess lecture note [Bur94].

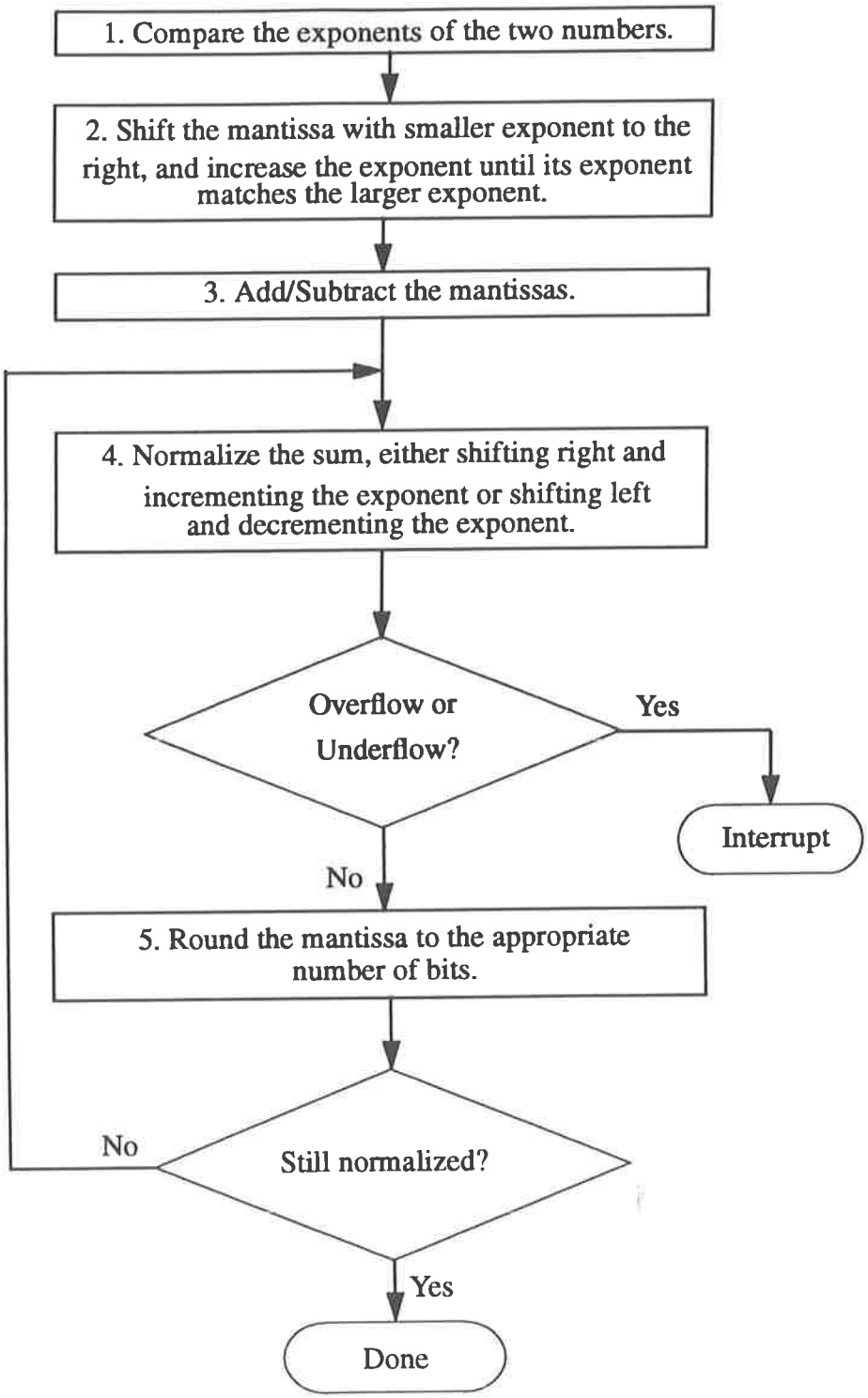


FIGURE 3-2. A floating-point addition/subtraction flow chart.

3.2.2 Mapping Into Hardware

In terms of the floating-point adder/subtractor algorithm, from circuit consideration point of view, a subtractor can perform the comparison between two 11-bit exponents. After obtaining the difference, two logic circuits would be required to check if the difference is larger than $|56|$. Once an overflow occurs, the alignment shifter sets the mantissa to 0. If there is no overflow, the 55-bit (or 56-bit) shifter aligns the exponents. A 56-bit addition can add the two mantissas. The function of normalizing the mantissa result into the standard format, will be completed by a priority detector, encoder and normalization shifter. An 11-bit incrementer and an 11-bit decrementer (or 11-bit subtractor) are used to adjust the exponent result in terms of the different normalized mantissa results. It is necessary to check for exponent overflow after incrementing the larger exponent, and for exponent underflow after subtracting from the larger exponent. The output exponent can be selected by a 3:1 MUX. Figure 3-3 is the block diagram of the process. More detailed logic circuit

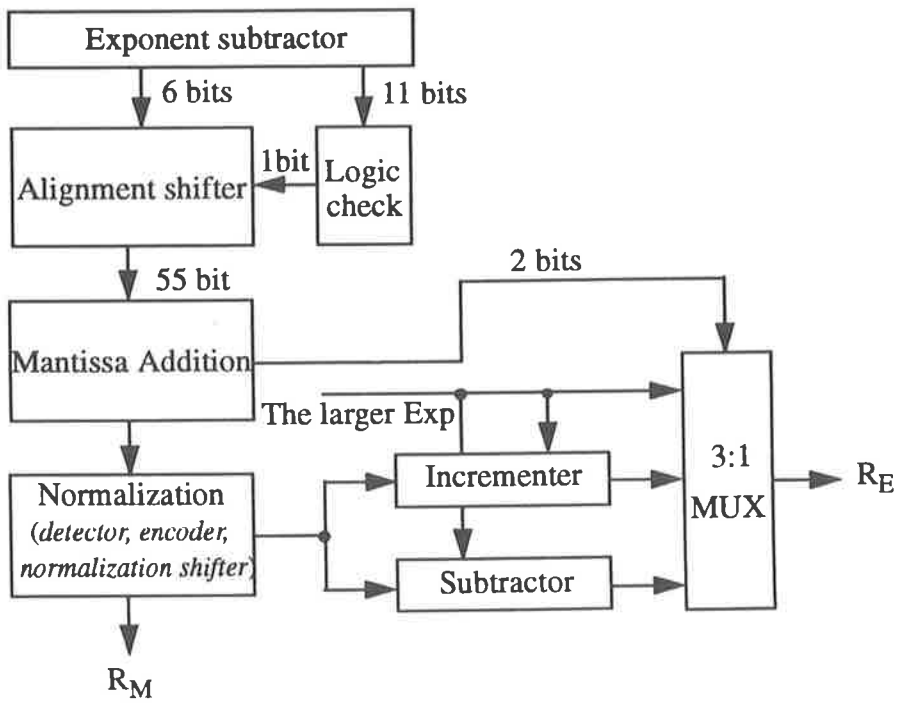


FIGURE 3-3. A block diagram of the addition/subtractor process.

designs will be presented in the next chapter.

To deal with the procedure in terms of the flow chart, we can divide the procedure into two parts: the *exponent portion* and the *mantissa portion*.

3.2.3 Exponent Portion Architecture

The algorithm of the addition/subtraction process has been presented in the previous subsection. A block diagram illustrating the exponent datapath is shown in Figure 3-4. There are several steps to perform in the exponent portion:

- *Compare the exponents*. The biased exponents are positive numbers. Hence, the difference between the two numbers is the difference between the two exponents. An 11 bit subtractor is employed to compare the two exponents.

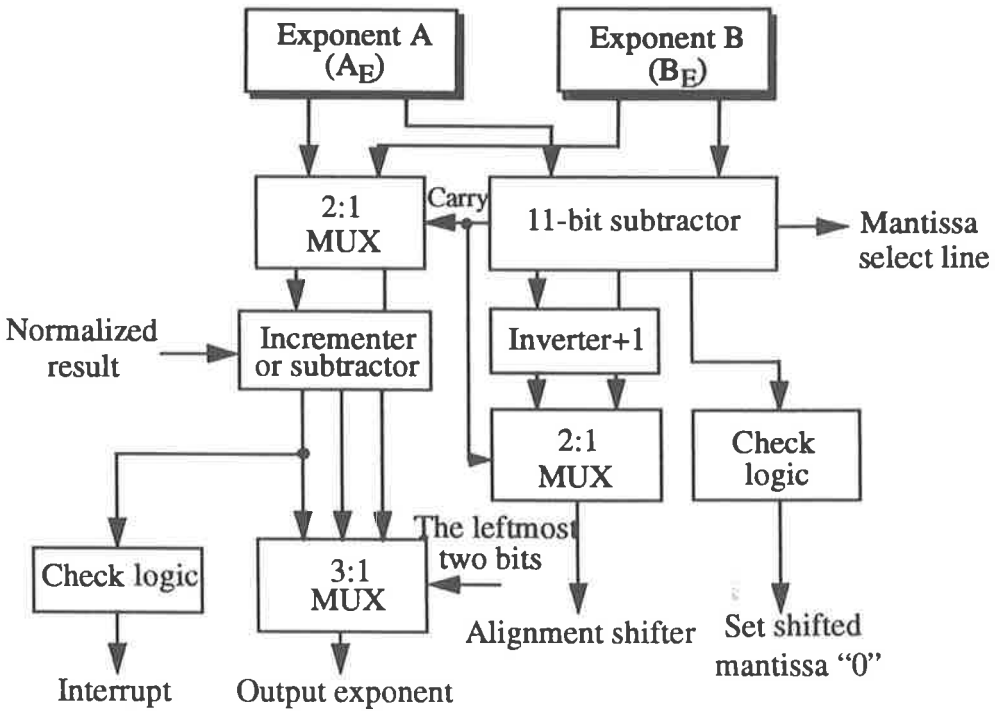


FIGURE 3-4. A block diagram of an exponent datapath.

- *Shift the mantissa of the number having the smaller exponent to right.* Depending on the exponents, assume the exponent of A is A_E and the exponent of B is B_E, if $A_E - B_E > 0$, the B mantissa will be shifted right; if $A_E - B_E < 0$, the A mantissa needs shifting right. The two cases can be distinguished by examining the carry out of the 11th bit of the subtractor.

- *Determine how many bits need shifting.* $|A_E - B_E|$ denotes how many bits the mantissa associated with the smaller exponent is shifted. If the carry out of the 11th-bit of the subtractor is 0_2, the result of the exponent difference is the number by which the B mantissa needs shifting. If the 11th-bit carry of the subtractor is 1_2, the result of the exponent difference must be converted into a signed-magnitude number by an 11-bit inverter and an incrementer. This signed-magnitude number is the number of bits by which the A mantissa must be shifted.

- *Check the maximum number bits to be shifted.* Since there are 53 mantissa bits and considering that rounding of the normalized result mantissa (precision) adds an extra 3-bits, the maximum shifted number, which is the difference between the two exponents becomes 56 bits. If the number of bits to be shifted is greater than or equal to 56, the shifted mantissa value is all 0, the mantissa result will be equal to the value that has the larger exponent. After subtracting the two 11-bit exponents, the maximum number of bits to be shifted is 56, that is 00000111000 in binary notation. Two situations in checking maximum numbers may happen. Assuming $A_E - B_E > 0$, if the 11-bit exponent result is greater than $(00000111000)_2$, then the value of the mantissa having the smaller exponent will become all 0. If $A_E - B_E < 0$, the result may be less than $(11111001000)_2$ or $(-56)_{10}$, in which case the mantissa having the smaller exponent is all 0 after shifting. In hardware implementation, once $|A_E - B_E| > 56$ is verified, the mantissa with the smaller exponent is set all to 0 immediately.

- *Increment or subtract the exponent after normalization.* Since the smaller exponent has to match the larger exponent, the larger exponent between A_E and B_E is selected as the output exponent. In fact, there are two other cases for consideration. After adding/subtracting the mantissas, the result of the mantissa must be normalized. Since the true mantissa range is from 1 to 2, after adding two mantis-

sas, the range of the mantissa of the result will be from 2 to 4, when the two mantissas have the same signs. If the result equals 4, the carry bit of the most significant bit of the mantissa result will be 1 and overflow occurs. Then the mantissa result must be shifted once to the right making the hidden bit 1, and the larger exponent must have 1 added to it. When the two mantissas have different signs, the result may be less than 1. To comply with the IEEE Standard Format, the result must be shifted from right to left until the hidden bit is 1. The worst case occurs when only the lowest bit of the result is 1, in which case, the result must be shifted by 56 bits from right to left. Meanwhile, the larger exponent will be reduced by 56. As may be seen from this, if there are n ($n < 56$) leading zero bits in the result's mantissa from left to right, we need to shift $n + 1$ bits to make the hidden bit 1. After shifting $n + 1$ bits to the left, $n + 1$ is then subtracted from the larger exponent.

- *Check underflow, overflow.* Through adjusting the exponent, during normalization, an underflow or overflow may be produced. If this occurs, then the procedure will be interrupted. For example, the occurrence of underflow could be signalled and the underflow could be translated into a zero value.

- *Choose one of the exponents as output.* Once there is no underflow, overflow or infinity, the output exponent will be the larger exponent of the two input exponents, after incrementing the exponent, or subtracting $n + 1$ bits (n is the shifted left numbers). At this point, a 3:1 selector will be adopted to determine the output exponent. Role of 3:1 MUX will be described later.

3.2.4 Mantissa Portion Architecture

The remaining part of floating-point addition/subtraction is the mantissa portion. Its architecture is illustrated in Figure 3-5. After comparing the exponents, depending on the value of the carry, we can deal with the part of mantissas as follows:

- *Select the mantissa that needs shifting.* If $A_E - B_E$ is greater than 0, A_E is greater than B_E, then B_M must be shifted. In contrast when $B_E > A_E$, A_M will be shifted right. The first stage two 2:1 multiplexers in Figure 3-4 are used to select which mantissa needs shifting, and the select line is controlled by the carry.

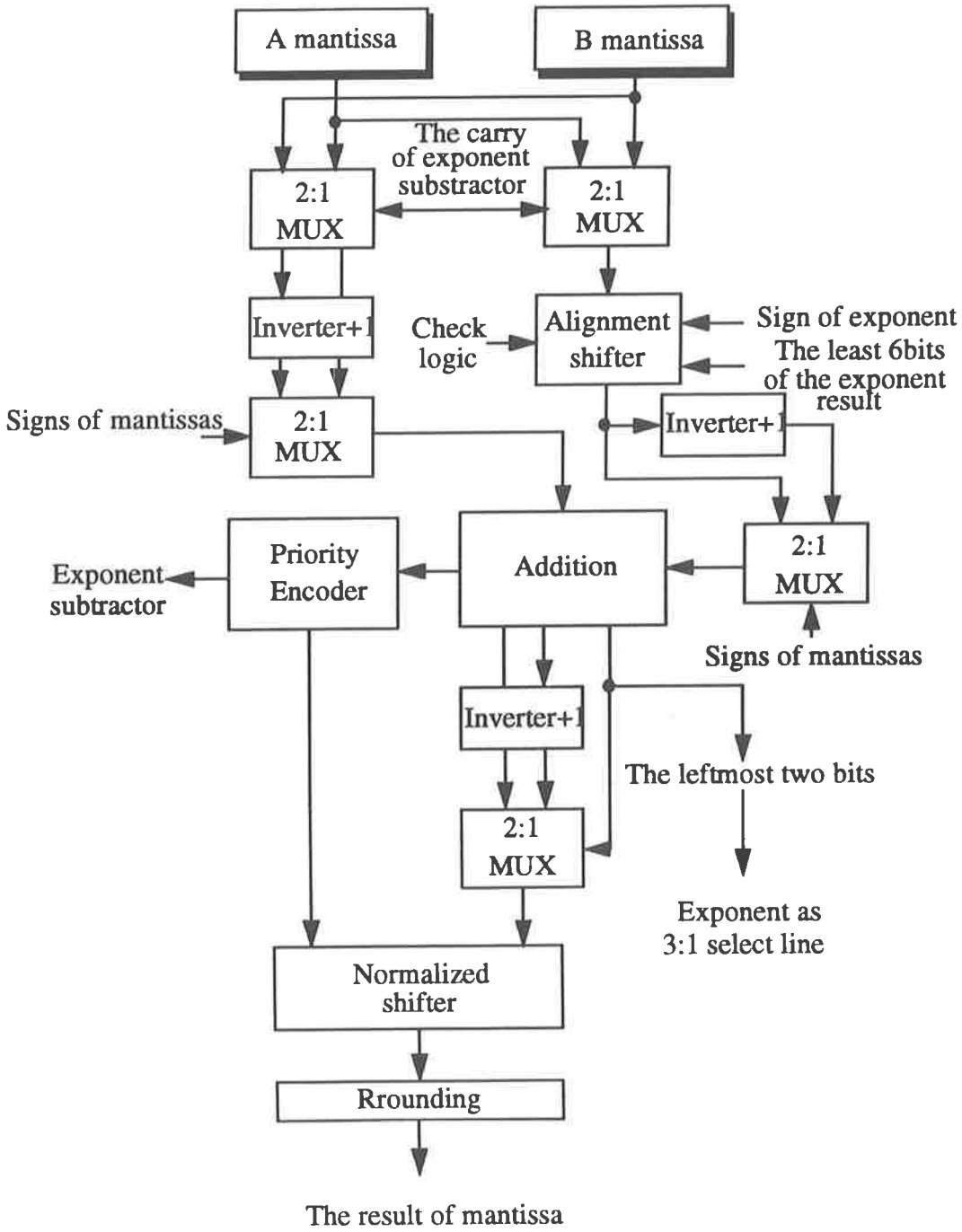


FIGURE 3-5. A block diagram of the mantissa datapath.

- *Shift the mantissa having the smaller exponent by n bits.* The difference bit between the exponents is given by the least 6 bits of the result of subtracting the two 11-bit exponents, since the maximum number of bits to shift is 56, which is less than 2^6. Knowing the exponent difference, the mantissa with the smaller exponent shifts n bits (< 56) is shifted in order to align with the mantissa having the larger exponent, using the alignment shifter. If the number of bits to be shifted is greater than or equal to $|56|$, the shifted mantissa is set all 0 in the alignment shifter.

- *Convert A and B mantissas into two's complement representation.* When adding the mantissas, there are two possibilities: one is that the signs of mantissas are the same, in which case the signed-magnitude mantissas can be added immediately; the other is that A_M's sign is opposite to B_M's. For convenience, the mantissa with a negative sign is converted into two's complement representation. The two multiplexers are able to choose the negative mantissa value.

- *Add the mantissas.* Regardless of the representations of the mantissas, the chosen mantissas are added directly with fixed-point operands. Nevertheless, to determine the result of the exponent output, a 57-bit addition including a 52-bit mantissa, three extra bits of precision in the LSB and two extra bits of judgment in the MSB. We can use the carries of the 2 most significant bits to judge the exponent output. Therefore, it is necessary to adopt a 56-bit addition.

- *Choose the result and the sign of the output mantissa.* There are two cases when comparing the signs of mantissas: (1) the two mantissas have the same sign, and the result becomes the sum of the two mantissas (in signed-magnitude form) and its sign is the same as the sign of mantissas; (2) in the other case, when the mantissas have different signs, the result has two possibilities: (a) the negative mantissa value is greater than the positive one, the result is negative expressed in two's complement form which requires converting into a signed-magitude form, and the sign is negative; (b) the positive value is greater than the negative one, then the result is the current representation of the output value, and the sign is 0. A 2:1 MUX, therefore, enables us to select the correct output mantissa.

- *Normalize the mantissa result.* After adding two mantissas, not only is the result in

signed-magnitude form, but it must also satisfy the IEEE 754 standard form. It is worth noticing that once overflow occurs, the mantissa result must be shifted to the right once, and concurrently, the larger of the two exponents is incremented. If no overflow occurs, but the hidden bit is 0, the mantissa result shifts from right until the hidden bit is 1. For these reasons, a normalized shifter (left shifter), a priority detector and an encoder are required to accomplish a shift of up to $n + 1$ bits and detect the position of the first significant bit from the left which is 1, and determine how many bits need to be subtracted from the output exponent.

- *Adjust the result of the exponent.* If the mantissa results in overflow, the result shifts right one bit and the exponent portion is incremented; if the result shifts n bits left to make the hidden bit be 1, $n + 1$ is subtracted from the exponent portion. Hence, there exist three possible results for the output exponent. To achieve the correct result, the carries of the leftmost two bits of the mantissa results are used to choose one of the results as the final exponent in 3:1MUX.

- *Round the result.* According to the IEEE 754 double-precision standard formats, the mantissa is fixed at 52-bits. Therefore, a 56-bit mantissa result must be rounded off. Much research has been reported in this aspect [Zyn88] [Bur94]. Due to time limitation, the rounding logic part is not designed, then we only leave the 52 leftmost bits as the mantissa result.

3.3 Summary

In this chapter, some number representations and essential floating-point concepts have been introduced. There are several ways to represent a number in an arithmetic operation. As described in Chapter 1, we require a double precision floating-point adder/subtractor for the Solid Modelling accelerator.

The algorithms for addition and subtractor have been described and an architecture for processing the exponent and the mantissa sections have been developed. To realize the architecture and principles, the relevant digital logic circuits are considered in the next chapter.

Chapter 4 Logic Circuit Design Methodologies

Abstract: *This chapter describes the construction of a fast* ***shifter*** *and a multi-bit* ***mantissa addition****, a rapid 11-bit exponent* ***subtractor*** *and* ***priority detector*** *and* ***encoder*** *for floating-point addition/subtraction. We describe the realization of these circuits in GaAs considering trade-offs between power dissipation, area, speed and so forth.*

4.1 Introduction

In the first two chapters, we explained the aim of the project and introduced GaAs technology. The proceeding chapter presented an addition/subtraction algorithm. In **3.2.2**, we mentioned the circuit considerations of the algorithm functions of the adder/ subtractor.

This chapter aims to deal with some high performance and optimum digital circuit design strategies to complete the exponent and mantissa architecture in hardware using GaAs technology.

4.1.1 Overview

Because of the need to align mantissas and normalize mantissa, floating-point addition and subtraction operations are very complex and slow. In order to achieve high-speed operation, it is not enough to simply use a fast GaAs technology. The difficulties of logic implementation in GaAs mean that it is necessary to develop the architecture and implementation of the adder/subtractor with the limitations of GaAs in mind.

As will be seen later, the speed of the *mantissa shifter* and *mantissa adder* are the major factors which determine the speed and the area of the addition/subtraction procedure [Bir90], therefore, a rapid 55 (56)-bit shifter and a mantissa addition should be designed for optimum speed. Moreover, a *priority detector* in normalization affects the processing speed significantly. On the other hand, the remaining circuits such as exponent comparison *subtractor*, *incrementer* and *encoder* are less critical. In this chapter, we focus on the selection of fast, optimal and appropriate circuits for each part of the adder.

4.1.2 Integer Addition

Addition is the most basic arithmetic operation in computers. All arithmetic data operations within a computer utilize the adder as the central computational element. Its computation time influences processor speed directly. Here we will describe and compare some combinational adder circuits, including *ripple carry*, *carry look-ahead*, *conditional sum*, *carry skip*, and *carry select* adders [Hwa79] [Was82]

[Hen90].

A ripple carry (RC) adder is the simplest and slowest adder. It provides the most compact and simplest layout design, but it has a long delay [Hil78] [Ras86]. A carry look-ahead (CLA) adder is the fastest and also the most complex adder. A practical problem of a carry-look ahead adder is that the layout of the adder is rather irregular and the area usage is inefficient [Nga86]. Although a carry look-ahead adder is the fastest available [Hen90], due to fan-in and fan-out limitations in GaAs, the transistor counts are high, and the connection wires are long and complicated. Thus, as the number of bits increases, the VLSI area increases, the effort for layout rises, and undesirable delays may occur [Mil86]. Usually, CLA adders are used in small sizes.

The speed of carry propagation determines the addition calculation speed. In other words, a fast addition requires rapid carry propagation. Between a simple ripple carry and a fast carry look-ahead adder, carry-skip and carry-select adders have been applied widely for a mult-bit addition. Babbage invented carry-skip adders in the 1800's. Kantabutra [Kan93$_1$] [Kan93$_2$] has reported one-level carry-skip adders and two-level carry-skip adders respectively for accelerating carry propagation in practice. A number of examples are presented, which demonstrate that the carry-skip adders are fast, occupy small layout area, and have low power dissipation in CMOS technology [Kan93$_1$]. Nevertheless, a carry-skip adder is 30%s lower than a carry-select adder for the same numbers of bits [Tya90]. In spite of this, the carry-select adder occupies an area that is almost twice as large as that of the ripple carry adder, and 35% larger than that of a carry-skip adder in CMOS environment [Tya90].

The conditional sum adder was proposed in 1960 by Sklansky [Skl60] as a new way to increase carry propagation, and Winograd verified that it is the fastest adder among all adders [Was82] except CLA adder. Due to its irregularity, and complex interconnections, it may use a considerable chip area with GaAs. Therefore, it is seldom used in practical applications [Unw93].

In the 80's, *Ling*'s adder [Lin81] [Was82] and *binary carry look-ahead* (BCLA) adder [Bre82] improved both regularity and speed. These improvements have brought significant performance gains in CMOS technology [Qua92] [Bec88]. Considering the low fan-in and fan-out requirements, an increasingly popular way to perform fast

carry propagation is to use the binary carry look-ahead adder in GaAs technology [Bus91]. The carry propagation of the BCLA adder is based on parallel computations. It is similar to the CLA adder in carry propagation, but the BCLA adder has a much improved regularity and diminished area. Of course, it is slower than the CLA adder.

Recently, more and more research is being done on using different number representations to speed up the arithmetic operations by reducing carry propagation, such as *redundant* [Hwa79] [Sri92] [Lyn92] and *residue* number representation [Lu92]. These have only been reported in fixed-point arithmetic operations in CMOS and the algorithms are relatively complicated.

Among multi-stage sub-adders, Hwang introduced a group carry generation method to connect them efficiently for accelerating addition [Kwa79]. This partition of group carry generation is the same as a carry generation in carry look-ahead adder, so that its propagation is rapid, however, its layout is irregular. Fujii and Ide provided new ideas to improve group carry generation regularity in carry select adder design individually in 1992 and 1993 respectively, but these new methods are only efficient in CMOS technology [Fuj92] [Ide93].

The fastest adder in silicon may not be the fastest in GaAs, and also it may not be feasible in GaAs implementation. We should choose a suitable adder for the word length required [Mil88].

The selection of an adder always depends on the chips specifications: area, speed and power. In general, these specifications have trade-offs in design. There are many trade-offs in adder design using GaAs technology [Car93], which are clearly different with in CMOS. The addition/subtraction of mantissas is performed exactly as with integer operands [War82]. Historically, in order to minimize the delay through the critical path -- carry-in to sum out, a combined adder consisting of a ripple carry adder and the carry select adder is popular in GaAs adder design [Ras86] [Mil88]. That is due mainly to regularity. A typical carry select adder breaks up the n bits into several sub-adders. Commonly, each sub-adder section consists of ripple carry adders [Hwa79] [Hen90] in duplicate, one of which assumes carry-in 0, while the other assumes carry-in 1. The sum is determined by the previous sub-adder carry. Obviously, the carry select adder takes almost twice the hardware in design, so reducing

the area in a carry select adder application is important [Tya90]. Moreover, a speed is the most important objective in this design, consequently, we replace the RC adders with BCLA adders for both the 11-bit subtractor and the mantissa adder.

For regularity and convenience, we employ a simplified carry generation for propagating select signals in the combined adder, although this propagation is not fast enough. It will be discussed in the mantissa adder subsection.

4.1.3 Shifter

For high speed variable shifters, the barrel shifter is an appropriate choice [Ide93] [Fuj92]. In this design, two full barrel shifters are used, one to make the exponents equal and the other to normalize the mantissa. The disadvantage of barrel shifter is that they usually occupy a large area and use a large number of transistors which results in high power dissipation. As a result, it is necessary to minimize the transistor count in design and implementation.

In the normalization subsection, the combinational logic circuits of a priority detector and an encoder will be discussed in detail. Finally, the rounding of the normalized result mantissa will be presented briefly at the end of this section.

4.2 Exponent Structure

The major hardware structures in the exponent portion are the 11-bit subtractors and the incrementer. We also describe the overflow check logic circuits and the 3:1 MUX selector logic function.

4.2.1 11-bit Subtractor

Two 11-bit subtractors can be used to simultaneously evaluate $|A_E - B_E|$ [Jii92] [Kat90] to accelerate this operation, even though it uses additional space. The result of the two operations $A_E - B_E$ and $B_E - A_E$ can be selected by the carry of $A_E - B_E$. If the carry is 0, the $A_E - B_E$ value is selected as the exponent result (R_E), if the carry is 1, $R_E = B_E - A_E$. R_E can be selected by a 2:1 MUX as shown in Figure 4-1 (a).

Figure 4-1 (b) represents a different architecture of which occupies less area of

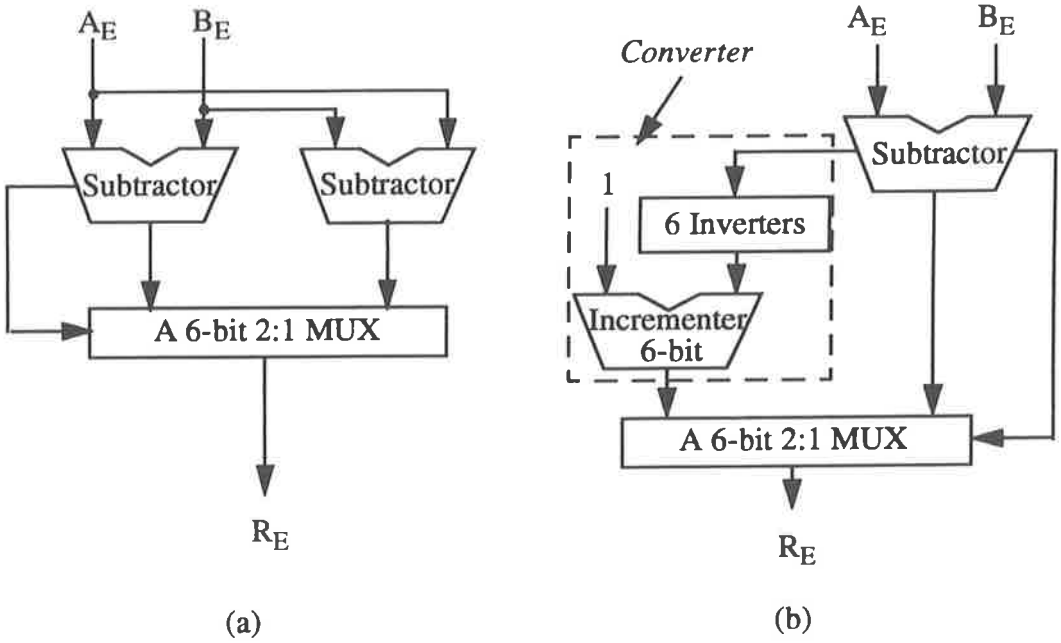


FIGURE 4-1. The block diagrams of an 11-bit subtractor.

Figure 4-1 (a). The subtractor calculates $A_E - B_E$. If $A_E > B_E$, then R_E is the real exponent difference; if $A_E < B_E$, R_E will be represented in two's complement representation. Consequently, we must convert R_E into magnitude form. Actually, the conversion has to be performed only on the lowest 6 bits of R_E as the maximum number of bits which can be shifted is (56). For this purpose, 6 inverters and a 6-bit incrementer are required. Although Figure 4-1 (b) saves area, transistor count and power dissipation, its operation is slower and its design is more complex than that in Figure 4-1 (a). In order to reduce area, Figure 4-1 (b) is used in our design.

A subtractor can be designed from an adder [Man88] [Lew83]. One of the operands is complemented, and the carry-in is set to 1. That is $A_E - B_E$, can be calculated as $A_E + \overline{B}_E + 1$. Figure 4-2 is a diagram of an n-bit subtractor.

For high speed operation, a hybrid adder design may be a better choice to adopt than a single adder. The specified examples lead to two different types of hybrid adders in Figures 4-3, and 4-4, which are a combination of a CLA adder and a RC adder, a combined adder of RC adder and CLA adder [Hen90], respectively.

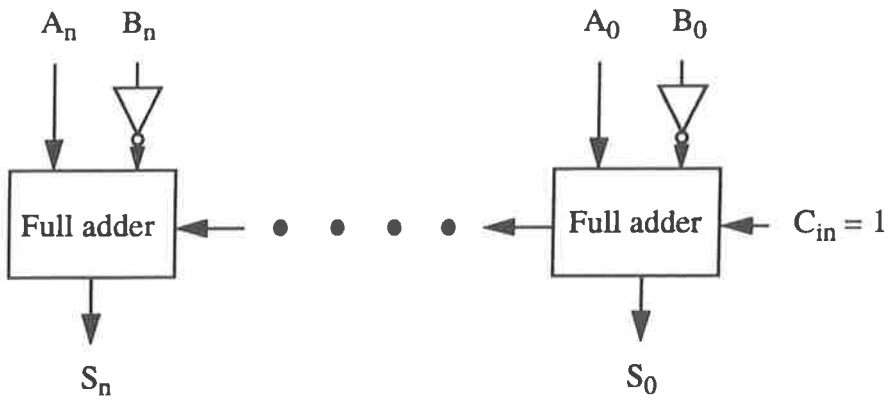


FIGURE 4-2. A n-bit subtractor constructed from an adder.

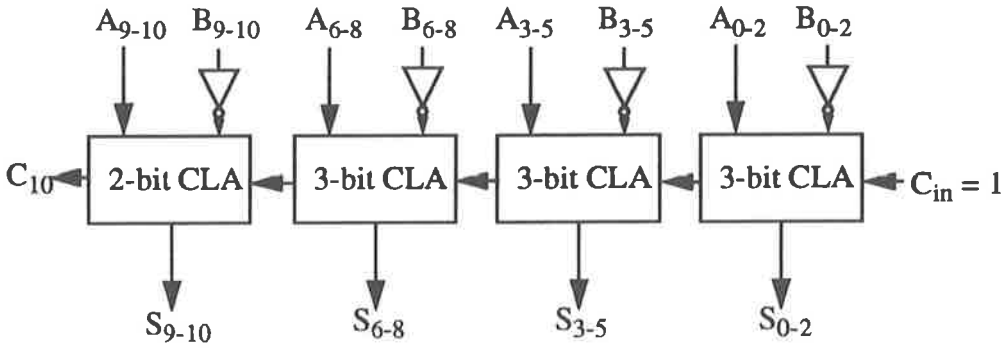


FIGURE 4-3. An 11-bit combined subtractor of CLA adder and RC adder.

Since fan-in and fan-out limitations exist in GaAs technology, the CLA adder is difficult to implement for more than 3 bits. Because of that, a 3-bit CLA adder is regarded as the largest feasible CLA in this process sub-adder. However, the carry propagation among these sub-adders uses ripple carry propagation connections, so that the carry propagation takes a long time. Figure 4-4 represents another hybrid adder, which is too complex to implement in layout easily. On the other hand, due to interconnection complexity, its speed may not be fast enough [Hen90].

Not only for speed, but also for placement and regularity, a new hybrid adder

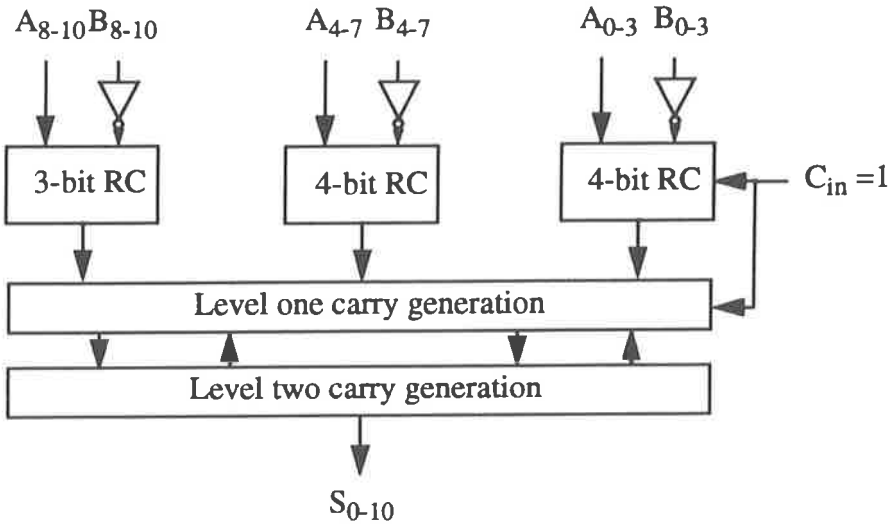


FIGURE 4-4. An 11-bit combined subtractor of RC adder and CLA adder.

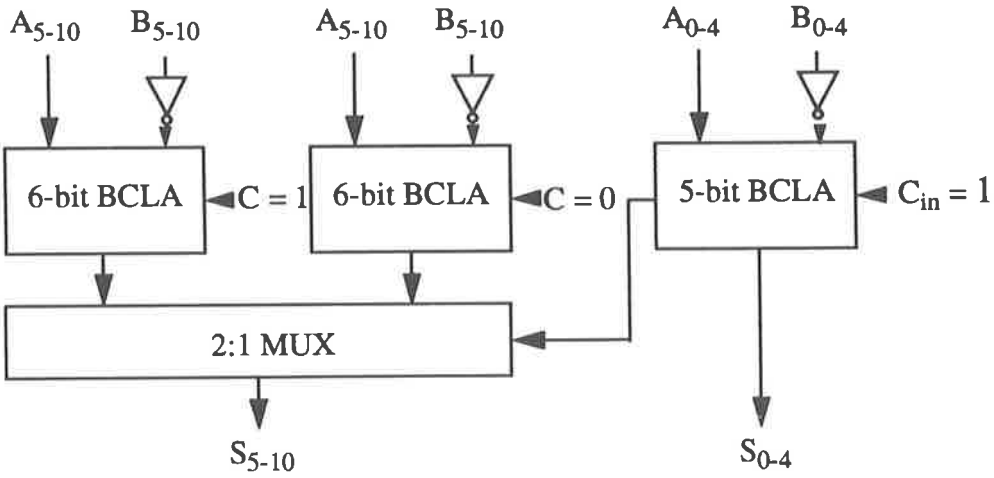


FIGURE 4-5. An 11-bit combined subtractor of BCLA adder and carry select adder.

consisting of BCLA adders and a carry select adder is developed for a long length addition. For convenience, we give an 11-bit subtractor combined by BCLA and carry

select adder as an example. We divide the 11-bit subtractor into two segments as shown in Figure 4-5. The 6 most significant bits are simultaneously subtracted in two separated BCLA adders, one with $C = 0$, and the other with $C = 1$. At the same time, the 5 least significant bits are subtracted and provide the carry to select one of the two 6 MSB BCLA adders' sums as output using a 2:1 MUX. The operational result takes approximately a 6-bit BCLA adder delay plus a 2:1 MUX delay.

A complete structure of the 6-bit BCLA adder is shown in Figure 4-6. A struc-

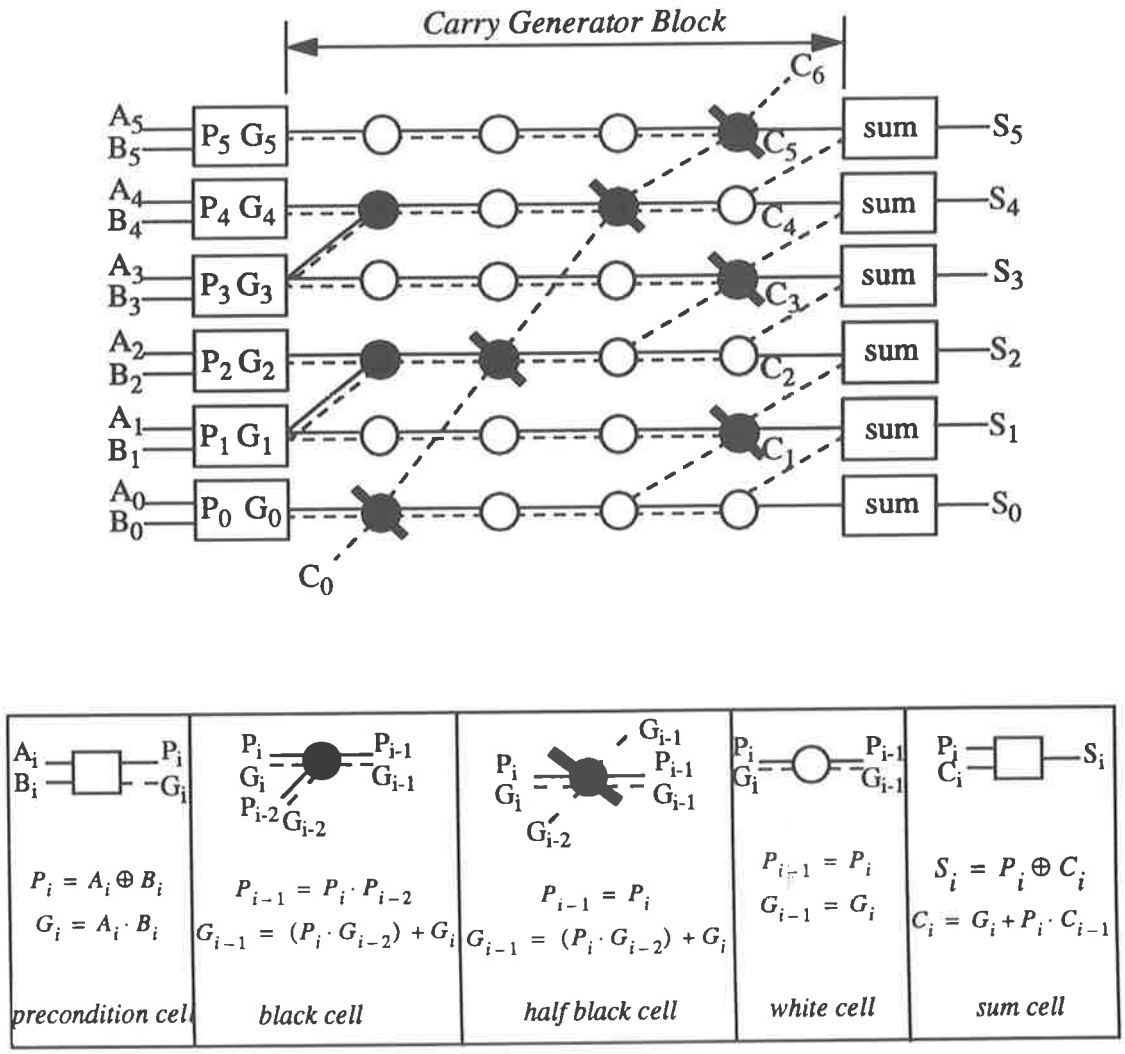


FIGURE 4-6. A block diagram of a 6-bit BCLA adder structure.

ture of the 5-bit BCLA is analogous to that in Figure 4-6, using 5 rather than 6 bits. A 6-bit BCLA adder only needs 4 carry generator blocks. It can be seen easily that as the odd bits use less area than the even bits, the free area can be exploited for placing the 2:1 MUX. The details will be explained in the next chapter.

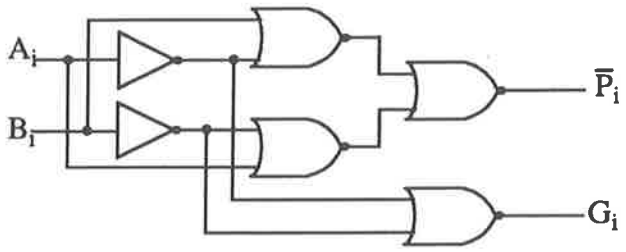
Assuming that the inputs are $A_i, B_i (i = 0...5)$ and C_0, the outputs of the pre-condition cells are $P_i = A_i \oplus B_i$ and $G_i = A_i \wedge B_i$, and the outputs of the sum cell are $C_i = G_i + P_i \cdot C_{i-1}$ and $S_i = P_i \oplus C_i$. The white cell, the black cell, and the half black cell are carry generator cells whose expressions are presented in Figure 4-6. Where $P_i, P_{i-1}, P_{i-2}, G_i, G_{i-1}$ and G_{i-2} are the propagate signals from the precondition cells to generate the sum output, also P_{i-1} and G_{i-1} are the current output through carry generator cells; and the P_{i-2} and G_{i-2} are from the previous stage.

The logic for the above cells can be implemented with inverters and NOR gates as shown in Figure 4-7, so all cells can be readily implemented in GaAs DCFL as described in the next chapter.

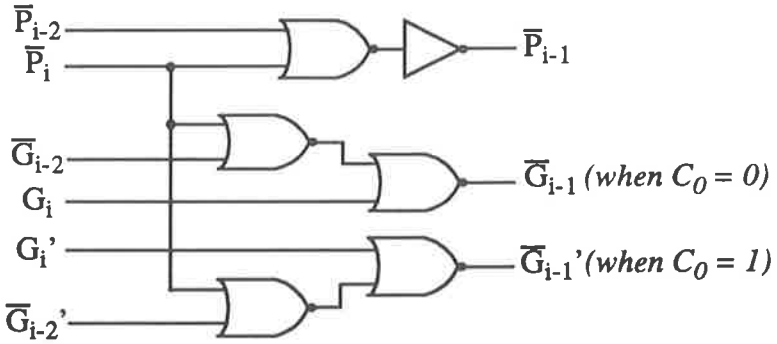
The BCLA adder for the top 6 bits is duplicated to deal with the two cases in which the carry-in is 0 and 1, in the carry select adder shown in Figure 4-5. The pre-condition cell in the two cases can be shared, and part of the black cell can be shared, but the rest of the cells must be separated for each adder. This is because when the carry is 0, the G_i is different from when the carry is 1. In other words, the half black and the sum cell must have two separate parts. The two black cells can share the part $\overline{P}_i + \overline{P}_{i-2}$ so that an inverter and a NOR gate are saved.

4.2.2 Overflow Check-Logic Circuits

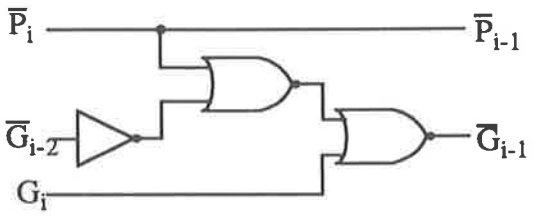
After subtracting the two exponents, R_E must be checked to see whether it is greater than 56 or not. Two related check-logic circuits are shown in Figure 4-8, where E_{rj} is the jth bit of the 11-bit subtractor result $(j = 0...10)$, C_{e10} is the carry-out of the 11-bit substractor. Figure 4-8 (a) is to check whether the $A_E - B_E$ result is greater than 56. If one of the five most significant bits is 1 or if bits 5, 4, 3 are all 1, then there is an overflow so that F_1 is high. Figure 4-8 (b) is to detect whether the $A_E - B_E$ result is less than -56. The F in Figure 4-8 (c) is a symbol to judge the



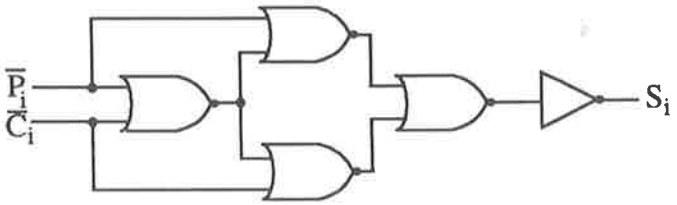
Precondition cell (shared part)



Black cells (shared some part)

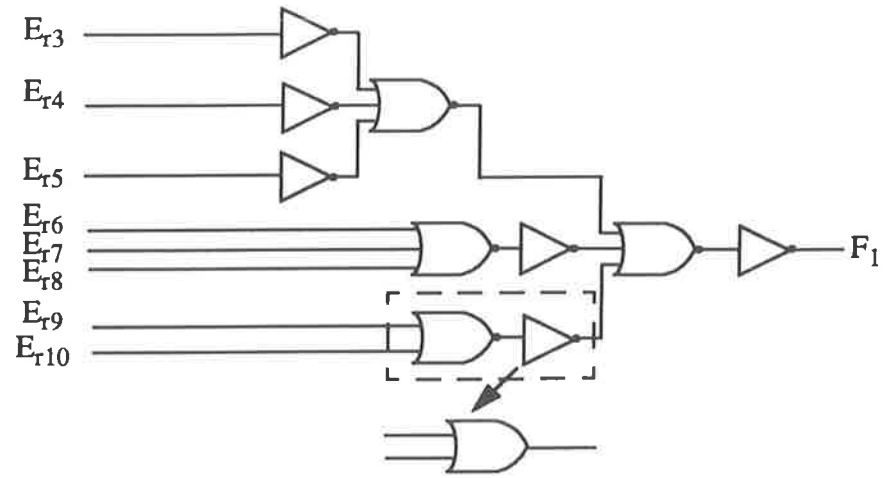


Half black cell (needed two parts)

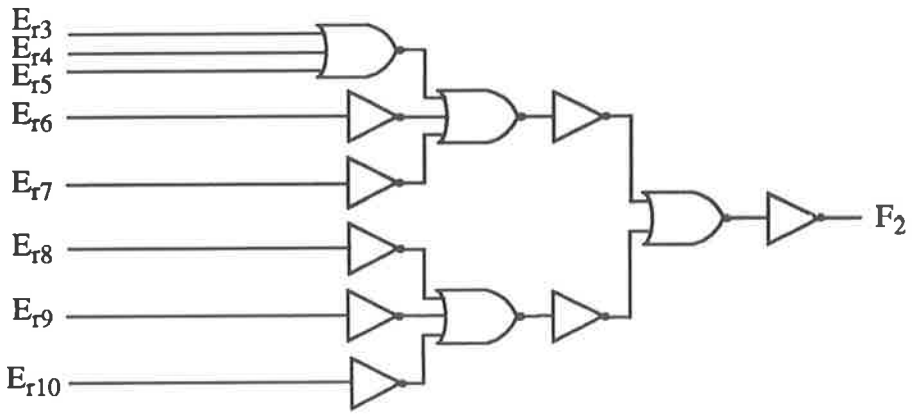


Sum cell (needed two parts)

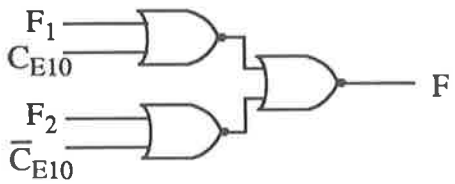
FIGURE 4-7. The logic circuit schematics of the cells in BCLA adder.



(a)



(b)



(c)

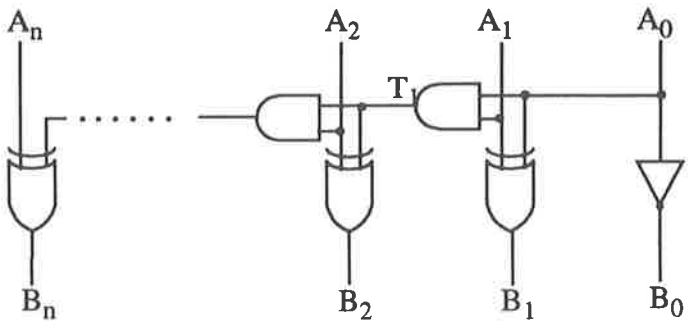
FIGURE 4-8. Check logic circuits in GaAs environment.

occurrence of an overflow. If $F = 1$, that means $|A_E - B_E|$ is greater than |56|, then the mantissa having the smaller exponent will all be set to 0 so that the result of adding the mantissas is equal to the mantissa value with the larger exponent. This function will be performed in the shifter subsection. For saving the area relatively to achieve the same function, if a NOR driving only one inverter in DCFL can be realized by a SDCFL OR gate shown in Figure 4-8 (a).

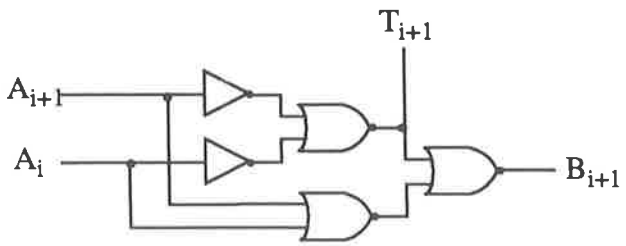
4.2.3 Incrementer

Referring to Figure 4-1 (b), a 6-bit incrementer is required to find the two's complement of R_E when $A_E - B_E < 0$. We also require an 11-bit incrementer for normalization.

To produce an incrementer, the adder circuit may be simplified into an EXCLUSIVE OR to obtain the result of the incrementer and AND gates to propagate the carry [Lew83], as shown in Figure 4-9 (a), in which an iterative structure is applied to n bits



(a)



(b)

FIGURE 4-9. N-bit incrementer logic diagram.

of arbitrary length. Figure 4-9 (b) shows the logic of the XOR and the AND gate. When adding 1 to the rightmost bit, the LSB of a number will change to the opposite state. If LSB = 1, the bit is changed to 0 and there is a carry; if LSB = 0, it is converted into 1.

Note that because of the linked chain of AND gates, the value of the nth bit is available after $(n-1)$ AND gates plus one XOR delay. For large n, this may be unacceptable. A faster carry circuit can be used at the expense of circuit complexity in this case. However, in some situations, the incrementing operation has to wait for another operation, in which case the incrementer in Figure 4-9 is sufficient. This is the case for

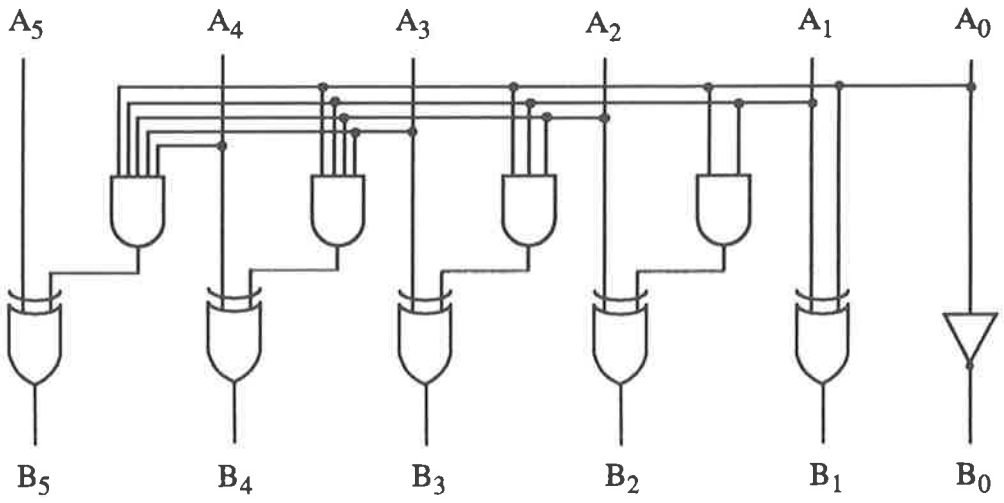
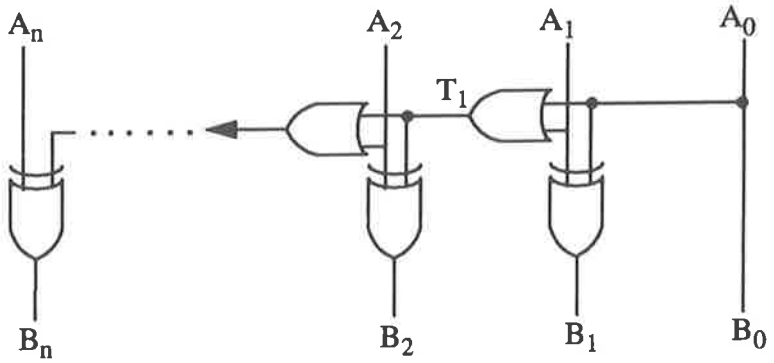


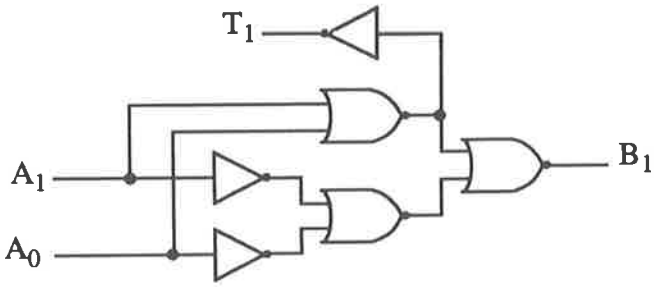
FIGURE 4-10. A 6-bit incrementer with carry propagation form of CLA adder.

the 11 bit incrementer because the operation time should match the speed of the adjusted exponent 11-bit subtractor. There are some advantages in accelerating the 6-bit incrementer by using the carry propagation method of a carry look-ahead adder. The logic diagram is shown in Figure 4-10. We can see that the 5th bit AND gate has five fan-ins and each bit slice is irregular, so that it is suitable for only a narrow incrementer.

When adding two mantissas, if the signs are different, by convention, the mantissa with the minus sign is converted into two's complement form and then added to the other mantissa. This conversion requires 52 inverters and a 52-bit incrementer. Alternatively, a 52-bit two 's complementer shown in Figure 4-11 can be used. This



(a)



(b)

FIGURE 4-11. An N-bit two's complementary logic diagram.

two's complementer takes approximately 52 OR gates' delay plus one XOR delay, thus it may take a lot of time. The operation, however, is not critical. This is because it must wait for the result of the exponent difference calculation. In fact, an efficient way to perform this operation simply and save a number of transistors will be shown in detail in the mantissa subsection.

4.2.4 A 3:1 Selector And Overflow Detector

As discussed earlier, there are three possible cases for the exponent output, depending upon the mantissa result. After normalizing the mantissa, the final exponent output may be the result of: (a) incrementing the larger exponent (there is an overflow in mantissa result); or (b) subtracting some number from the larger exponent; or (c) being the larger one of the two input exponents. We can increment and subtract some numbers from the larger exponent simultaneously. Consequently, an 11-bit 3:1 MUX will be employed to select the final exponent output (see Figure 3-3). The 3:1 selector output is chosen by the two select signals, which must be generated from the mantissa adder.

Assuming that the two select lines are X and Y respectively, which are the 55th and 56th bits of the mantissa result, and Z is as the output exponent. Table 4-1 defines the requirements of the 3:1 selector in a truth table form, where Sub is to express that $n + 1$ (n is the number of bits to be shifted from the right to the left) is subtracted from the larger exponent, NC is no change, x can not occur, and Inc expresses that the larger exponent needs incrementing.

Table 4-1. The truth table of 3:1 selector.

X	Y	Z
0	0	Sub
0	1	NC
1	0	Inc
1	1	x

Figure 4-11, based on the above truth table, is a one bit slice of the selector in NOR gates. Eleven similar cells are used.

When incrementing or subtracting, overflow or underflow in the exponent may occur. If the exponent result is negative, that is the carry-out of the 11th bit is 1 or if the result is 0, an underflow exists. Alternatively, if the carry-out of the 11th bit of the incrementer is 1 or if the result is equal to 2047, an overflow occurs. Regardless of overflow or underflow, in both cases, a flag will be set to 1. Figure 4-13 shows this logic function.

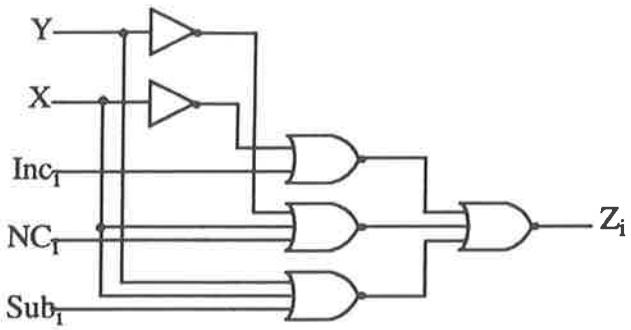


FIGURE 4-12. A cell logic diagram of 3:1 selector.

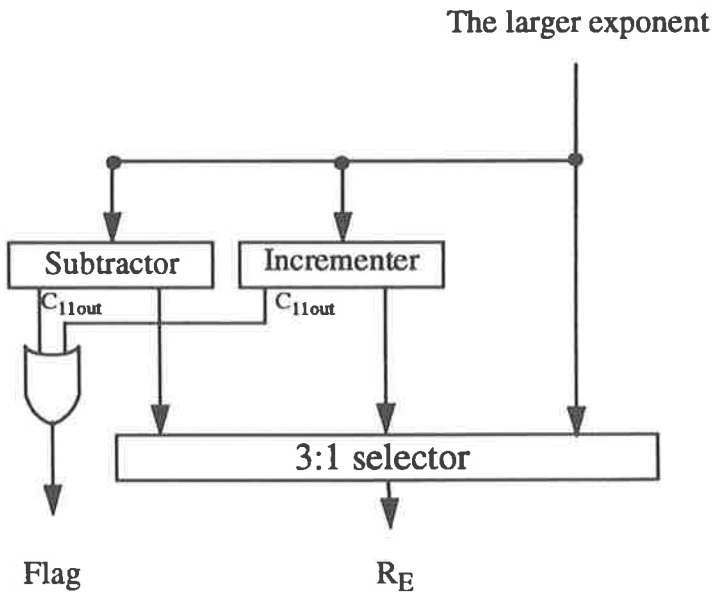


FIGURE 4-13. The diagram of exponent overflow and underflow detection.

4.3 Mantissa Structure

As will become clear later, the speed of a floating point addition/subtraction is determined mainly by the shifter and the mantissa addition operations. A barrel shifter is

employed in this design for maximum speed. A 56-bit mantissa addition has been considered in detail not only for speed but also for regularity, placement and area. Normalization functions will be discussed in two parts: (1) priority detector; and (2) encoder. Rounding will be discussed at last.

4.3.1 A Multi-bit Shifter

For high speed operation, normally a leading alternative is a barrel shifter, which is fast but uses a lot of area. To reduce the area, a tree shifter has been invented and applied in some technologies [McG90], however, this approach can not be employed in GaAs, as it needs a large area.

Since the maximum number of bits to be shifted during alignment is 56, the least 6 bits of the exponent difference are used as the shifter control lines. There are 6 stages which shift 1 bit, 2 bits, 4 bits, 8 bits, 16 bits, and 32 bits, respectively. That is the first stage shifts 1 bit, the second stage shifts two bits and so forth. For example, if the shifting mantissa is all 1, and the exponent difference is $54 = (110110)_2$, the mantissa with the smaller exponent needs shifting right by 54 bits. The control lines of the second, the third, the fifth and the sixth stage will all be high. Figure 4-14 is a block diagram of 55-bit barrel shifter.

The six C_i bits are the difference between the two exponents. If $C_i = 0$, the stage k ($k = 0 \ldots 5$) does not shift its input; otherwise it shifts its input by 2^k bits. To shift the input by N bits, the barrel shifter uses the kth bit of the binary representation of N to control the kth shift unit. Figure 4-15 illustrates an 8-bit block shifter diagram. The maximum number of bits which can be shifted in this example is 7. There are three stages, each consisting of 8 MUXs. The logic diagram of a MUX is shown in Figure 4-16. When the left input of a MUX is 0, the MUX can be simplified to save several gates. However, because of regularity of the layout, there is no area saving and the power saving is small, so that we chose not to make this optimization, to improve regularity. In practice, we usually design only one level 8 MUX and duplicate it to perform the function of the Figure 4-15.

Assume that S_0, S_1 and S_2 are the select lines and A_0 - A_7 are the data. The first stage is a 1-bit shifter and its right input of the leftmost MUX is A_7 and the left input

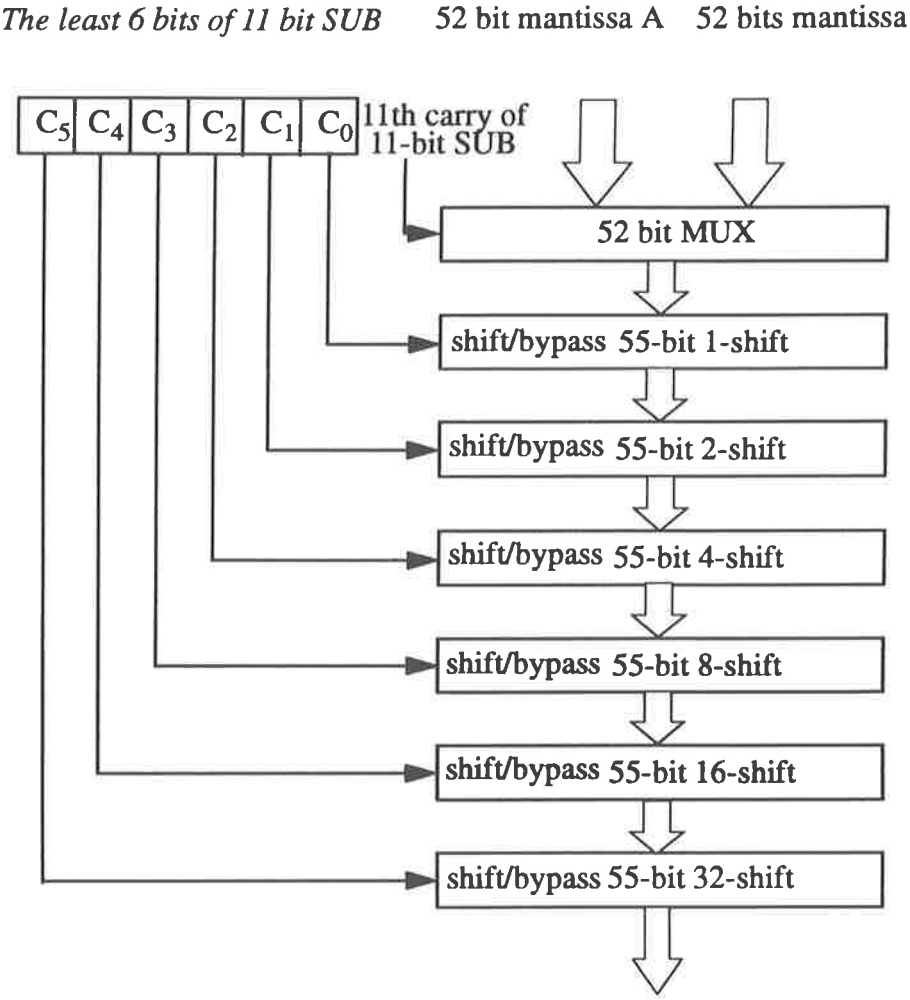


FIGURE 4-14. The block diagram of a 55-bit barrel shifter and mantissa selection

connects to ground. When the select line S_0 is low, the leftmost MUX selects the right input signal as output, that is C_7 equals A_7. Consequently, $C_i = A_i$ $(i = 0...7)$. If the select line S_0 is high, the left input (V_{DD}) of the MUX is selected as the leftmost MUX output, that is $C_7 = 1$. At the same time, $C_6 = A_7$, $C_5 = A_6$, *etc.* The data shifts one bit from the left to the right. After shifting, A_0 will be lost.

The second stage shifter is a 2-bit shifter used to shift the data C_7 - C_0. From

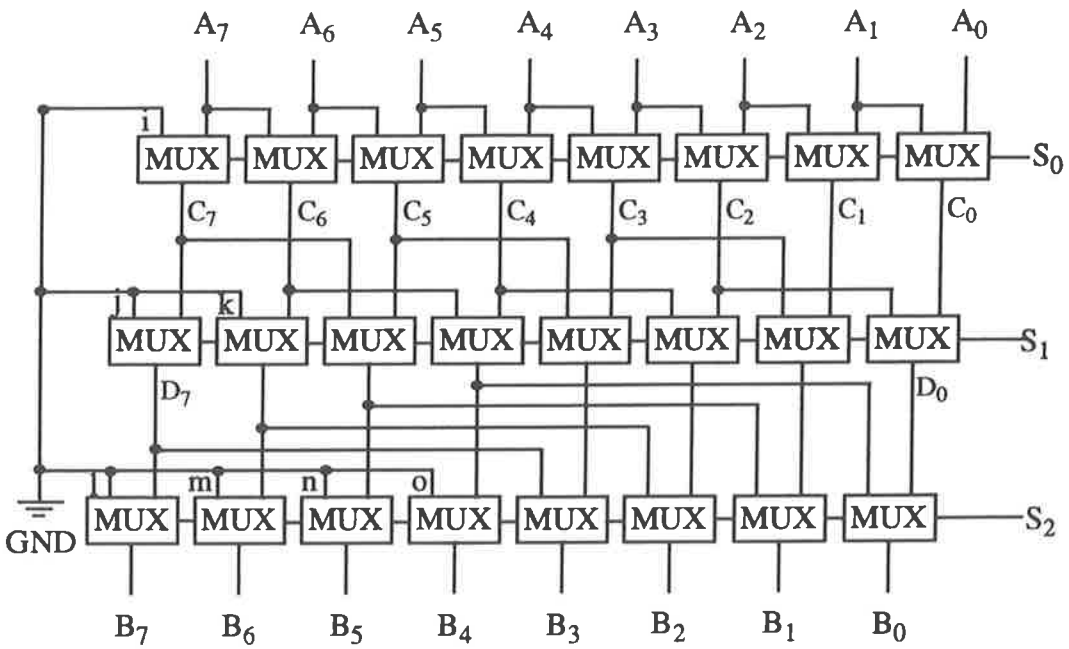
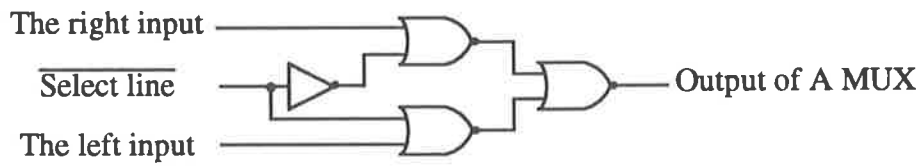
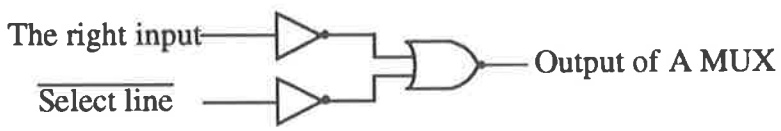


FIGURE 4-15. A schematic diagram of a barrel shifter.





(When the left input = 0)

FIGURE 4-16. A logic diagram of a MUX.

Figure 4-15, the two leftmost MUXs are connected to GND. The left input of the third MUX of the second stage connects the first MUX output C_7 of the first stage. The rest can be deduced similarly. All of the right MUX inputs of the second stage are linked to the first stage in the same order. If $S_1 = 0$, the second stage output D_7 - D_0 is exactly equal to the first stage output C_7 - C_0. When $S_1 = 1$, the second stage output D_7 - D_0 is based on shifting right by two bits of the C_7 - C_0 to make the two leftmost bits D_7 and D_6 equal to 0. The relation of D_i and C_i satisfies $D_i = C_{i-2}$. Subsequently, C_1 and C_0 will be lost.

In the third stage, the four leftmost bits connect to GND. If $S_2 = 1$, the data D_7 - D_0 will be shifted 4 bits concurrently from the left to the right. Meanwhile, the four lowest bits of D_7 - D_0 will be missed. Obviously, when shifting the data, the precision of the data will decrease gradually.

For example, if the input data A_7 - A_0 is 11111111, and S_0 S_1 S_2 are all high, then the first, second and third stages will shift 1 bit, 2 bits and 4 bits respectively. The final shifting result of B_7 - B_0 is then 00000001.

In general, an n bit shifter requires k stages, where k is the smallest integer greater than $\log_2 n$. In our shifter design, the maximum number of bits to be shifted is $n = 55 \Rightarrow k = 6$, therefore, there are 6 select lines to control the 6 stages. These 6 select lines are to be connected to the lowest 6 bits of the 11-bit exponent subtractor. Once the value of the lowest 6 bits is greater than $|56|$, the shifted mantissa will shift all 55 bits, leaving 0. A modified MUX is located in the sixth stage of the barrel shifter to set the mantissa having the smaller exponent to 0 when there is a shifting overflow. Figure 4-17 is a diagram of a modified MUX of the last stage of the 55-bit barrel shifter. The signal F is formed from a combination of F_1 and F_2 (see Figure 4-9).

The significant consideration of this shifter design is due to the hidden bit. When shifting one of the mantissas, the hidden bit must be shifted with its value from the left to the right. From this point of view, it makes the design more difficult. There are two types of schemes to address this problem: 1. readily extend one more bit $(55-bit \Rightarrow 56-bit)$ in the leftmost of every level and set the first level input of the 56th bit to 1; 2. design glue-logic circuit for the function shown in Figure 4-18. Even

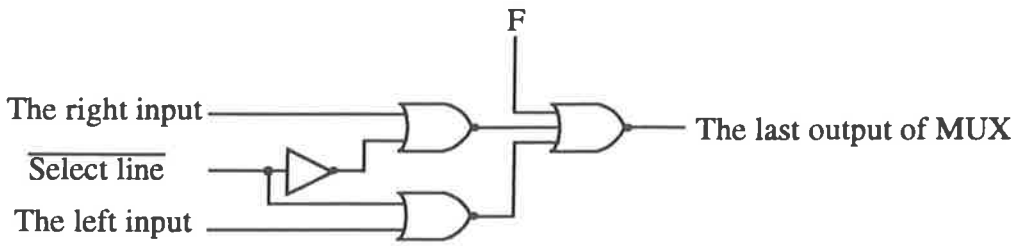


FIGURE 4-17. A modified MUX diagram.

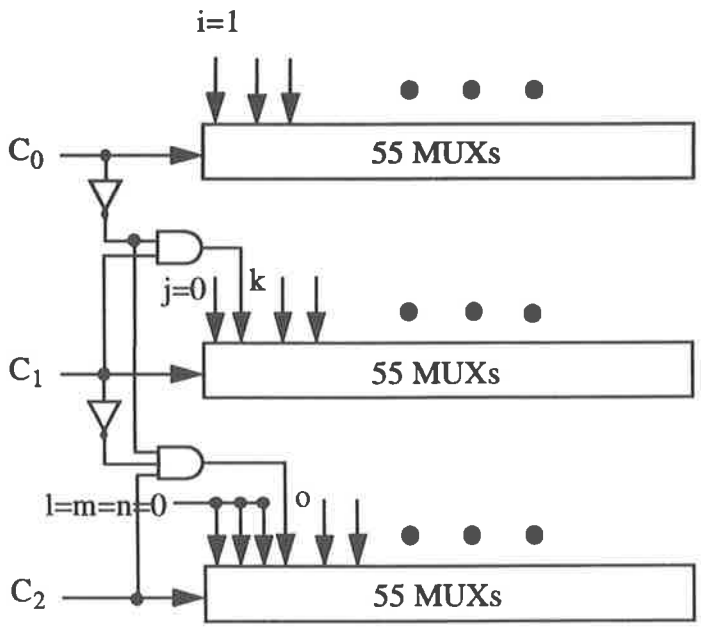


FIGURE 4-18. A modified block of FIGURE 4-15.

though scheme (1) uses slightly more gates than (2), it is faster and much more regular, and especially easy for implementation in GaAs.

Notice that for a 55-bit shifter, the least significant 3 bits are set to 000, the

remaining 52 bits are set to mantissa value in the first stage; however, for a 56-bit shifter, the 3 least significant bits not only are set to 000, but the most significant bit is also set to 1 in the first stage.

Above we have explained entirely the multi-bit barrel shifter. If using the 56-bit shifter in our design, this shifter requires 56×6 MUXs resulting in a large amount of power dissipation and area. In a CMOS environment, implementing the shifter can use n-channel pass transistor gates, so that the power and the area can be reduced significantly. However, this is not possible in GaAs technology. The reason will be explained in the next chapter.

4.3.2 A 56-bit Mantissa Adder

After aligning the mantissas, the two mantissas can be added together using a 56-bit addition. The inputs of the 56-bit addition are assumed to be data A and data B when A has a larger exponent than B. When adding A and B, the 3 least significant bits of A are set to 000, and the 56th bits of the two operands have two possibilities: if both signs (A_s and B_s) are the same, because of the hidden bits, the 56th bits are '11'; otherwise the 56th inputs are '01' (or 10), because when the signs are different, one of the mantissas requires conversion. At the same time, the hidden bit converts too. Hence, we can obtain the two logic expressions: $A_{56} = \overline{A}_s + B_s$ and $B_{56} = A_s + \overline{B}_s$ as the 56th inputs.

The two arbitrary signed and arbitrary magnitude mantissas, can be treated as an unsigned binary integer [Hen90]. If the sign of the mantissa is positive, the mantissa field directly represents its magnitude, and if it is negative, it must be converted to two's complement to simplify the addition of the two mantissas.

In general, the addition and subtraction operations can be combined into one circuit with one common binary adder and one XOR gate. Figure 4-19 shows a 4-bit adder-subtractor circuit diagram for the full adder operation. If A_s and B_s are the same, $F = 0$, and also $T_i = A_i$. The circuit operates as $A + B$, and the sign follows A_s or B_s. When the A_s and the B_s are different, then $F = 1$, $C_0 = 1$ and $T_i = \overline{A}_i$. The operation becomes B plus the 1's complement of A plus 1, that is B plus the 2's complement of A that is $= B_i - A_i$. The result $S_{0\text{-}3}$ is considered in two cases:

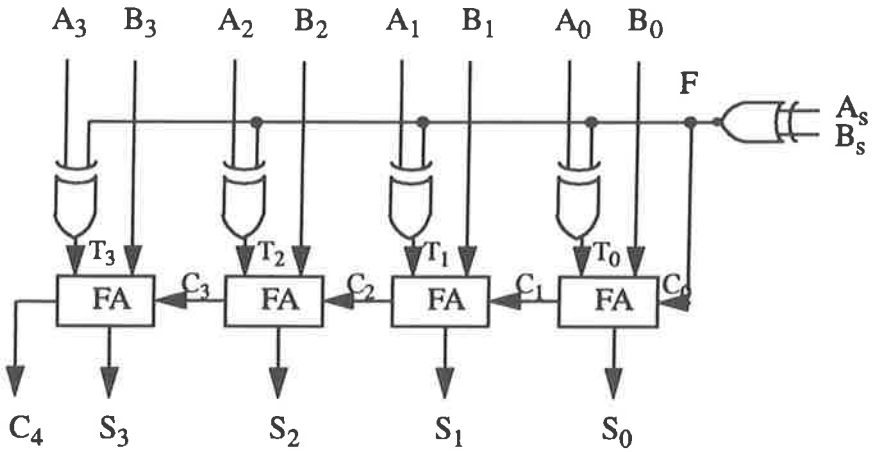


FIGURE 4-19. An adder-subtractor circuit.

$|A| > |B|$ or $|A| < |B|$. If $|A| < |B|$, $C_4 = 1$, and the result is positive. If $|A| > |B|$, there is no overflow ($C_4 = 0$), and the sum of $B - A$ must be converted into the signed-magnitude. The S_s depends not only on A_s and B_s, but also on C_4. We assume that "0" represents A, B or S is positive and "1" represents negative in A_s, B_s and S_s columns; and "0" represents there is no borrowed bit and "1" represents there is a borrowed bit in C_4 column. The S_s logic function is described in Table 4-2. Additionally, in some cases, overflow may occur when A and B are with different signs.

Table 4-2. The truth table of the signs.

A_s	B_s	C_4	S_s	Overflow
0	0	0	0	no
0	0	1	0	no
0	1	0	0	yes
0	1	1	1	yes
1	0	0	1	yes
1	0	1	0	yes
1	1	0	1	no
1	1	1	1	no

According to the truth table of Table 4-2, the sign of the result is $S_S = \overline{A}_S \cdot B_S \cdot C_4 + A_S \cdot \overline{B}_S \cdot \overline{C}_4 + A_S \cdot B_S \cdot \overline{C}_4 + A_S \cdot B_S \cdot C_4 = B_S \cdot C_4 + A_S \cdot \overline{C}_4$. The relevant logic circuit is shown in Figure 4-20.

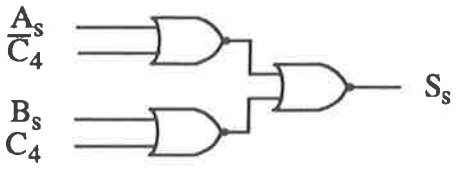


FIGURE 4-20. The logic circuit of the result sign.

Figure 4-21 is the block diagram of the mantissa adder. Figure 4-22 is another

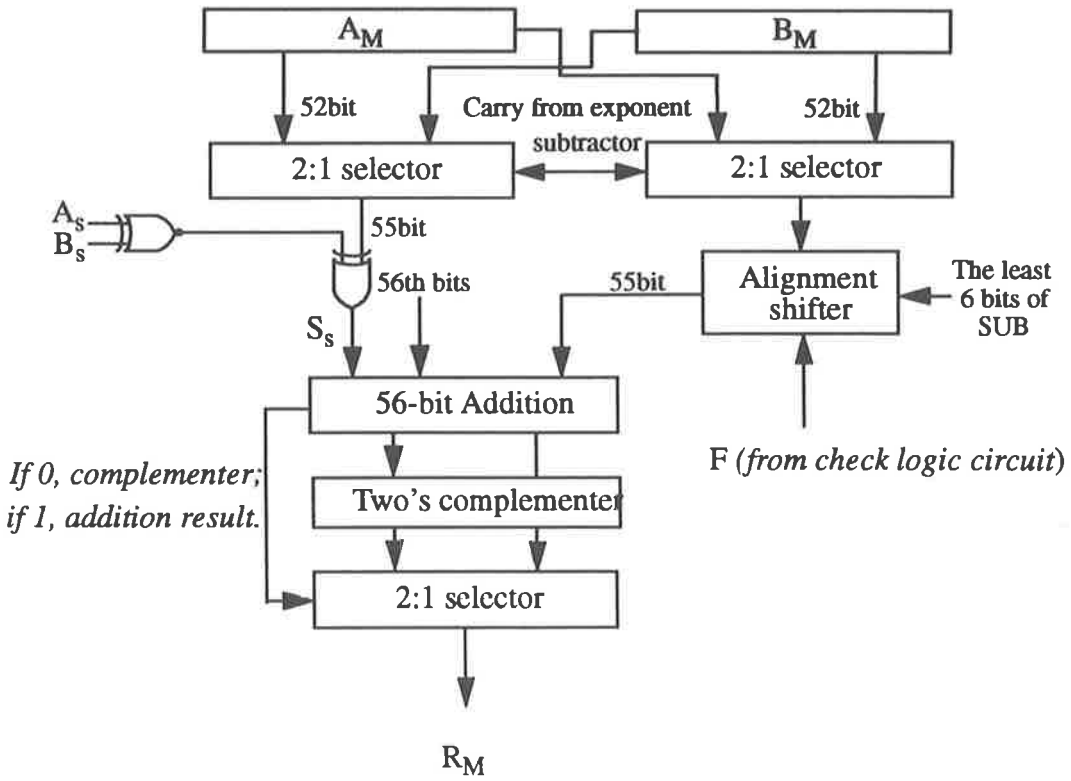


FIGURE 4-21. The mantissa addition operation diagram.

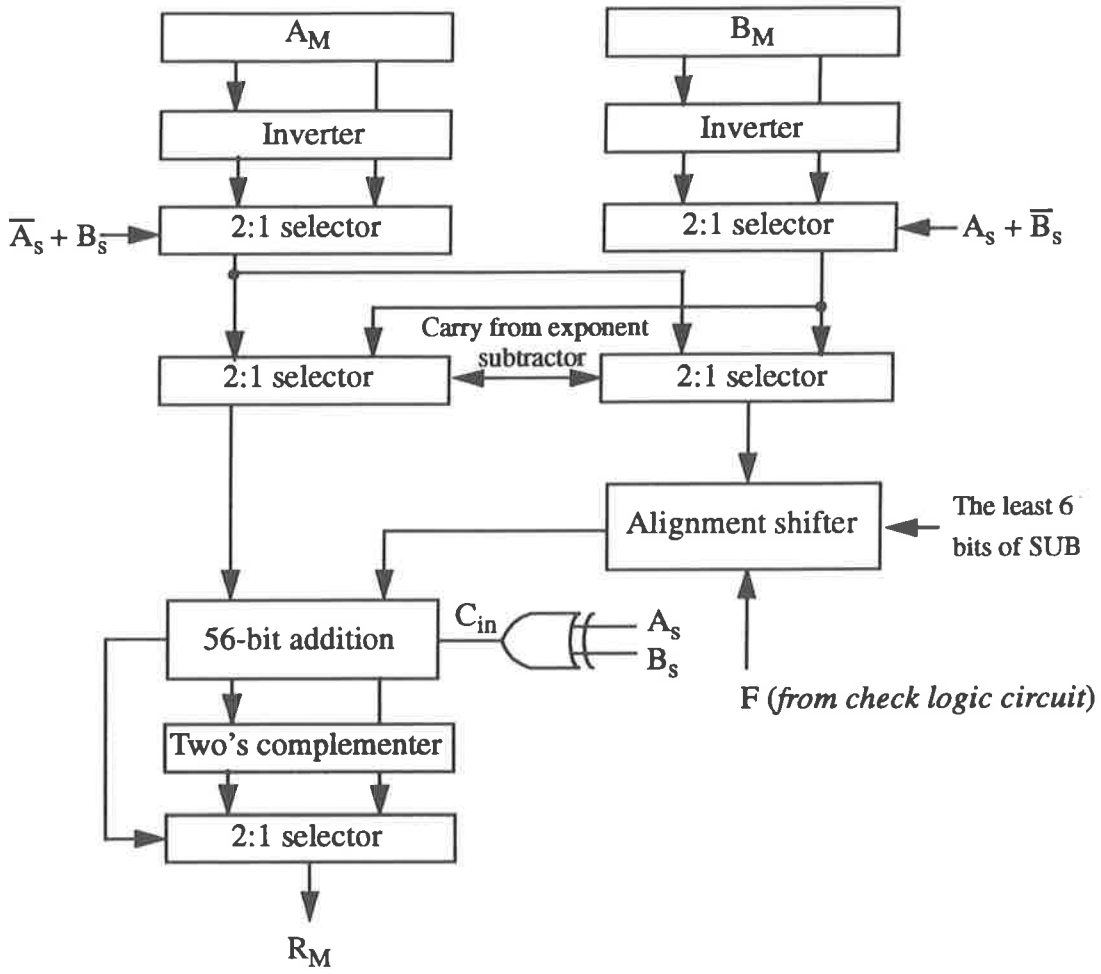


FIGURE 4-22. Another block diagram for the mantissa addition.

special example to perform the addition of signed-magnitude mantissas. Firstly, the two mantissas are complemented. Secondly, if the signs of the mantissas are the same, the mantissas can be added directly in the signed-magnitude form and the S_s is the same as that of the two mantissas' signs. If the signs are different, the mantissa with a minus sign is complemented and 1 is added. For accelerating the conversion, we can invert one of them first, and then let the carry-in of the 56-bit addition be 1 to attain the logic function of Figure 4-10.

The sign of the result is determined by the carry-out of the 55th bit. If the carry of the 55th bit is high, the result is positive; if it is low, the result is negative and

should be converted from the two's complement form to a signed-magnitude form. Both circuits in Figure 4-21 and Figure 4-22 have their own pros and cons. The circuit in Figure 4-22 is slightly faster and less area than th circuit in Figure 4-21. In terms of the placement requirements, either can be adopted. Figure 4-22 is employed for our design.

The mantissa adder can be constructed by compounding several adders. The efficient hybrid adder of binary carry-look ahead and carry-select adder is developed in this design. We can split the adder up into several segments, each segment consisting of two BCLA adders. Connections among the carry propagation can be used in the CLA adder [Hwa79]. Figure 4-23 shows a 16-bit carry-select adder with group carry

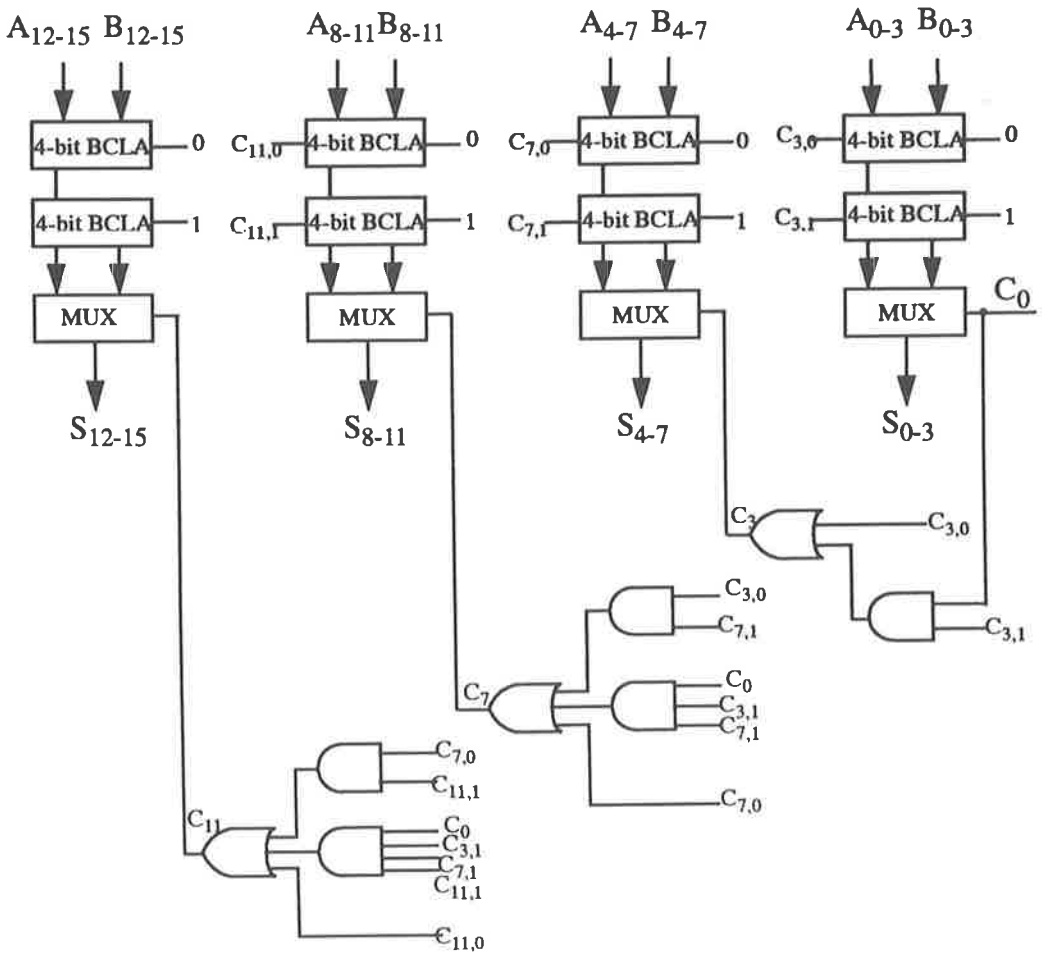


FIGURE 4-23. A 16-bit carry-select adder with carry generation.

generation. This 16-bit adder can be composed of different segments. In order to readily describe group carry generations, we divide it into four 4-bit segments, and assume that $C_{i,0}$ is the carry of the ith $(i = 3, 7, 11)$ bit when the carry-in of the block is 0; and $C_{i,1}$ is the carry of the ith when the carry-in of the block is 1.

The select line of the carry selector can be simplified as expressed by Hennessy and Patterson [Hen90]. According to Moorgan's law:

$$C_3 = C_{3,1} \cdot C_0 + C_{3,0},$$

$$C_7 = C_{3,0} \cdot C_{7,1} + C_0 \cdot C_{3,1} \cdot C_{7,1} + C_{7,0} = C_{7,1} \cdot C_3 + C_{7,0},$$

and in the same way,

$$C_{11} = C_{11,1} \cdot C_7 + C_{11,0}.$$

The connections between blocks are shown in Figure 4-23. It can be seen that the whole carry propagation is obtained through only one AND and one OR gate delay, even though they are irregular and require an AND gate with large fan-in.

A simpler way to build the carry generation circuits shown in Figure 4-24 is

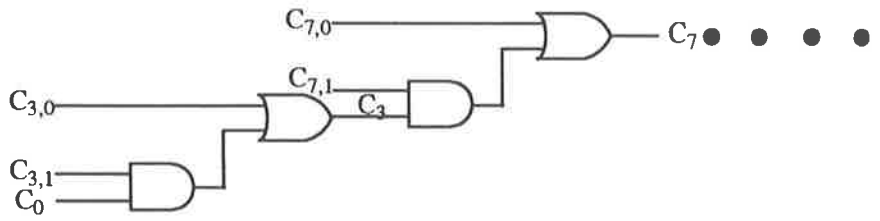


FIGURE 4-24. A simplified carry generation of a carry-select adder.

slower than that of Figure 4-23. However, it improves regularity and feasibility in GaAs technology. In addition, we can change the segments' compound to compensate for the carry propagation delay. Figure 4-25 shows how to realize the function in Figure 4-24 in terms of practical GaAs gates. This approach is employed in our design.

When adding two multi-bit numbers, to make up for the carry propagation delay, the time taken to complete a sum operation should be close to the time taken to

C_{i1}	C_{i0}	C_{i-1}	C_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

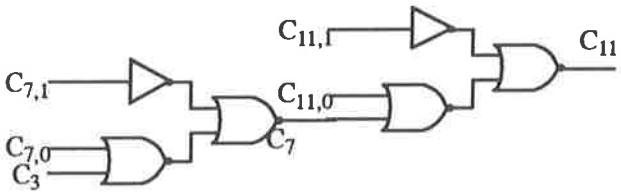


FIGURE 4-25. Carry generation of a carry-select adder in GaAs.

generate the carry of the previous stage. The total time to complete the previous stage's carry is an n-bit sum plus a delay for carry generation as in Figure 4-4 for example. Therefore, the following stage has time to complete a larger sum before the previous stage's carry is generated. This time is typically only enough for an extra bit of addition, and thus the number of bits per stage is incremented by 1 for each stage, i.e. n, $n+1$, $n+2$, etc. The multi-bit addition total delay is estimated by the following formula:

$$T_{multi-bit} = T_n + W \cdot \tau \tag{4-3}$$

where T_n is the delay of the minimum bit block; W is the number of divided segments; τ is two NOR gates' delay shown in Figure 4-25. n cannot be too small, otherwise, W will increase so that the total delay increases. On the other hand, n can not be too big as in this case T_n increases resulting in a large total delay increase, even though W decreases.

For the 56-bit addition to minimize the delay, a possible way is to divide it into seven segments: $4+5+6+7+8+8+9+9$. According to equation 4-1, the 56-bit addition time delay is approximately:

$$T_{56} = T_4 + 7 \cdot \tau \tag{4-4}$$

where T_4 is a 4 bit BCLA adder delay.

In fact, several factors, including area and placement and so on must be taken constantly into account in implementation. The formula 4-2 has not been accepted in our design. Details of how to split-up the 56-bit addition will be presented in the next chapter.

4.3.3 Normalization

There are two cases which require normalizing after adding the two mantissas: the first case is when adding two mantissas with different signs, the most significant bit of the result may not be 1 and it must be normalized. Table 4-3 represents a 56-bit mantissa value register containing a floating-point number. In this table, the leftmost bit is a sign, and the result requires shifting 8 bits from right to left so that the hidden bit is 1 (the hidden bit is implicit) after normalization.

Table 4-3. A 56-bit register of mantissa value.

Unnormalized Positive Number	0 0000000111110000...
After Normalization	0 11110...000000000....

There are three steps for normalization in this case: (a) determine how many bits need to be shifted; (b) shift the mantissa result into standard form; and (c) adjust the output exponent [Hok90]. A priority detector (a leading one detection), an encoder, and a 6-bit incrementer can accomplish the function of (a) as shown in Figure 4-26; the function of (b) can be completed by a 6-stage normalization (left) shifter. A 11-bit subtractor can adjust the exponent.

The second case for normalization is when an overflow occurs; the mantissa result must shift one bit right and the exponent must be incremented by 1.

For high speed operation, these two operations occur in parallel, as a result, it is important to accelerate the first case process as much as possible. A MUX is able to

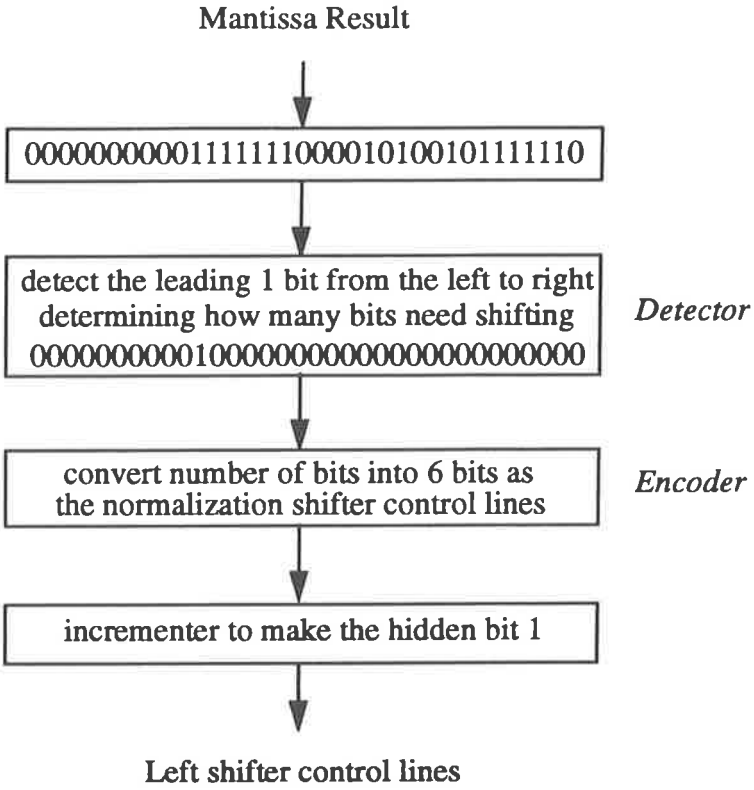


FIGURE 4-26. The flow chart of normalization step (a).

choose one of them as the mantissa output. Moreover, if the select lines of the left shifter are 000000, then the normalization result is exactly the result obtained after adding two mantissas. This special case is involved in the first case. Figure 4-27 is a block diagram of mantissa result processing.

In general, a widely used method to deal with the first case is to use a priority encoder [Man79] [War82] for acceleration. The logic of the priority encoder is such that whenever two input levels arrive simultaneously, the input having the highest priority will take precedence to ensure that only it is encoded. Figure 4-28 shows a 4-bit priority encoder. Referring to Figure 4-28, the priority detector can be implemented by NOR gates. It appears to make the circuit much simpler, but it is not practical for GaAs implementation when the number of inputs are high. As a result, we adopt an

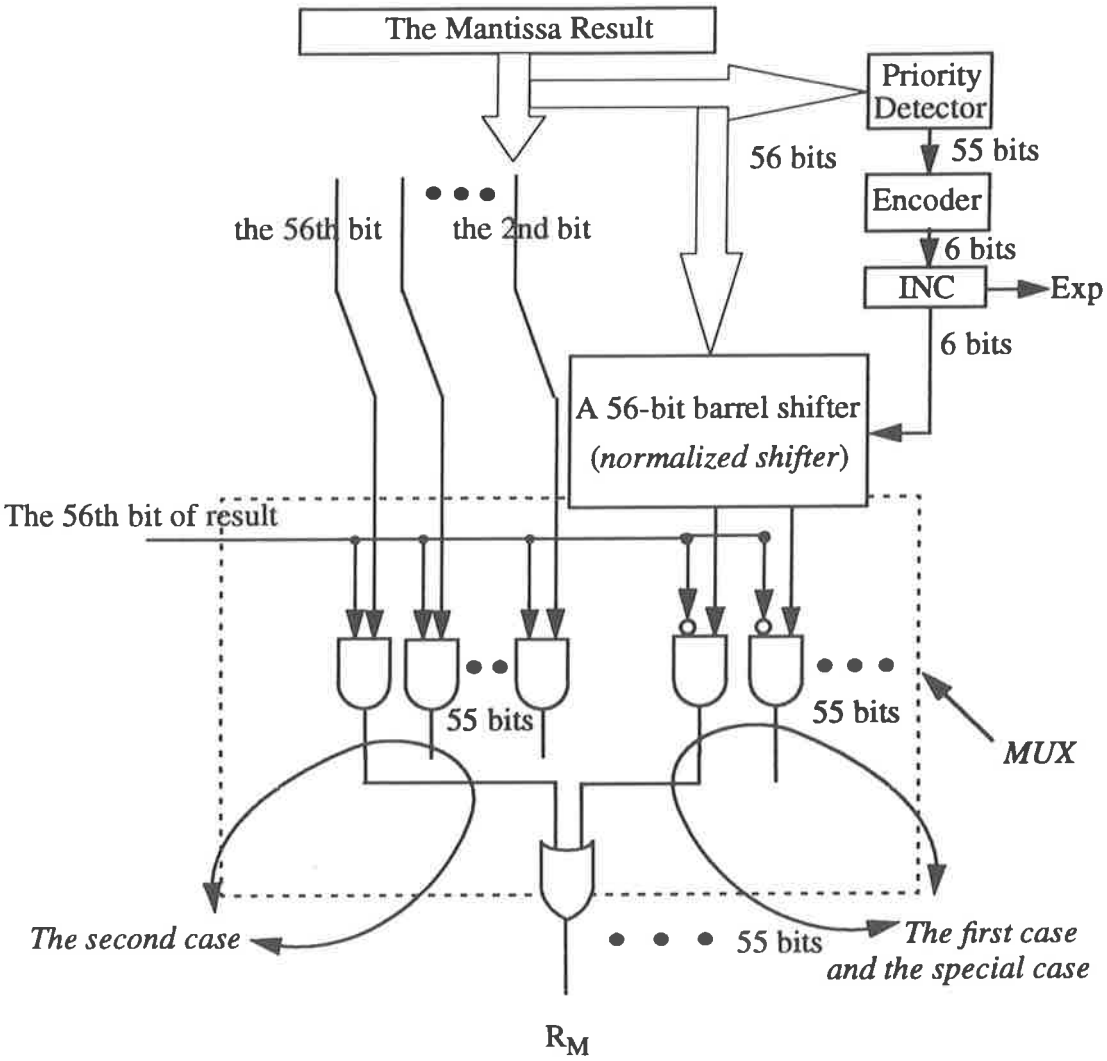


FIGURE 4-27. A block diagram of the mantissa result.

alternative approach to priority detection and encoding as shown in Figure 4-29.

4.3.3.1 Priority Detector

The first leftmost 1's position (the leading 1) of the mantissa is detected using a priority detector [Bur94], which detects and keeps it, and then sets the remaining bits on its right to zero.

If, for example, the input $A_3A_2A_1A_0 = 0011$ is as presented in Figure 4-29,

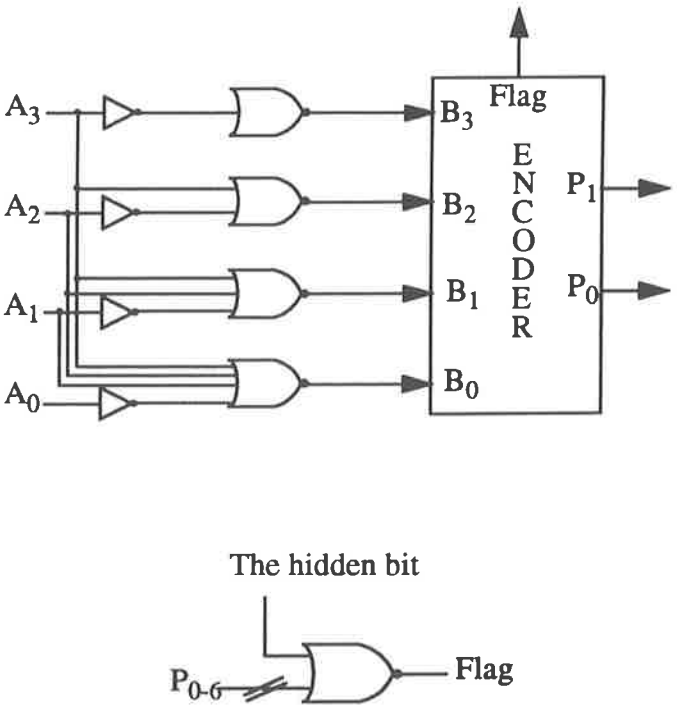


FIGURE 4-28. A 4-bit priority encoder.

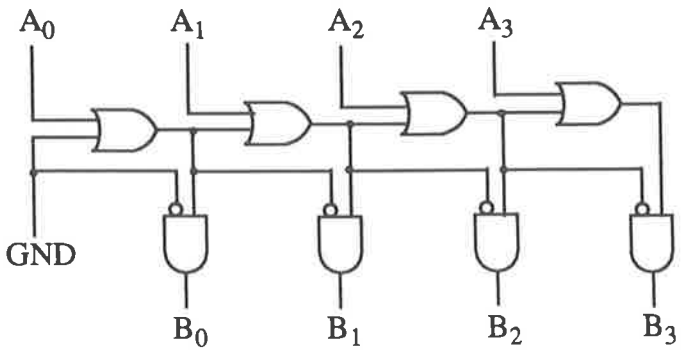


FIGURE 4-29. A simple 4-bit priority detector.

then the output becomes $B_3B_2B_1B_0 = 0010$. The position of the leading 1 of the number in this case is the 2nd from the right. In fact, considering the hidden bit, $A_3A_2A_1A_0$ must be shifted by 3 bits to accord with IEEE 754 standard. The priority detector consists of OR and AND gates. The connections between the OR gates are in a cascade form. If the input number is 55, the detector output takes $55 \times OR$ gate's delay. Clearly, it is simple but slow.

An accelerated priority detector shown in Figure 4-30, may be an acceptable method. It is much quicker than the detector of Figure 4-29, although this circuit

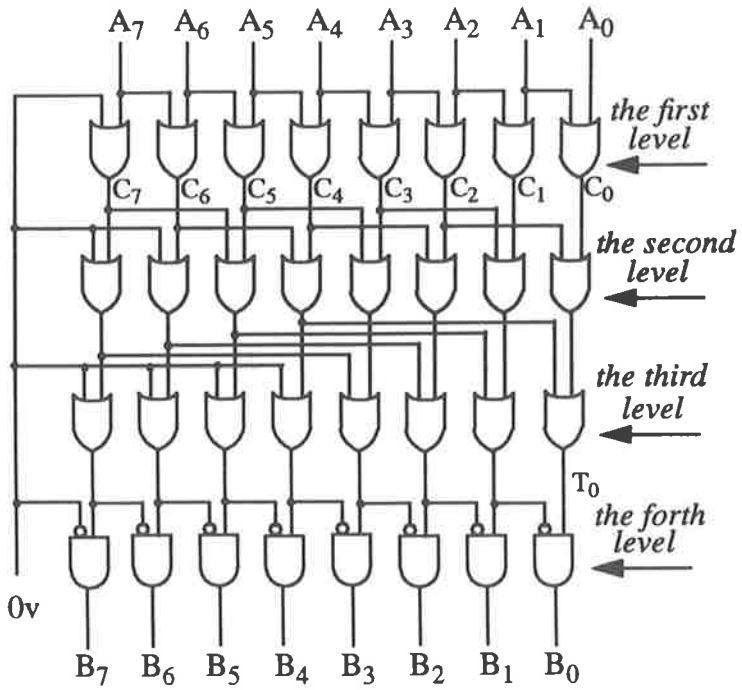


FIGURE 4-30. An 8-bit accelerated priority detector.

requires a larger number of gates and the connections are particularly complicated. For a 56-bit input detector, there are 6 OR gate levels and one AND gate level, and each level consists of 56 OR or AND gates. The interconnections are so complex that the signal propagation delay rises greatly. Also due to irregularity, the layout is very difficult. To overcome the signal propagation problem, an effective way to simplify the connections is to break the 56-bit inputs into 7 8-bit-input blocks, and then link

them together. Compared to a single 55 bit detector, it saves quite a number of gates and power (about 40%), and makes the 56-bit detector practicable and simple. The connections of a 16-bit-input detector between the first two blocks are represented in Figure 4-31. Figure 4-32 shows the connections among 7 blocks. The total delay is approximate 8 OR gates plus 1 AND gate as shown in Figure 4-32.

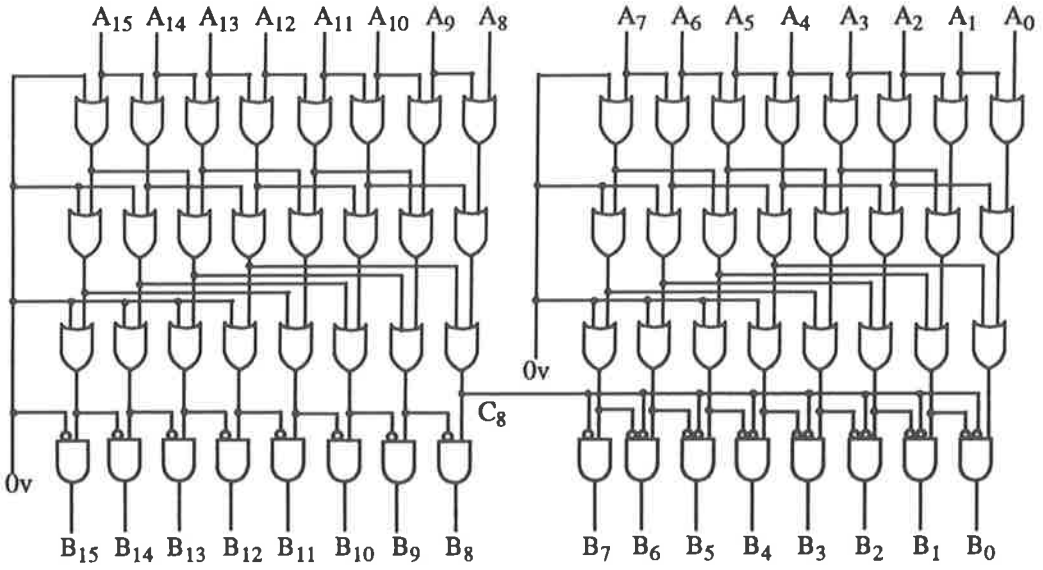


FIGURE 4-31. A diagram of 16-bit priority detector.

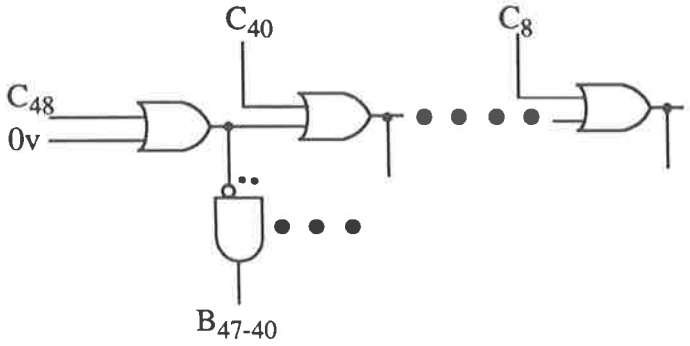


FIGURE 4-32. The connections among 8-bit detector blocks.

Referring to Figure 4-31, an 8-bit priority detector requires 24 OR gates and 8 AND gates. The 56-bit priority detector, thus, needs $24 \times 7 OR$, and $8 \times 7 AND$ gates besides the propagating gates shown in Figure 4-32. It not only occupies a large area, but also consumes much power and the interconnections are very difficult.

For a simple implementation of Figure 4-30 in GaAs, we can use PLA methodology. The first three levels are all OR gates, which are implemented as in Figure 4-33. Each output in each level can be obtained by a NOR gate and an inverter. In the

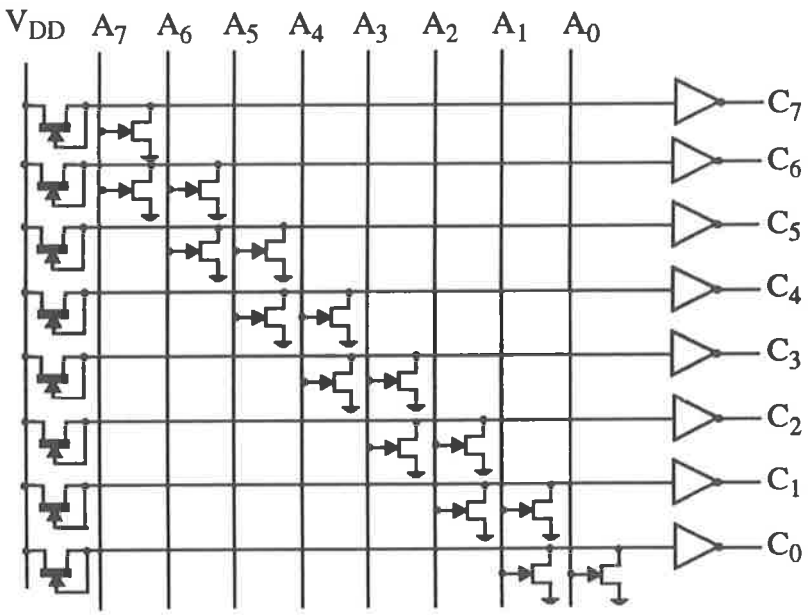


FIGURE 4-33. The implementation strategy of figure 4-30 in PLA.

next chapter, we will present its layout. Algebraically, the AND gates in the fourth level can be implemented using NOR gates. Its PLA pattern is similar to Figure 4-33, but only its inputs are different from Figure 4-33. Nevertheless, taking account into temperature influence in PLA circuits, several factors of PLA implementation including power consumption, delay and noise margin should be noticed and simulated carefully. Chapter 5 will present how these factors are affected as temperature changing.

4.3.3.2 Encode

Having detected the position of the leading 1, the next step is to convert this position to a binary number to control the normalization shifter. The position of the leading 1 of the 56-bit mantissa is encoded into a 6-bit number whose maximum value is less than 56. The 2^6 encoder with 56 inputs and 6 outputs can perform this conversion. The 6-bit output value is the shifting count, which must also be subtracted from the exponent.

This encoder can be implemented by means of a PLA. If implementing it directly, the maximum fan-in will be 28 which is unacceptable. In practice, we can break such a large encoder into several segments, and then connect them with some logic gates. This 56 bit input encoder can be divided into 7 segments, each consisting of an 8-bit encoder. For convenience in hardware implementation, we still use PLA methodology to regularize the 8-bit encoder. Figure 4-34 is an example of an 8-input

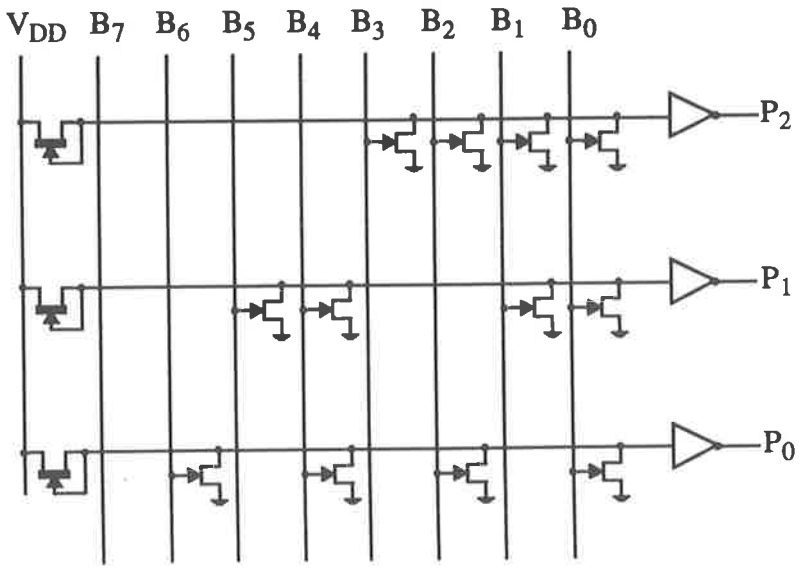


FIGURE 4-34. An 8-bit encoder in PLA.

and 3-output encoder in PLA. Even though the maximum fan-in is 3 in a DCFL logic family, NOR gates can still work properly if the fan-in number is 4 when its output drives only one inverter. At this point, the NOR gate output voltage is slightly lower

than the standard output voltage, which may reduce the noise margin. In this case, SDCFL NOR gates with 4-input can replace DCFL to ensure the reliability of the design. The results will be compared in the next chapter.

The two 8-bit encoders can be connected together to construct a 16-bit encoder as illustrated in Figure 4-35. Figure 4-36 shows the 56-bit encoder consisting of 7 8-

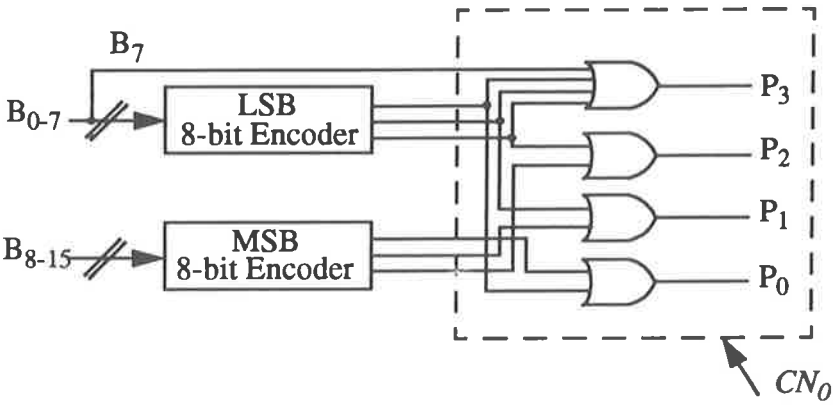


FIGURE 4-35. A 16-bit encoder consisting of two 8-bit encoders.

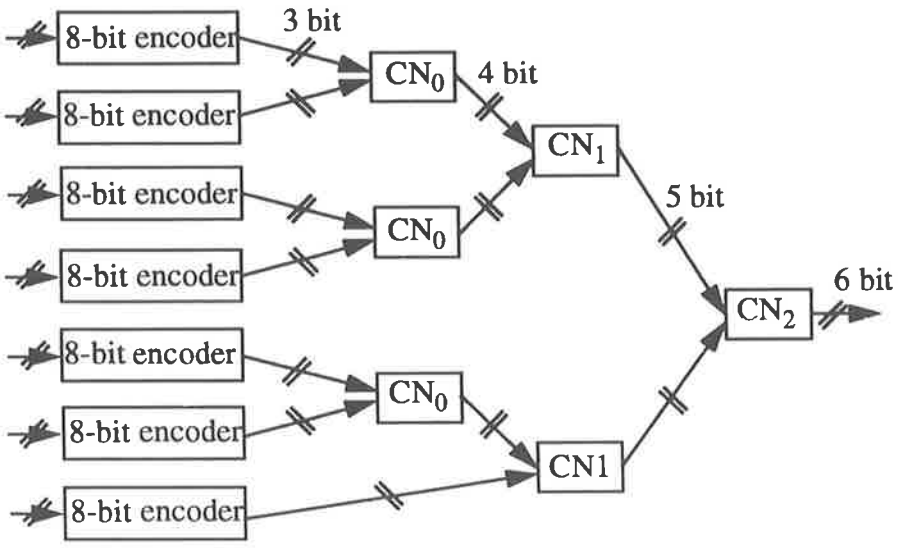


FIGURE 4-36. The connections of 7 8-bit encoders in parallel.

bit encoders connection. The total delay of the encoder is estimated as the delay of an 8-bit encoder plus three OR gates. Using GaAs DCFL, more gates will be required because of fan-in limitation and gate transformation to NOR gates. Certainly, the delay is greater than that of an 8-bit encoder delay and three OR gates.

4.3.3.3 Improved Encoder

Because of the hidden bit, the real shift count is equal to the value of the priority encoder plus one. The procedure for normalization is shown in Figure 4-37.

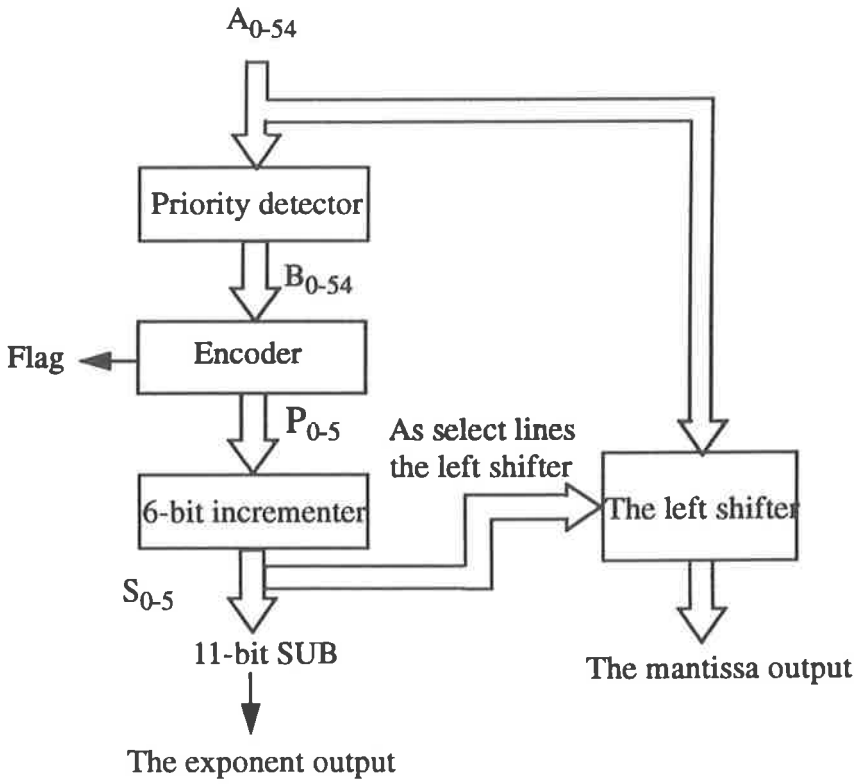


FIGURE 4-37. The normalization datapath.

In general, the faster is the normalization process, the better. In order to accelerate the process, further circuit improvements are shown in Figure 4-38 and Figure 4-39 which are consistent with Figure 4-34 and Figure 4-35. The improved circuit of the encoder automatically adds 1 to the shift count, and the 6 bit incrementer in Figure 4-37 can be omitted.

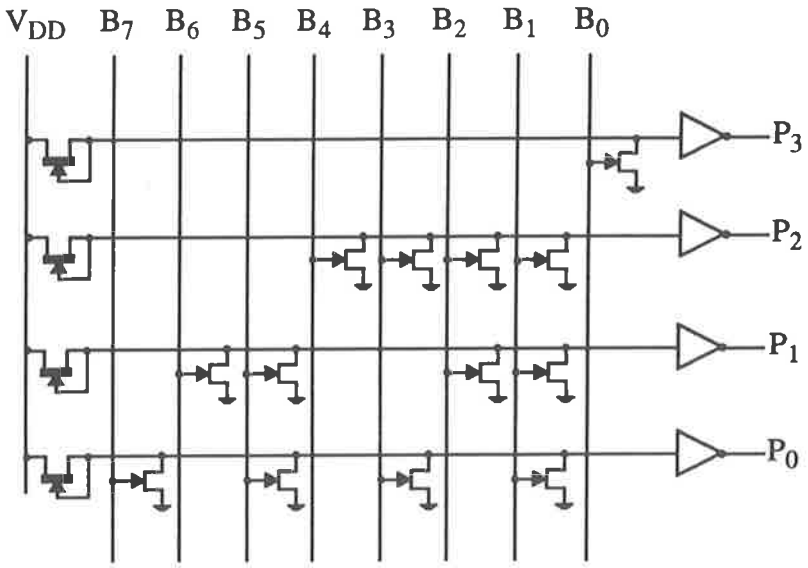


FIGURE 4-38. An improved encoder.

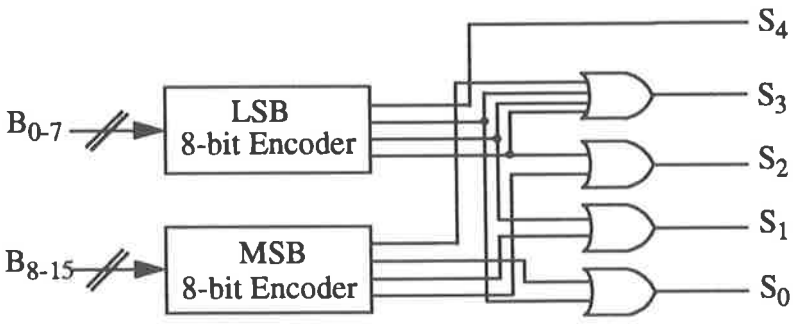


FIGURE 4-39. A connection between two improved 8-bit encoders.

A special case in normalization is if the values of the 52-bit mantissas with different signs are the same, i.e. the 6-bit output of the priority encoder becomes 000000, then the normalization shifter does not shift at all. The mantissa is equal to $0.000000\ldots \times 2^{e-1023}$, that is regardless of the exponent value, the whole opera-

tion result is 0. We can use a flag to represent this case, and set the result of addition/ subtraction to 0 in case of confusion with $1.000000\ldots \times 2^{e-1023}$.

4.3.4 Rounding

Like multiplication and division, the extra 3-bits used for increasing precision, which are at the rightmost position of the 56-bit mantissa result, must be rounded off to satisfy the IEEE standard requirement [Bur94]. Notice that when the mantissa result is 1111111..., and the rounding result has a high carry, the exponent must be increased by 1. There are several approaches which have been reported. An idea to deal with this problem has been presented by Burgess [Bur94] in details. Due to limited time, we do not take this logic design into consideration in this work.

4.4 Summary

Following the description of the addition and subtraction algorithm of the previous chapter, some corresponding digital logic circuits to implement the algorithm have been developed and compared in this chapter. In particular, several different adders and their hybrid examples have been given and analysed. Due to the limitations of GaAs technology and in order to achieve high speed calculation, a multi-bit hybrid carry-select and a BCLA adder is proposed as an optimum and feasible option in our design.

We employ barrel shifters for aligning mantissas and normalising the mantissa to accelerate the shifting procedure. The barrel shifter consists of 6 stages; each stage has 55 (or 56) MUXs. The 6 stages shift 1 bit, 2 bits, 4 bits, 8 bits, 16 bits and 32 bits, respectively. The required number to shift is decided by 6 select lines. The 6 signal select lines in the alignment shifter result from the difference between two exponents, and the 6 select lines in the normalization shifter are determined by the number of bits to be shifted left until the hidden bit becomes 1.

In order to obtain the normalization shift count and overcome the limitations of GaAs, a PLA based priority detector and encoder were developed. We individually break the detector and the encoder into 7 8-bit blocks and then interconnect them. Compared to the alternatives, these circuit improvements either save a number of

gates, or greatly reduce the delay. An improved approach which saves a 6-bit incrementer has been described in the last section of this chapter.

In the logic design stage, we should consider a circuit's realization in physical layout at the same time. Moreover, the factors of speed, area and so forth are interdependent as well. For time-matching, it is often preferable to use some simple circuits instead of rapid and complex ones. For this reason, several circuits to solve the same problem are presented in this chapter. In the next chapter, we will realize the logic circuits presented in this chapter to complete the physical design of the adder/subtractor.

Chapter 5 Circuit Physical Design And Simulation Results

Abstract: *The layout of some typical circuits such as the barrel shifter, the combined adder of the BCLA adder and the carry select adder, and the priority detector are presented as specific examples in this chapter. First of all, the functions of the selected circuits are verified using* ***OCTTOOLS*** *at gate level. Before implementation, for arranging layout blocks within a chip to minimize chip area and maximize speed, it is imperative to map out a floorplan. The circuit's layout is generated by the layout tool* ***MAGIC****. Functional simulations on layout are performed using* ***IRSIM****, and circuit performance including the worst time delay, and the average power dissipation are measured using the* ***HSPICE*** *circuit simulator.*

5.1 Introduction

In the second chapter, GaAs technology was analysed at gate level. In this chapter, we will use the Vitesse H-GaAs-2 $0.8\mu m$ with three levels, E/D mode GaAs fabrication technology, and at 2V supplies, to realize the project's goal -- i.e. implementation of a floating-point adder/subtractor for the special-purpose accelerator.

DCFL, SDCFL and SBFL are the three logic families employed to complete the circuit realization. DCFL is widely used in the assign of fundamental logic elements, due primarily to its low power and high density. SDCFL and the SBFL are generally used when high fan-out and fan-in are required. Experimental results discussed in Chapter 2 have demonstrated that both the SDCFL and the SBFL have good low noise margin. SBFL also has a good high noise margin. Although the SDCFL inverter delay is slightly smaller and occupies less area than the SBFL's, it consumes about 5-6 times the power of SBFL. In this design approach, we make efficient use of both as buffers in different cases. The SBFL NOR gates are never used in design as explained before, however, the SDCFL NOR gates are sometimes employed. Detailed application examples will be given in the next several sections.

Building a workable chip is the ultimate design purpose. The layout performance is greatly improved by selecting suitable circuit parameters. In other words, the performance of all parts of the circuit relies heavily on the geometry of ratioed gates. As stated in the second chapter, there is a significant difference in speed performance between identical circuits having different transistor ratios and sizes. If we choose small transistors to implement a circuit, the circuit operation may be very rapid, and have a high density. However, other performance parameters may be poor. Therefore, for a high speed operation, simply scaling down the transistor size is not adequate. An alternative approach to achieve higher speed performance is to optimize circuit structure and chip design.

In order to support such VLSI circuits, a process must have, in addition to good device switching times, high levels of integration, good yields, reasonable power dissipation, and dense, multilevel interconnect. For a digital logic family, gates should have good load-driving characteristics, reasonable noise margins, and support by appropriate design automation tools.

Normally, in VLSI circuit design, speed is not the only important criterion in judging a circuit [Bre82] [Okl85]. There are many other important criteria, such as power density, layout area and noise immunity, all of which must be compromised. Other practical aspects such as interconnection which permits a high communication bandwidth with minimum delay, driving capacity, thermal management, regularity, routing (data buses, power and ground buses) and placement must be considered as well.

In a large chip, the major determinant of chip density is the width of the tracks providing the interconnection. Narrow tracks cause larger delays while wide tracks take larger area. In order to minimize the area of the module and reduce the interconnection of the modules, an efficient floorplan, which shows modules size and placement, should be prepared early in the design process. In mapping a floorplan, modules should be placed so as to minimize interconnection length. Moreover, for fabrication, the chip's frame shape should be as square as possible. Taking into account these factors, the floorplan shown in Figure 5-1 was adopted for the adder/subtractor.

In the design of digital VLSI circuits, normally, low power dissipation and high speed are conflicting criteria. For example, to increase speed performance, the more or larger transistors are required. At the same time, due to heat dissipation limits, an exceedingly high density chip is not suitable for GaAs design [Mil86$_1$]. Since heat density is limited [Ras86], larger area (high cost) is required for dealing with high power dissipation. Hence, the speed-area trade-off is power density. To address these trade-offs, the best approach is with both minimum power and maximum speed using a certain area. In addition, to overcome noise disturbance, it is necessary to ensure that every output point of any gate should have a high enough high-voltage (V_{OH}) or a low enough low-voltage (V_{OL}) in every stage. That is because they ensure the logic functions correctly in the presence of noise and ensure enough voltage to make the input transistor of the next stage turn-on or turn-off completely, and also to avoid crosstalk, we let the data path lines, the power and the ground buses run horizontally, and the control signals run vertically, as indicated in Figure 5-1.

Figure 5-2 illustrates a typical ring-notation mask layout of an inverter. Clearly, the V_{DD} and GND routing lines should be wide enough to avoid defects due to the large current density electron migration phenomena. Nevertheless, considering the

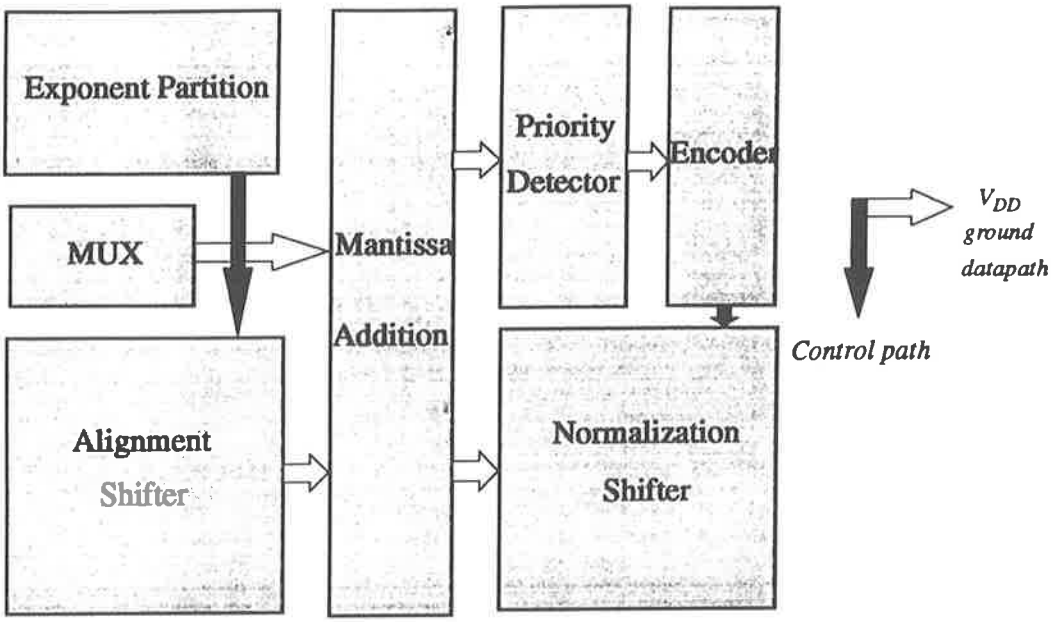


FIGURE 5-1. Floorplan of a floating-point double precision adder/ subtractor.

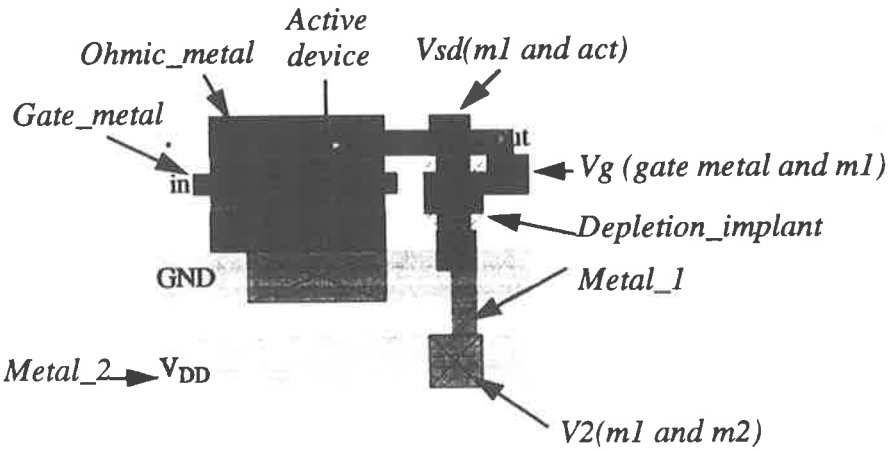


FIGURE 5-2. One typical layout example for an inverter.

area, in the actual design, it is unnecessary to have very wide routing lines.

In order to minimize fabrication cost, the layout area should be as small as possible, but not too dense. Too high density may produce coupling between signals and V_{DD} and GND, which may introduce errors in the operation of the circuit. High layout density may also produce high power density which may damage the chip. The other major techniques conserving the design space, are to creatively simplify the circuit such as a multi-MUX design which is a typical example as employed in the shifter and the addition designs in layout, and place modules carefully.

In this chapter, we shall present four characteristic layout examples which are the 56-bit barrel shifter, the 56-bit adder, the priority detector, and the encoder.

For simplicity of the design process, we break the multi-bit functions into several segments. Each segment can be laid out as a small standard and geometrically regular cell, and then replicated, and finally are connected cells together to achieve the desired function. Applications of logic arrays and the regularity allow us to focus our time on adapting the key logic structure to extract highest performance from GaAs technology. In the addition and normalization section, several such sub-cells have been conceived.

The logic and circuit simulators developed in recent years are powerful tools for verification and optimization design. The CAD tools used in this process were **OCTTOOLS**, **MAGIC**, **IRSIM** and **HSPICE**. At first, the **OCTTOOLS** was used to check every logic circuit of the whole process designed in Chapter 4 at the gate level. The layout editor **MAGIC** was used for physical implementation. The resulting layout was simulated by **IRSIM** and **HSPICE**.

When simulating the result with **IRSIM** and **HSPICE**, there is a problem that when a gate drives 4 fan-out, regardless of the logic family, its output high-voltage may be around 450mV. **IRSIM** is a fast functional simulation tool for the digital circuits. Unfortunately, it can not accurately provide as much detail of circuit performance for GaAs technology as for CMOS. This value is regarded as a high-voltage output with the **IRSIM** tool, however with **HSPICE** this voltage may not be high enough to drive the next gate because of noise effect. As a result, in physical layout, although the circuit function might be verified by **IRSIM**, it may not work in practice.

From this point of view, we must ensure the layout can work with **HSPICE** simulation.

IRSIM, therefore, is generally used to verify the layout logic functionality. The resulting performances such as the output values of low and high-voltages, the maximum current, the average, peak power dissipation, delay and so forth can be measured by the circuit simulator **HSPICE**. The area and transistor count can be evaluated in **MAGIC**. In fact, it is difficult to choose the best circuit from many different alternatives even by using powerful software tools. In next sections, details of design tradeoffs will be presented.

5.2 Barrel Shifter

Unfortunately for GaAs technology, implementing a barrel shifter requires a large number of transistors. As the transistor count increases in GaAs, automatically, the power dissipation and the area rise rapidly.

In the implementation of a CMOS barrel shifter, n-channel pass transistor gates based on N-MOSFET are generally better, have good noise immunity, are not sensitive to increase in fan-in and fan-out, and are easy to layout [Mil86$_2$]. In GaAs technology, it is possible to use differential pass transistor logic in some applications [Pas91][Mit92][Lon92]. Experimental results demonstrate that if using E-mode devices to configure the transmission gates, the high noise margin is decreased; if using the D-MESFET technology, the noise margin is good enough, but a negative voltage supply must be provided. For these reasons, we still use DCFL, SDCFL and SBFL as the three logic families of choice for this implementation in spite of occupying large space.

There are two most difficult problems considered in the barrel shifter physical design. The first problem is how to place each level optimally for placement of the whole module. Because a signal which propagates through a long wire between two stages incurs a large time delay, the 56 MUXs in each stage can be divided into two segments for complicated connections, as shown in Figure 5-3. In addition to that, the second difficulty is that each control signal must control 56 MUXs in every stage and require a long wire to propagate the control signal. For solving the second problem to

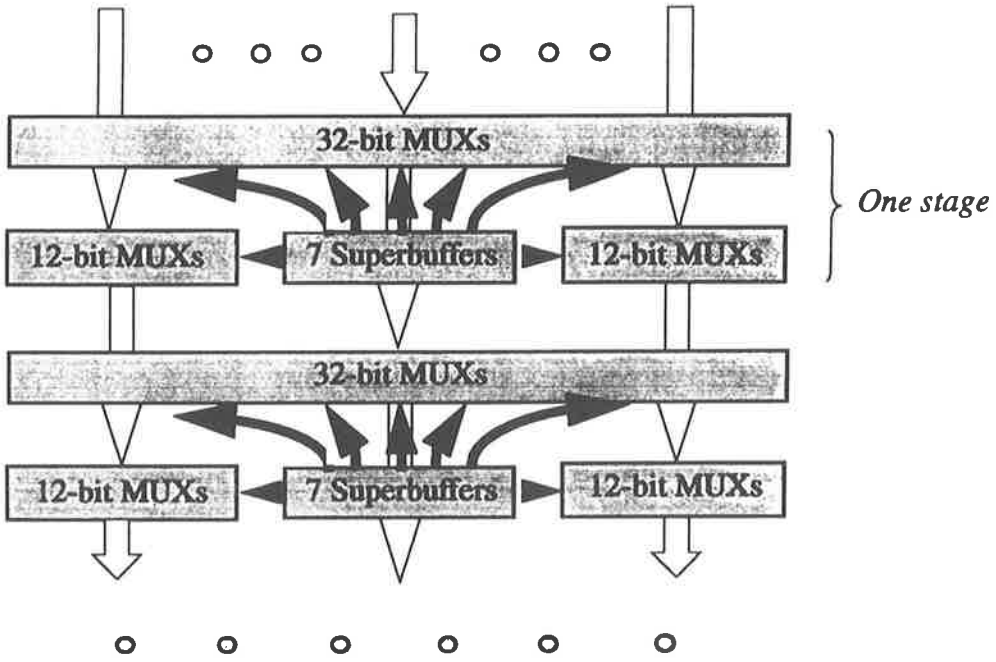


FIGURE 5-3. A 56-bit barrel shifter floorplan.

increase the speed and noise immunity of the shifter, we use a group of 7 amplified SBFL buffers, each driving 8 MUXs. At the same time, there is another group of 7 amplified SBFL buffers and a SDCFL inverter for minimizing area and power consumption, to generate the opposite control signal for MUX. Thus, 336 DCFL inverters can be saved in the shifter. The two groups of SBFL buffers at each level are placed between the MUXs of the second row as indicated Figure 5-3.

Figure 5-4 is the layout of the alignment shifter. It consists of 14 rows, where the leftmost two rows are a 2:1 MUX to choose the required mantissa for shifting (see Figure 4-21), and the remaining rows consist of a 6 stage barrel shifter. The total shifter occupies 1.08 mm by 1.20 mm and includes around 3500 devices.

Assuming that mantissa A is all 1, and mantissa B is all 0, the results of functional simulations of the barrel shifter via **IRSIM** are shown in Figure 5-5. The signal S_0 is to select the shifted mantissa A or B. If S_0 is high, A will be shifted. *Controlline*

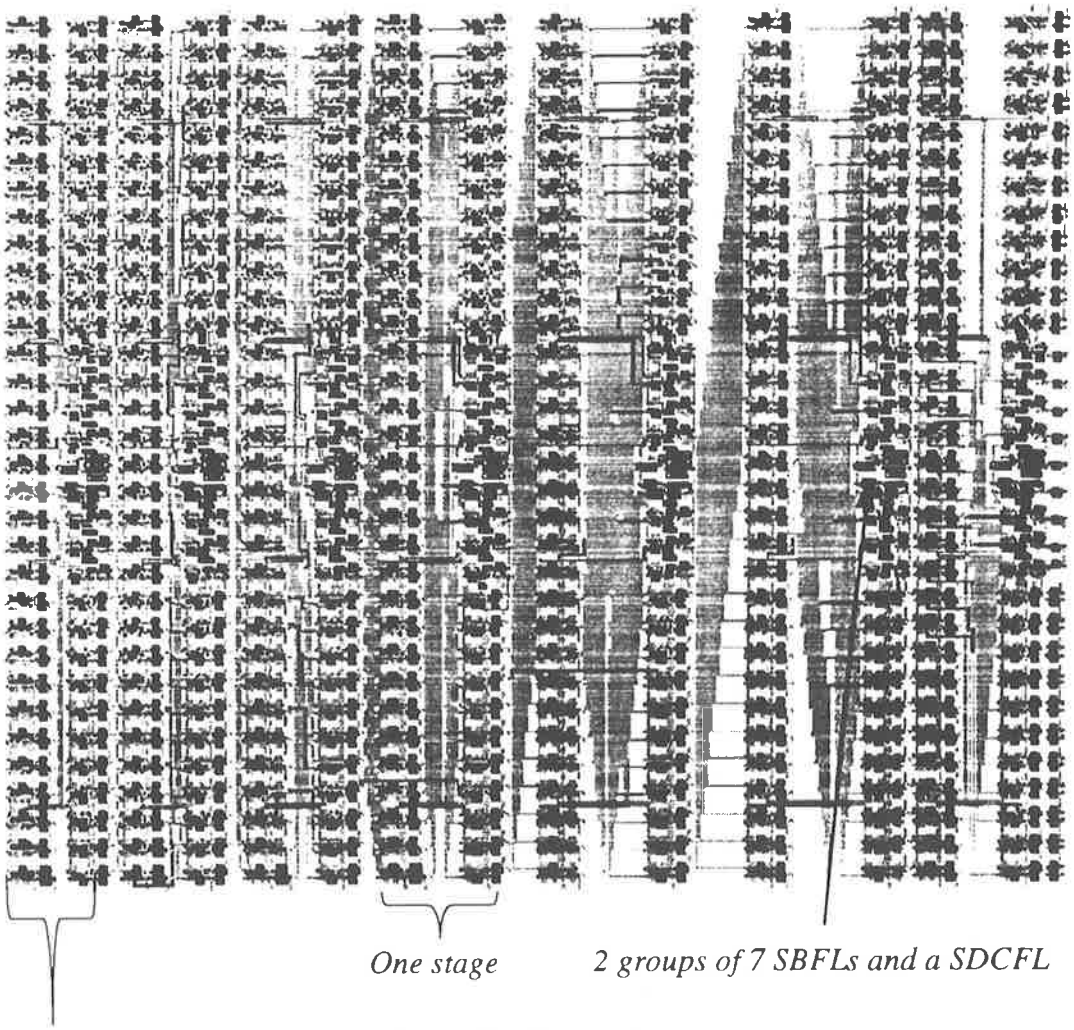


FIGURE 5-4. The layout of the 56-bit barrel shifter.

determines the number of bits shifted. All data are expressed in hexadecimal representation, except *controlline* which is represented in a binary.

The **HSPICE** simulation results are shown in Figure 5-6. The mantissas of A and B have the same values as for the **IRSIM** simulation. P_0 represents one bit of the shifted mantissa which is the one with the smaller exponent. The third plot shows the

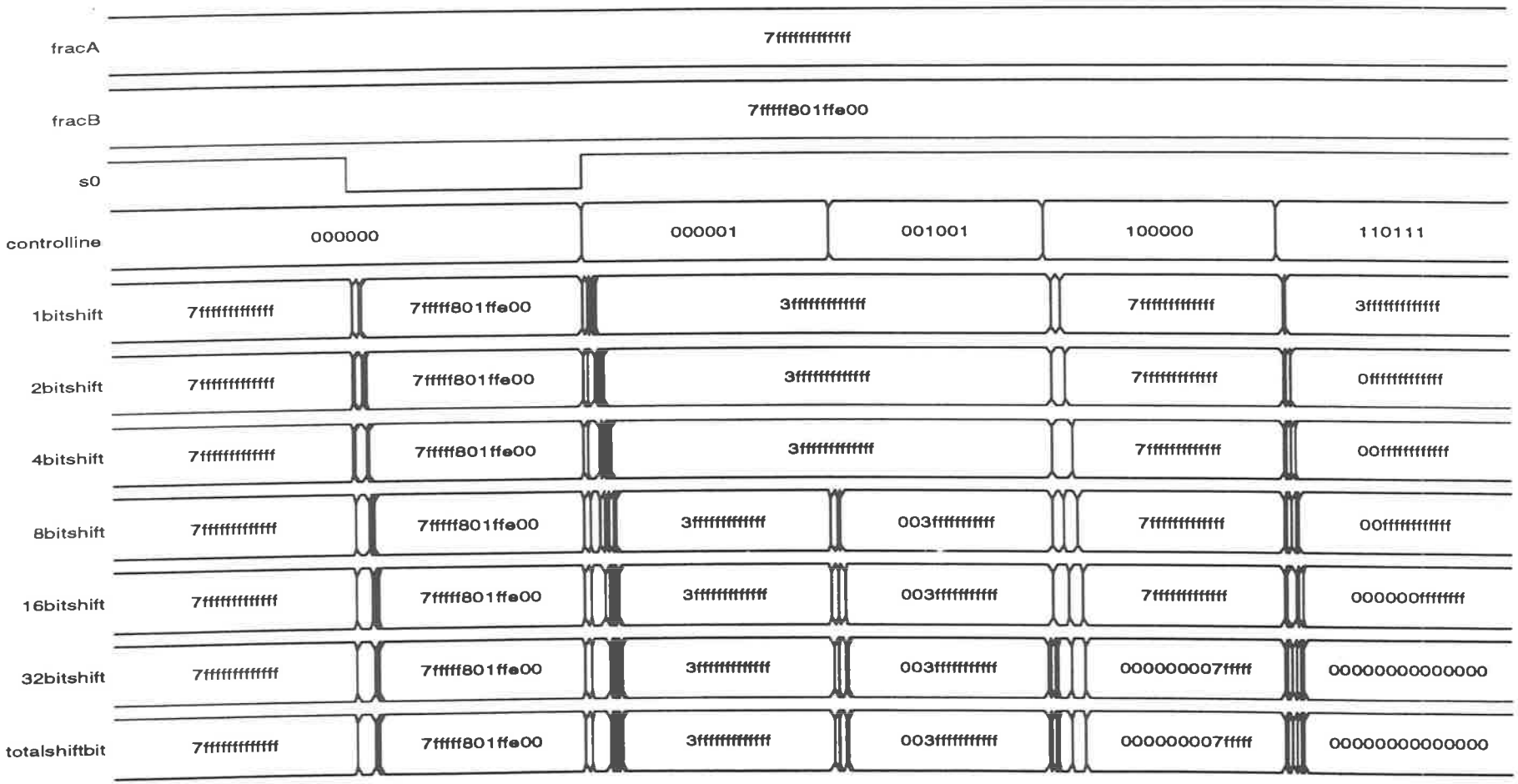


FIGURE 5-5. Functional verification of a 56-bit shifter.

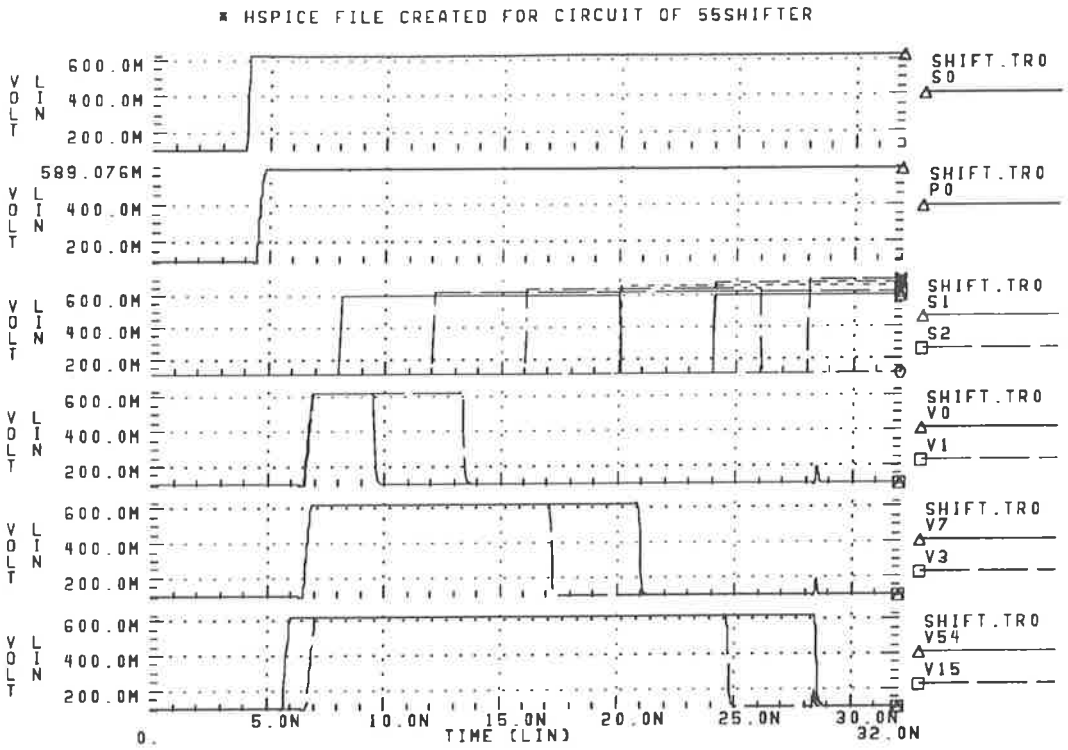


FIGURE 5-6. HSPICE simulation results for the barrel shifter.

6 bit control signals, which determine the number of bits to be shifted. If they are all low, no stage shifts. The last three waveforms show several results in which the observable points are selected. The measured results indicate that the average power dissipation is slightly less than $320mW$ and the critical delay through the 6 stages is approximately $2.3ns$.

The normalization shifter also uses the barrel shifter structure with the layout as shown in Figure 5-4, but with the two leftmost rows deleted.

5.3 Mantissa Adder

As explained in Chapter 4, a 6-bit hybrid adder of the BCLA and the carry select adder has been developed. The floorplan and its relevant layout are respectively illustrated in Figure 5-7 and Figure 5-8. The *exor1* shared part calculates $P_i = A_i \oplus B_i$.

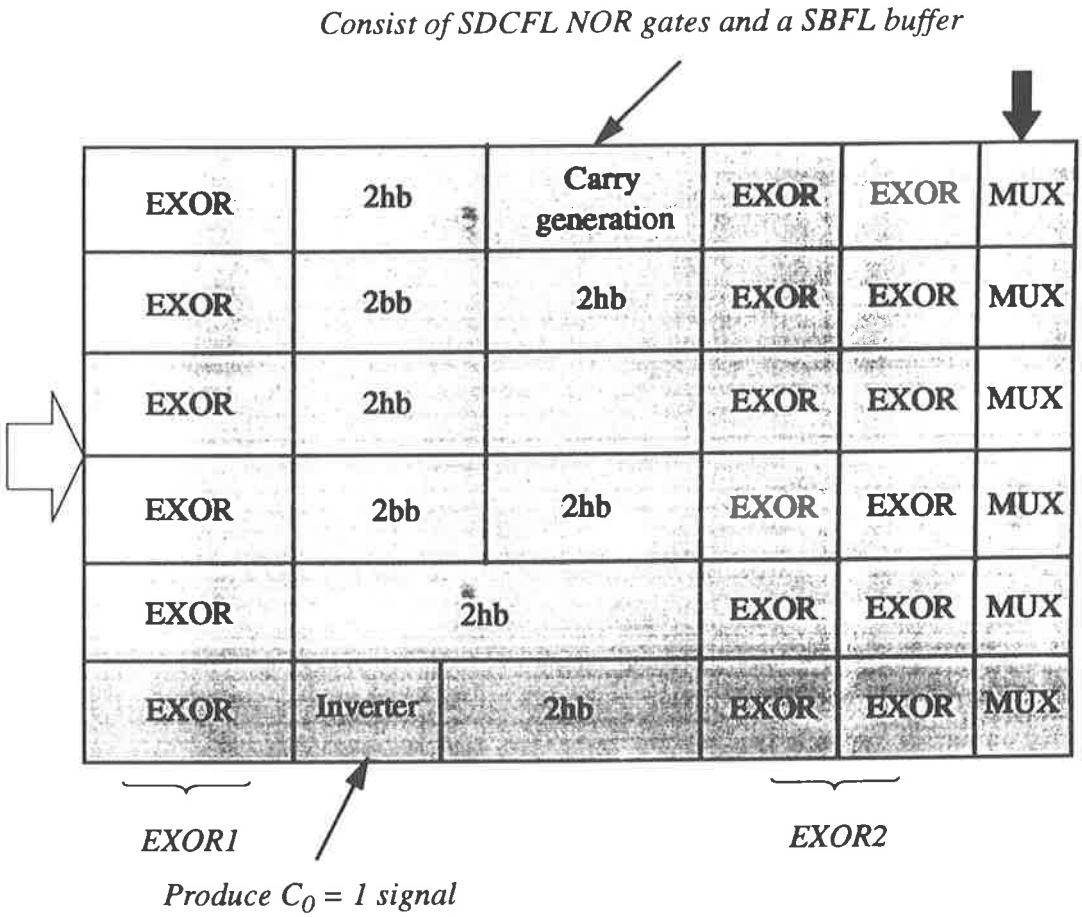


FIGURE 5-7. Floorplan of a 6-bit combined adder.

The bb_i and hb_i are the black cells and the half black cells for carry propagation. The *exor2* part is to calculate $S_i = P_i \oplus C_i$ when $C_0 = 0$ and $C_0 = 1$. The resulting sum is selected by the 2:1 MUXs. To obtain $C_0 = 0$ the signal can be connected to GND. It is not practical to connect C_0 to V_{DD} to set $C_0 = 1$ as the very high input signal ($2V \gg 0.65V$) would pass a huge current through the input transistor. Therefore, we use an inverter whose input is connected to GND to obtain $C_0 = 1$ signal with a typical high-voltage input.

Referring to Figure 4-6, an odd bit of the BLCA adder such as 4-bit, 6-bit or 8-bit in the carry generator block has only one half black cell without black cell. The carry generation of the carry-select adder selecting the output of the next subadder

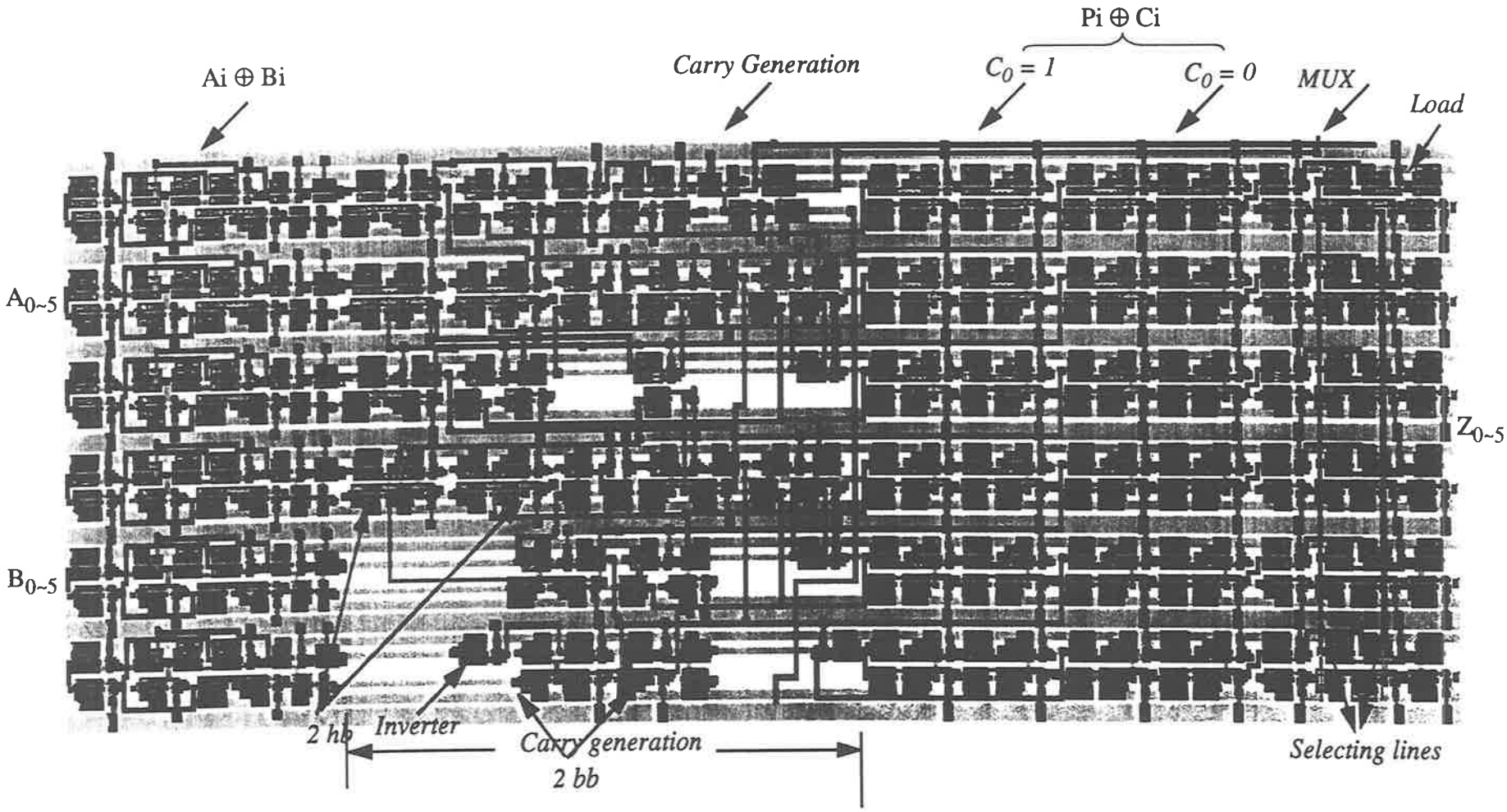


FIGURE 5-8. The 6-bit combinational adder of the carry-select and the binary look-ahead adder layout.

block could be located in the white cell of the carry generator block to save space. The output of carry generation as a selecting line should control the n MUXs of the next subadder output (n is the size of the next subadder in bits), therefore, the carry generation can make effective use of a SDCFL inverter (see Figure 4-24). Meanwhile, using one SBFL buffer connected to the output of the SDCFL yields an inverted select line, saving n inverters. Thus, if there are W (by definition formula 4-2) subadders in the 56-bit addition design, $W-1$ SBFL inverters can substitute for the 56 inverters to produce the other select signals of the MUXs.

For optimum placement within the floorplan, the adder should be laid out to be as long and as narrow as possible. An 8-bit BLCA adder requires two *bb* (black) cells and one *hb* (half black) cell, and all three cells are relatively wide. To reduce the width, the maximum number of subadders that we accept in the design is 6, in spite of losing the speed due to W increasing. Its width is about $520\mu m$ in contrast to $600\mu m$ for an 8-bit adder, both of which have the same height of $1890\mu m$. The addition is structured into 4 (4-bit BLCA adder) + 4 + 6 + 6 + 6 + 6 + 6 + 6 + 6 + 6. Hence, the 56-bit addition requires 9 carry generations to select the output of each subadder. The longest path (bottleneck) counted for the 4-bit BLCA adder circuit is about $1.5ns$ plus the delay of the nine carry generators which is about $9 \times 300ps$, namely the total delay is $4.2ns$. Its density is about $4400 transistors/mm^2$ for this addition.

The results of functional simulation of the 56-bit addition are presented in Figure 5-9. Although the resulting '*SUM*' shows the effects of logic delays, the exact delay time can not be measured by **IRSIM**. It was found that when input states of two mantissas *'fracA'* and *'fracB'* change from one value to other, the worst delay occurs when A remains in the 101010... state, while B changes from 111111... to 010101.... Figure 5-10 shows this critical case of the 6-bit combined adder. Thus it can be seen that the worst delay of the 56-bit adder occurs at the same time as that of the 6-bit adder. It is worth noting that although this hazard does not influence logic functions in synchronous design, it will affect asynchronous design.

The accurate delay can be measured by **HSPICE**. The worst case delay of a 4-bit hybrid adder measured is $1.7ns$ and a 6-bit is $2.8ns$. The case often regarded as giving the worst case delay, when both operands change from 000000 to 111111, gives a delay through the 6-bit adder of only $1.02ns$. Obviously, it is not the overall critical

FIGURE 5-9. The 56-bit adder functional verification.

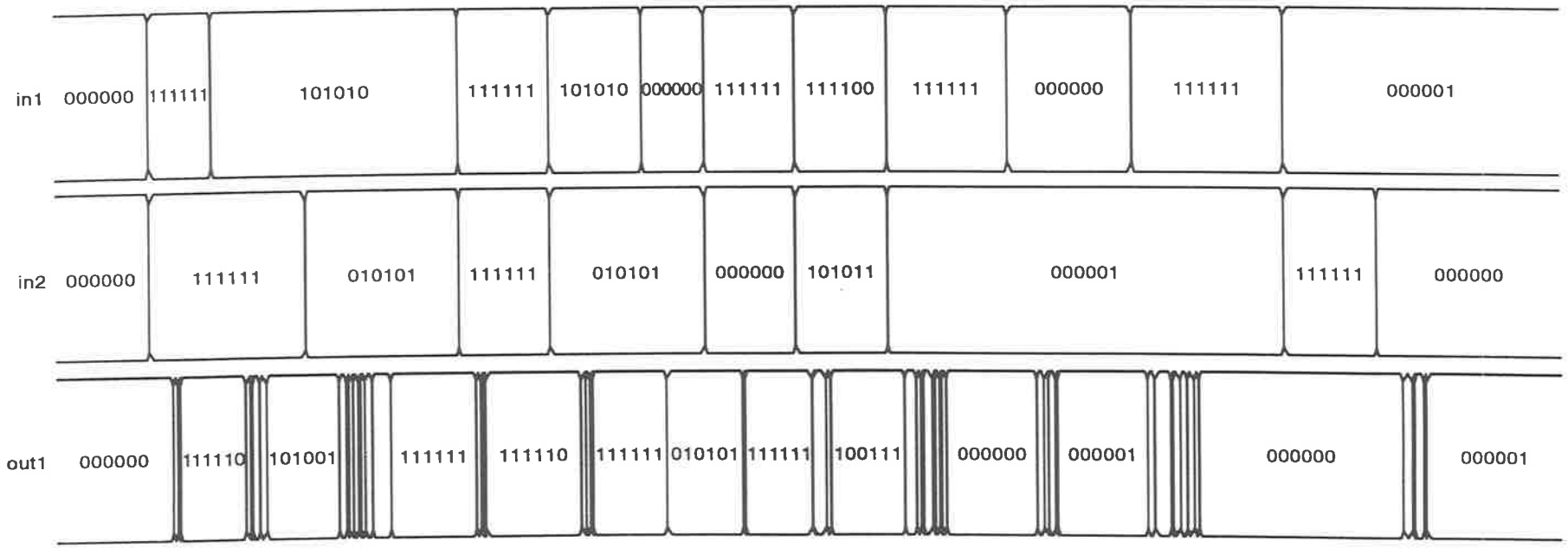


FIGURE 5-10. The critical datapath through the 6-bit addition.

delay in our design.

Figure 5-11 shows the simulation results for several combinations of operands.

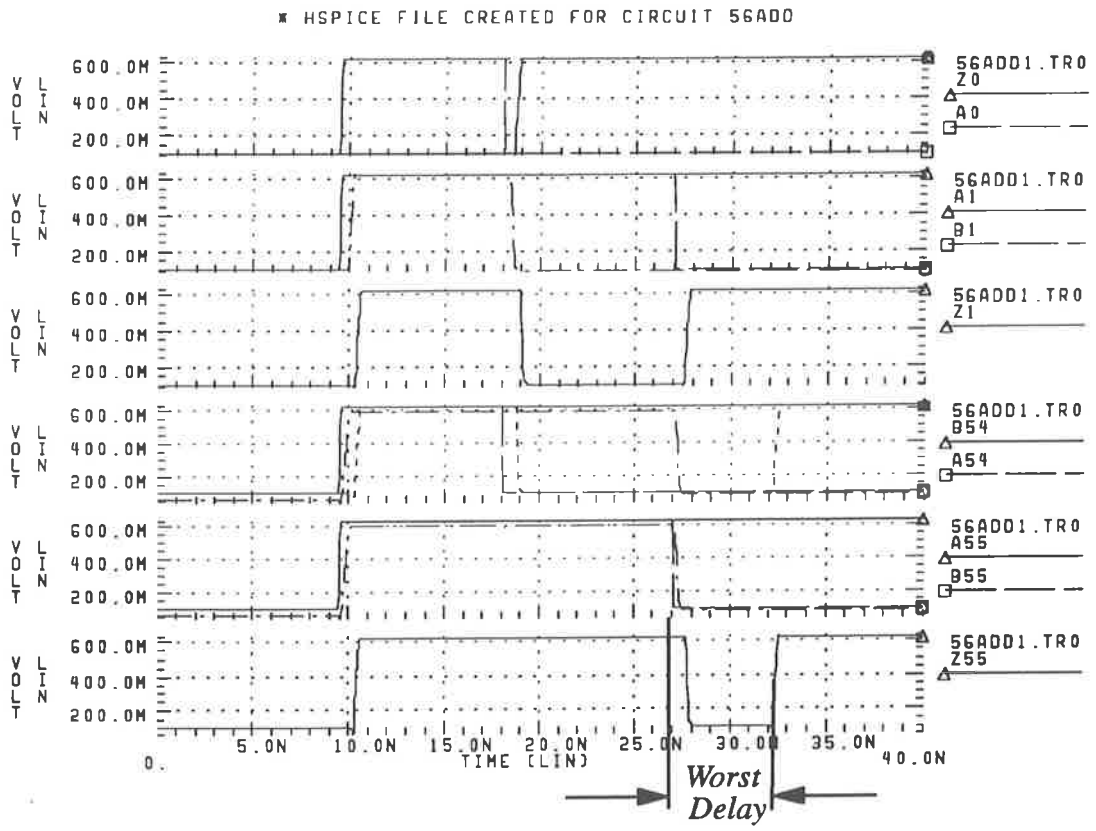


FIGURE 5-11. A 56-bit addition worst delay by HSPICE.

In this figure, Z_i is the ith sum of $A_i + B_i + C_{i-1}$. At the beginning, A and B are 0000... + 0000...; after the first $9ns$, A and B are changed to 1111... + 1111...; then after another $9ns$, A is changed into 1010... and B remains at 1111... At $t = 27ns$, B is set to 0101... and A remains 1010.... The average power consumed is approximately $400mW$. The worst delay is $5ns$ when there is a hazard. Obviously, the computation of the 56-bit adder is not fast enough, however, taking account into this chip area and placement

One method to speed up the addition operation is to reduce the transistor ratio in the whole floating-point adder/subtractor process. Then, the circuit will consume more

power, and the circuit reliability will weaken as described in the second chapter.

To ensure the circuit reliability and reduce power dissipation, a better way to increase addition speed is to improve and simplify the logic circuits as discussed in the preceding chapter.

5.4 Normalization

The realisation of the normalizer for a floating-point adder/subtractor is the third main task in the circuit design and implementation. There has not been much work published in more details on the design of normalizer before, especially in GaAs. We divide the normalizer into three parts: the priority detector, the encoder and the left shifter.

5.4.1 Priority Detector

The detector has been divided into seven 8-bit blocks as explained in the previous chapter. From Figure 4-30, the connections between two levels in the 8-bit block are very complex in implementation. Using a PLA methodology, we developed and completed the 8-bit detector. The floorplan consisted of four levels is shown in Figure 5-12. Figure 5-13 is the corresponding layout.

The level has been laid out for minimum area, but with regard to interconnection alignment, regularity and crosstalk. Thus, the 8-bit sub-cell only uses 166 transistors. The layout area occupies only $0.146 \times 0.260 mm^2$.

Considering temperature influence to PLA implementation, the output and the power dissipation of this 8-bit detector have been simulated and compared under three different conditions: 25, 75 and 125 degrees centigrade (DC). Analysing the PLA simulation results shown in Figure 5-14 (a) and (b), the delay and the power dissipation go high as temperature is increased, as opposite, the noise margin is decreased. Hence, a chip usually has good properties in low temperature condition. Regardless of any temperature, the detector in PLA is with reasonable properties.

The 56-bit detector was constructed vertically from 8-bit sub-cells. The connection between two sub-cells consists of a two-input NOR gate and a SBFL buffer, which drives the next stage with 9 fanouts (see Figure 4-31 and 4-32. The 56-bit

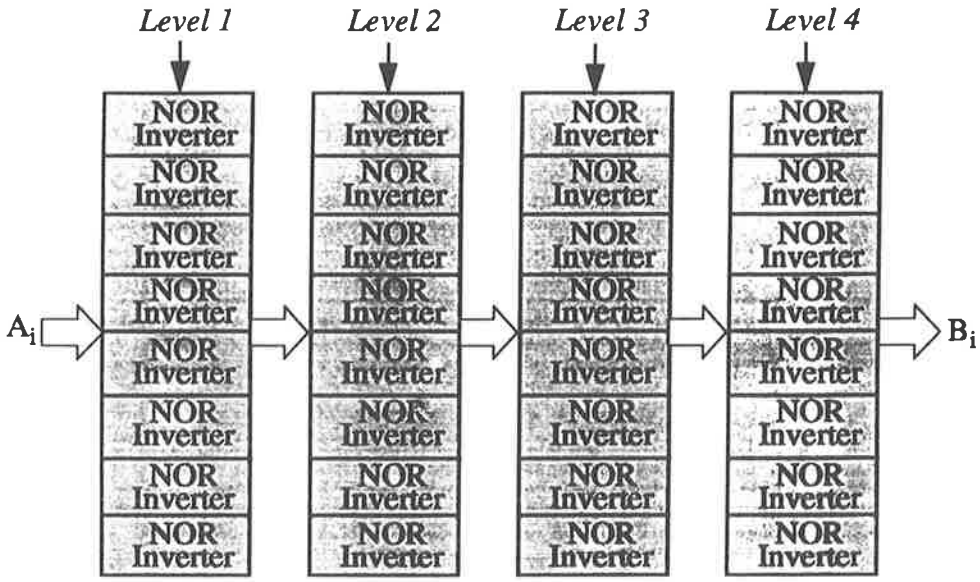


FIGURE 5-12. A 4-level 8-bit detector floorplan.

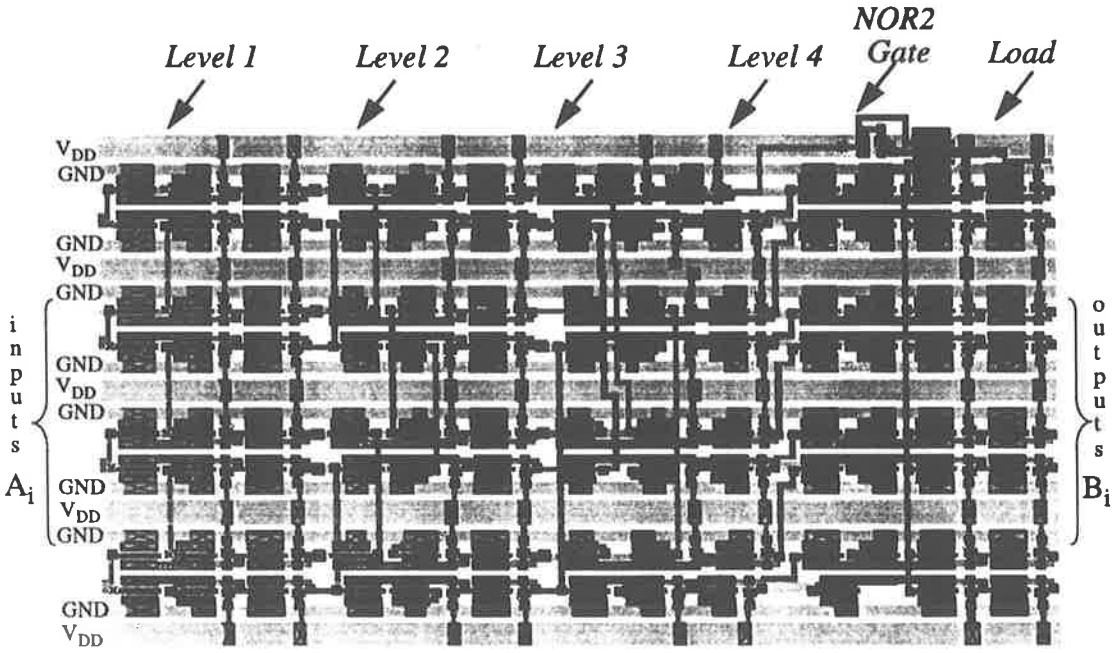
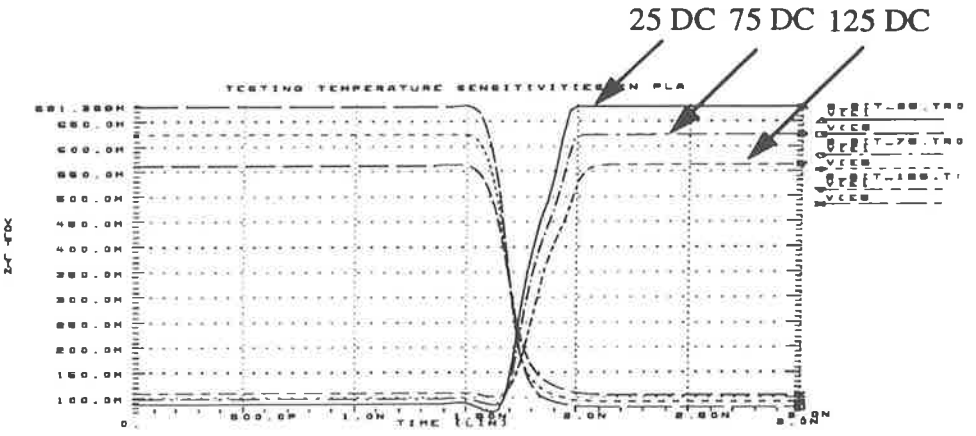
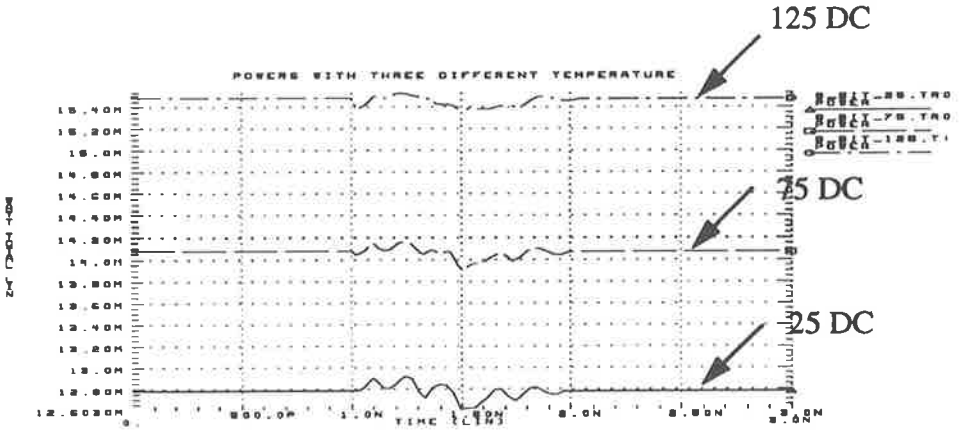


FIGURE 5-13. An 8-bit priority detector layout.



(a)



(b)

FIGURE 5-14. a) An output of the detector, and b) power dissipation under three different temperatures.

detector uses 1270 devices and occupies roughly an area of $1.00 \times 0.27 mm^2$.

Operation of the 56-bit priority detector was verified by the simulation as shown in Figure 5-15. The *'fracsum'* as the detector input A_i represents the result after adding two mantissas. The *'leadingonebit'* expresses the results B_i from the detector. Once the first leading 1 from the left is detected, all bits after the leading 1 bit will be set to 0. Figure 5-15 illustrates this fact and indicates when the worst delay happens in

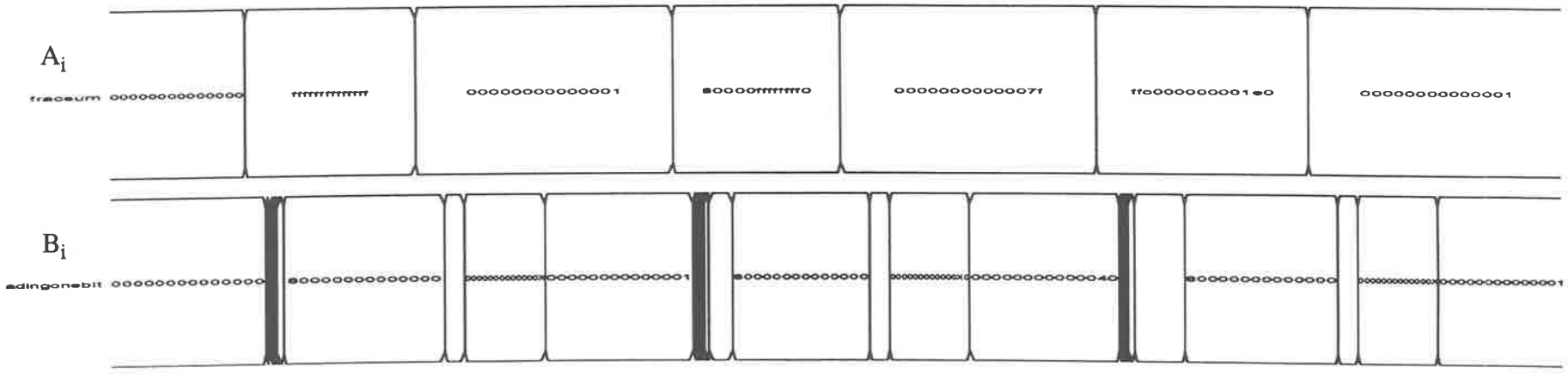


FIGURE 5-15. The 56-bit priority detector simulation results by IRSIM.

the detector. As each block consists of an 8-bit detector, the critical delay path occurs when there is a change from a state where the leftmost block has at least one 1 to another state where the rightmost block has at least one 1.

The **HSPICE** simulations in Figure 5-16, giving several special input $V(A_i$

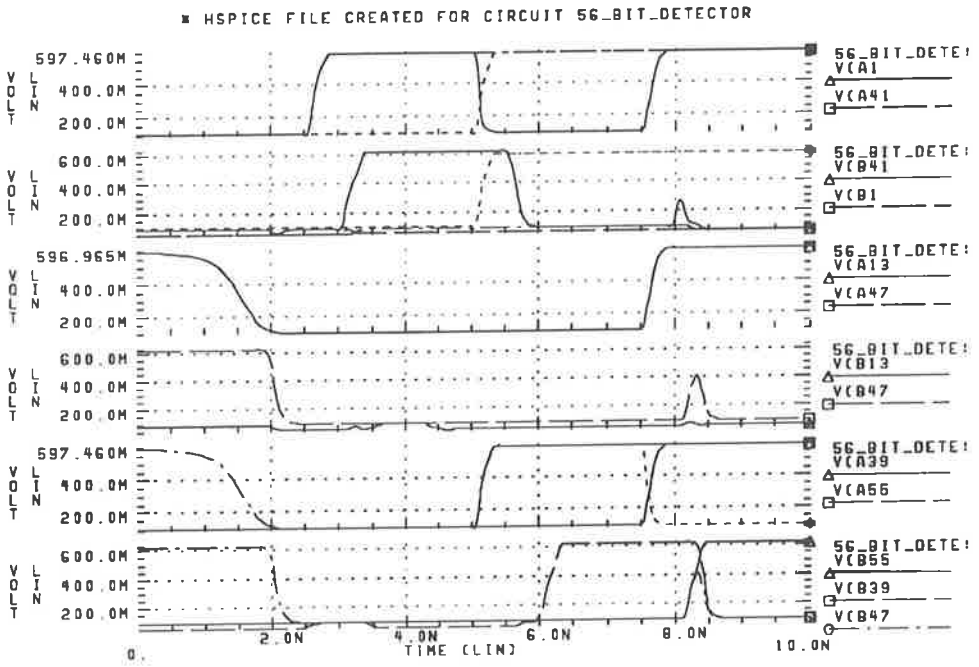


FIGURE 5-16. Experimental waveforms for the 56-bit detector.

and output $V(B_i$ examples, show that the worst overall delay of the 56-bit detector is $2.1ns$. The average power consumption is $0.14W$ when temperature is 75 degree centigrade. As the 56-bit detector is mainly built in DCFL, the power consumed is quite low, which, as we have emphasised many times before, is a primary performance criterion in VLSI design.

5.4.2 Encoder

Having detected the leading 1 bit of the mantissa result after adding the two mantissas, the next step is to convert it into a 6-bit control signal for the normalization

shifter in the physical layout. As described in Chapter 4, this was implemented as a PLA. The standard sub-cell layout of the 16-bit encoder is built separately using DCFL NOR4 and SDCFL NOR4, as shown in Figure 5-17 and Figure 5-18. Although

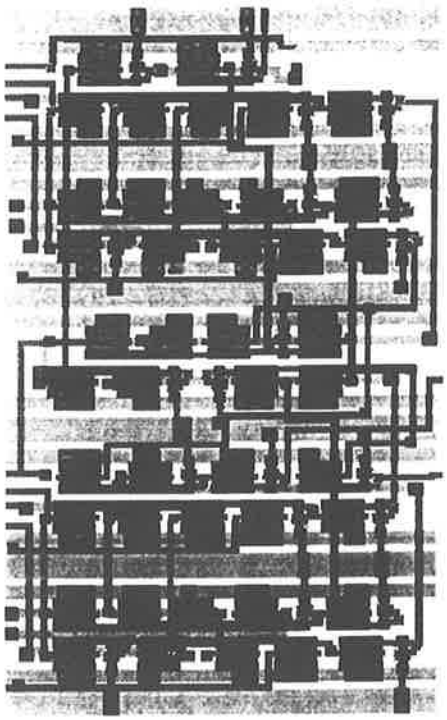


FIGURE 5-17. The layout of a 16-bit encoder PLA using DCFL.

these two components of the 16-bit encoders use 64 (in DCFL) and 78 (in SDCFL) devices, respectively, they occupy nearly the same area. Referring to Figure 4-36, the 56-bit encoder required can be connected by three 16-bit encoders and an 8-bit encoder.

The result of the functional simulation of the 16-bit encoder is shown in Figure 5-19. The *'leadingone'* represents the 16-bit encoder's input which comes from the priority detector output. The *'shiftselectN'* (N = 1, 2, 4, 8, 16) represents the encoder output, all of which are used as the control lines of the normalization shifter. Concurrently, the 6-bit encoder output must be subtracted from the exponent.

Assuming the *'leadingone'* 0000000000000010 as an example, in terms of the

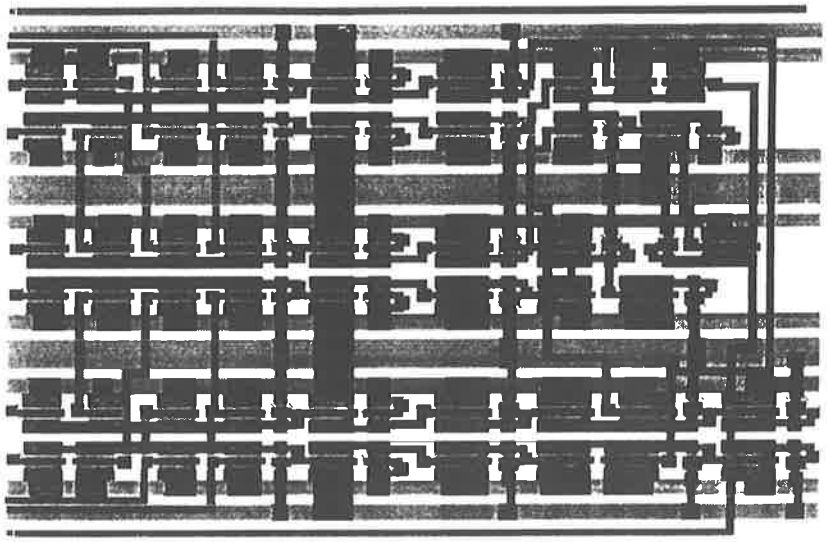


FIGURE 5-18. The 16-bit encoder PLA in SDCFL.

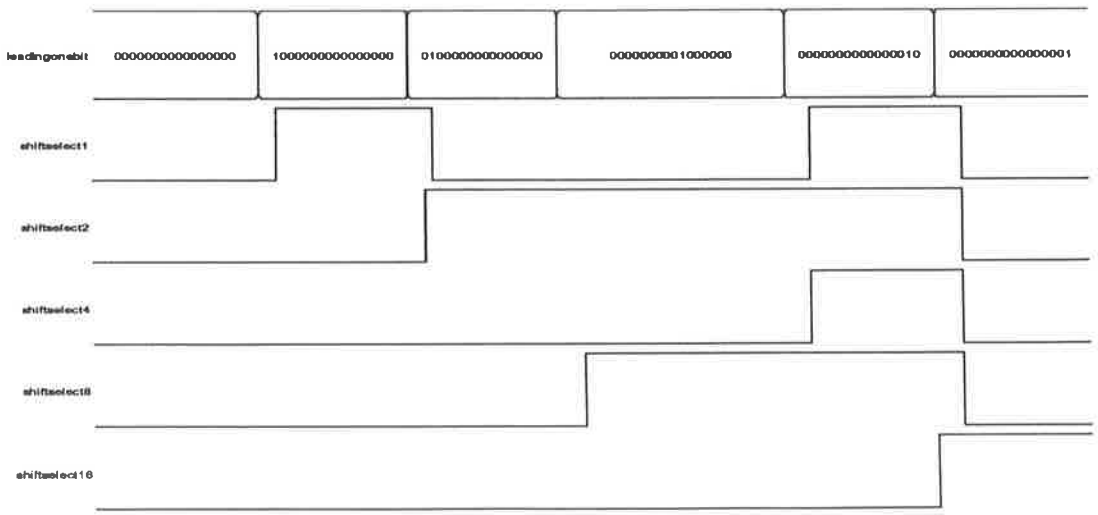


FIGURE 5-19. The verifications of the 16-bit encoder in SDCFL.

IEEE standard, the mantissa result needs shifting by 15 bits to make the hidden bit 1. The simulation result in **IRSIM** is shown that the states of *'shiftelect1'*, *'shiftelect2'*, *'shiftelect4'* and *'shiftelect8'* are all high, namely the left shifter will shift 15 bit from the right to the left to make the hidden bit 1.

Figure 5-20 shows the results of **HSPICE** simulation of the 16-bit encoder con-

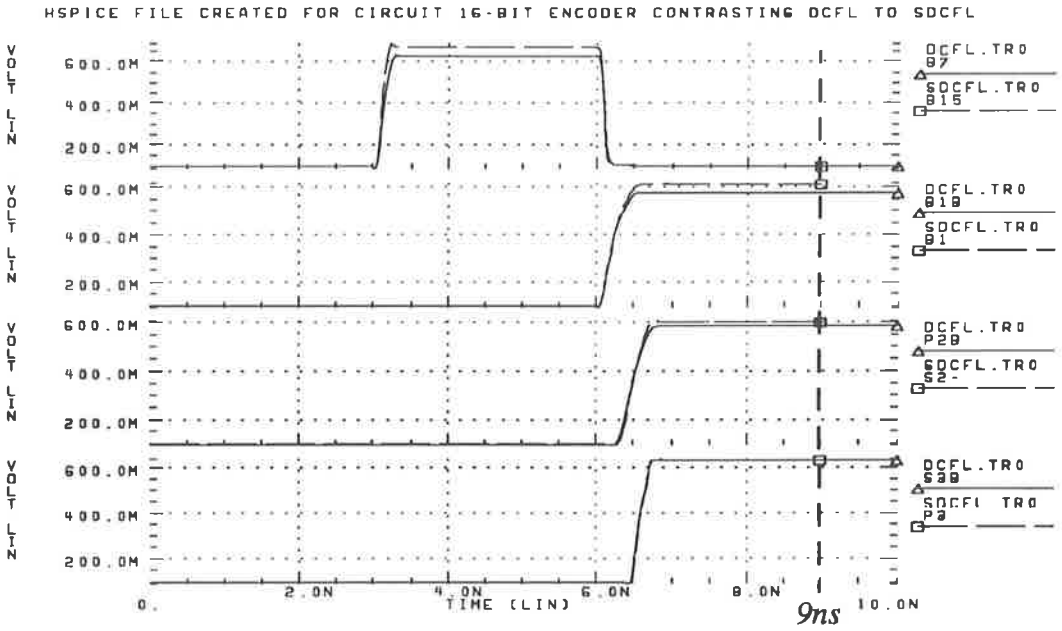


FIGURE 5-20. The 16-bit encoder performance comparisons between in DCFL and in SDCFL.

trasting DCFL to SDCFL. Both of the critical delay paths from input to output take roughly 440ps, which is measured when the state changes from 1000000000000000 to 0000000000000010, regarded as the critical condition, referring to **IRSIM** simulation results. The encoder simulation time in SDCFL is from $0ns$ to $9ns$, but simulation time in DCFL is from $0ns$ to $10ns$.

Figure 5-21 shows in detail the signals of Figure 5-20. The previous stage input of the first stage in Figure 5-17 and Figure 5-18 are the same, their output are the *B1b, B1, B7* and *B15* as the first stage input of the two encoders (B_{iDCFL} and B_{iSDCFL})

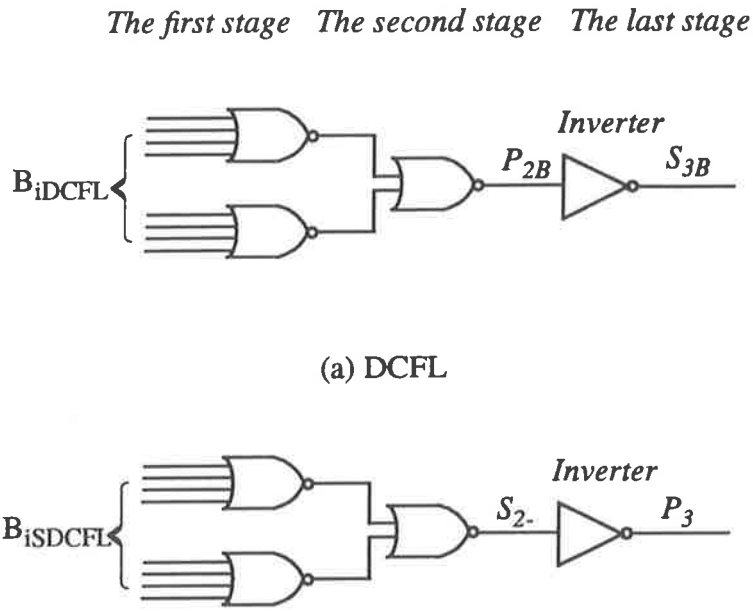


FIGURE 5-21. The related 4-fan-in DCFL and SDCFL logic circuits.

respectively. DCFL's high level input signals B_{iDCFL} are lower than B_{iSDCFL}, which means that Figure 5-21 (a) may be affected by noise in the first stage, even though the output of 4-fan-in DCFL is only slightly lower than SDCFL. Thus, once there is noise, the noise may influence the output of Figure 5-21 (a).

Assuming that a 4-fan-in DCFL NOR gate drives only one inverter as a load, the measured performances at the S_{3B} point (the output of the third stage) are similar to the SDCFL's performances at point P_3, including speed and the output low and high-voltage as shown in Figure 5-20 by the fourth waveform. Even so, the SDCFL encoder is still preferred because of better noise margin. Simulation results show that the SDCFL circuit needs much more power than DCFL circuit. However, it has a good noise immunity. It is worth mentioning again: the SDCFL implementation in

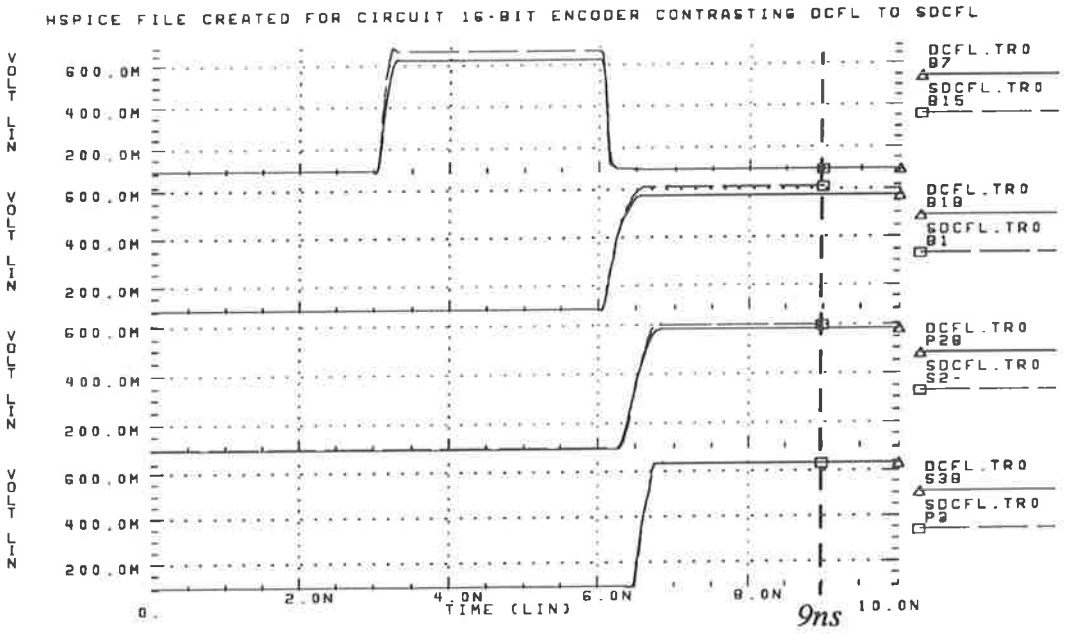


FIGURE 5-22. The 16-bit encoder performance comparisons between in DCFL and in SDCFL.

Figure 5-21 (b), can be optimised using the NOR-OR capabilities of SDCFL.

5.4.3 The left shifter

The left shifter function can be performed by the 56-bit barrel shifter, implementation of which is described in section **5.2**.

5.5 Exponent

The physical layout comprising 2000 devices has been built according to the floorplan of the exponent part in Figure 5-22. The area is $0.400 \times 1.700 mm^2$. Its width is just right for the alignment shifter (see Figure 5-1). The two 11-bit subtractors for comparison of the two exponents and for adjustment of the exponent employed the circuit of Figure 4-1 (b), and the 11-bit incrementer uses the circuit shown in Figure 4-9.

A one-bit sub-cell layout of the 3:1 MUX is illustrated in Figure 5-23. Eleven such sub-cells are required for selection of the output exponent. The two control lines

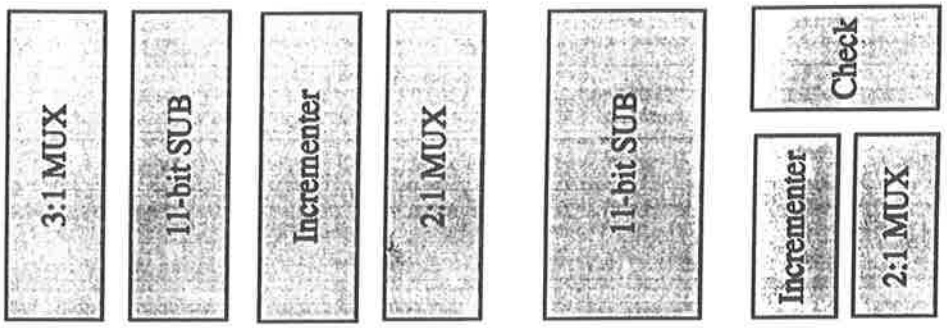


FIGURE 5-23. Exponent portion floorplan.

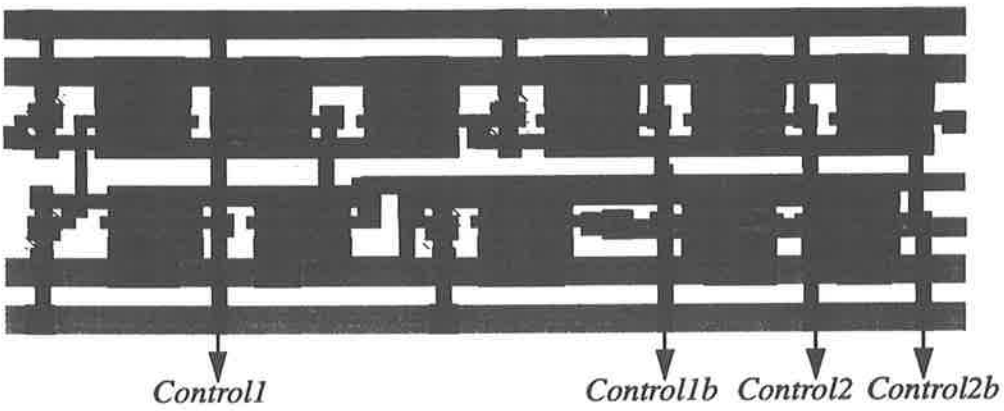


FIGURE 5-24. A 3:1 MUX layout for selecting the exponent output.

(*control1* and *control2*) from the leftmost two bits of the 56-bit mantissa addition and their inverted signals (*control1b* and *control2b*) determine the exponent output result. Its dimensions of the whole 11-bit 3:1 MUXs are $0.380 \times 0.120 mm$. The total delay time is $180 ps$.

5.6 The Evaluation of The Chip

The floating-point adder/subtractor is a static, 2V design with E/D MESFET DCFL, SDCFL and SBFL in the Vitesse HGaAs-2 process. The transistor ratio of DCFL is chosen to be 10 as discussed in chapter 2. The die size (excluding the input and output pads and registers but including datapath buses) is around 2 x 3 mm^2 using 15,500 transistors. The datapath speed illustrated in Figure 5-25 is estimated to be $20ns$ not

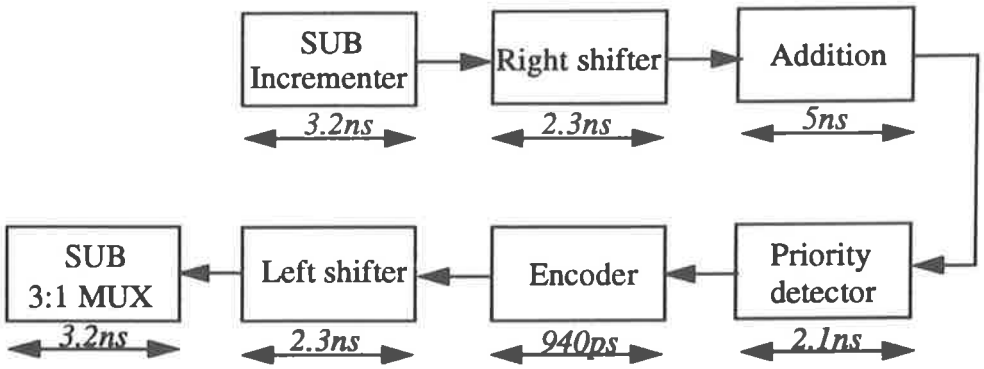


FIGURE 5-25. The estimation of the critical datapath for the floating-point adder/subtractor.

including rounding time and the input and output pads time delay but including inter-connection delay. The average power consumed is about 1.5W (excluding the output drivers). Table 5-1 summarises key figures of the adder/subtractor process.

Table 5-1. Summary of performance

Technology	0.8 Micron E/D MESFET
Basic gate topology	DCFL SDCFL SBFL
Whole process size	2 X 3 mm^2
Transistor count	15K
Power supply	+2V
Power dissipation	1.5W
Speed performance of operation	20ns

It is difficult to compare the performance of this adder/subtractor with other implementations. That is because performance is determined by many aspects such as minimum feature size, process, transistor size, noise immunity (can not measure), heat density, voltage supple value, algorithm, and improvement of process technique, whether to deal with the overflow or not and whether to deal with round off case or not. In terms of different requirements, a design can focus on different targets.

Milutinovic in 1986 reported a 32-bit integer adder performance characteristics of GaAs MESFET [Mil86$_1$], which used 2500 devices and consumed 1.2W power dissipation. Its speed performance was 2.9ns delay. As a comparison with our work, the 32-bit adder took slightly less time and used the same transistor counts but consumed 6 times less power. Considering the placement in our design, otherwise, our adder speed can be increased more. Yong in 1991 reported that a floating-point addition implemented in 1.0 micron CMOS technology [Yon91] in terms of IEEE standard required 2 clock cycles with a 60 MHz clock frequency.

In addition, unlike a floating-point multiplication and division, performance and detailed designs of a double precision floating-point addition/subtraction have not been extensively reported. As the design did not use reference material from other sources except the mantissa adder, it was more time consuming.

5.7 Summary

In this chapter, we have presented the layout of four major components of the adder/ subtractor: the barrel shifter, the 56-bit adder, the normalization priority detector and encoder and the exponent 3:1 MUX.

The factors that determined which of several alterative logic designs would be used were analysed and final layouts were presented. Functions and circuit simulation results from **IRSIM** and **HSPICE** were also presented. The logic functions in each part have been verified in **IRSIM** and their performances have been measured in **HSPICE**. The result is satisfactory and reasonable in each particular design under different simulation environments.

The bottleneck path should be chosen correctly to avoid inaccurate results. We can simulate different variable states in **IRSIM** and judge the worst case, then make

use of in **HSPICE** to measure the exact delay time. A typical adder example in the mantissa addition subsection has been given.

Since we are concerned here primarily with the investigation of trade-off between architecture and implementation technology, we have not proceeded beyond the design of individual functional units of the adder. Limited time and lack of funds for fabrication have meant that we have not proceeded to interconnect the units into a complete adder/subtractor. The next future work is to connect them, to simulate the function of all processes and to measure the properties of the floating-point adder/subtractor.

Finally the critical path through the adder/subtractor was identified and the worst case delay for the complete circuit was estimated to be $20ns$. The adder/subtractor counts of 15,500 transistor occupying a total area of $2 \times 3mm^2$ and consuming 1.5W.

Chapter 6 Conclusion

Abstract: *In this chapter, first we review the work, and then conclude the algorithm and architecture of the double precision floating-point adder/subtractor, and its relevant logic circuits and physical implementations. Some existing problems and improvements will be considered. At last, the improved and developed contributions in this work are listed.*

6.1 Overview

This thesis has described the design of the floating-point adder/subtractor processor which is a subsection of a Solid Modelling accelerator. It includes the algorithm, architecture, the related combinational *one-clock instruction execution* combinational logic design. Structured chip design and layout realisations with emphasis on GaAs technology were also presented. The floating-point adder/subtractor has generated a need for new design approaches that achieve optimal performance within this structure.

The design was based on IEEE 754 standard format introduced in **3.1.2.** Chapter 3 described the floating-point adder/subtractor architecture. Its practical logic circuit considerations, implementations and performance simulations were detailed in Chapter 4 and Chapter 5, respectively.

6.2 Summary and Conclusions

The most important property of GaAs is its speed advantage over silicon. As a result of the high speed requirement of the Solid Modelling accelerator, GaAs has been chosen in this design. The whole adder/subtractor process uses a power supply of 2V and is compatible with the predominantly used DCFL logic family. DCFL provides better noise margins and is less susceptible to capacitive load, making it useful for standard subcell based design. Its transistor ratio and dimensions at gate level have been a compromise of several factors: noise margin, time delay, power dissipation and area. At the same time, SDCFL and SBFL logic families were also used to address fan-in and fan-out limitations.

The algorithm for floating-point addition/subtraction is very complex. We have developed an algorithm and architecture for double precision floating-point adder/ subtractor for implementation in GaAs. The necessary serial steps for it are: compare the two exponents (used a substractor and a 6-bit incrementer), align the two mantissas in terms of the exponent difference (employed a barrel shifter), after adding the mantissas (used a hybrid adder of BCLA adder and carry select adder), normalize the mantissa result to be consistent with the IEEE 754 standard format (used a modified priority detector, an improved encoder and a barrel shifter). Finally, adjust the output

exponent (used a subtractor, an incrementer and a 11-bit 3:1 MUX).

High speed arithmetic is our design aim, however, other criteria including area, noise immunity and power dissipation for the whole process must be considered when designing the circuits. The fourth chapter discussed optimal and appropriate ideas and methods, which were realized in Chapter 5.

The delays in both gates and wires affect the speed of logic circuits. When speed is the objective, if the signals are to be transmitted over a long distance, the signals can be amplified by a large transistor as a driver to overcome the interconnect-delay limit. Furthermore, to produce high performance circuits, increasing the parallelism should be considered.

The floorplan in Figure 5-1 minimized interconnection delay and reduced design area. Some particular examples such as implementations of the barrel shifter, the priority detector and the encoder were given.

The use of CAD tools used for our VLSI design, were the simulators (**IRSIM**, and **HSPICE**) for checking logic function and measuring performance and layout generator (**MAGIC**).

The worst and overall delay of a 56-bit barrel shifter was simulated as approximately 2.3 nanoseconds and the average power dissipation is slightly less than 0.4W. The critical path of the 56-bit mantissa adder only takes about 5 nanoseconds. The whole adder/subtractor has a die dimensions of 2 by 3, using around 15,500 transistors. The speed performance of the whole adder/subtractor is estimated exceeding $20ns$ in a standard $0.8\mu m$ GaAs E/D MESFET process with an associated power dissipation of $1.5W$.

6.3 Discussions

If the discussion in Chapter 4 and Chapter 5 could be said to have some weak points, it is that the design consideration to normalize the mantissa could be pipelined for saving area in Chapter 4, and the mantissa adder could be divided in terms of 4+5+6+7+8+8+9+9 to reduce segment numbers for increasing speed. Due to limited time, connecting the units will be completed in the future. In addition, this design has

not used any OR gate in different logic families due to its low driving capability. Further, the more detailed analysis, investigation of other relevant work should be done more. However, comparison with CMOS and other GaAs addition/subtraction design is quite difficult due to many aspect reasons, all of which depend on the major goal of design (general purpose or specail purpose), depend on processing technology, depend on whether considering rounding stage or not, depend on cost and so forth.

6.4 Contributions

The principal contributions of this thesis are as follows:

- The critical comparison and selection of the GaAs transistor parameters and ratio.
- The modification of the double precision adder/subtractor algorithm and architecture.
- The improvement of the hybrid adder consisting of the BLCA and the carry select adders.
- The development of the encoder to omit delay and area of a 6-bit incrementer.
- The demonstration of physical implementation and realization of the barrel shifter.
- The applications of PLA methodology in the priority detector and the encoder with GaAs technology.
- Identifying of the bottleneck path for the mantissa adder.

References

[Bar92] **R. J. Barton,** "*Computer Architecture,*" Addison-Wesley Pub. Co., 1992.

[Bec88] **B. Becker,** "*Efficient Testing of Optimal Time Adders,*" IEEE Transactions on Computers, Vol. 37, No. 9, Sept. 1988.

[Bir90] **M. Birman, A. Samuels, G. Chu, T. Chuk, L. Hu, J. Mcleod, J. Barnes,** "*Developing the WTL3170/3171 Sparc Floating-Point Coprocessors,*" IEEE Micro, pp. 55-63, Feb. 1990.

[Bre82] **R. P. Brent, H. T. Kung,** "*A Regular Layout for Parallel Adders,*" IEEE Transactions on Computers, Vol. C-31, No. 3, pp. 260-264, March, 1982.

[Bur94] **N. Burgess**, lecture notes on computer architecture, Department of Electrical and Electronic Engineering, The University of Adelaide, 1995.

[Bus91] **E. Bushehri,** thesis title: "*Critical Design Issues For Gallium Arsenide VLSI Circuits,*" Department of Electrical and Electronic Engineering, University of South California 1991.

[Car93] **P. P. Carballo, R. Sarmiento, A. Nunez,** "*Integer and Control Units for A GaAs 32-Bit RISC Processor,*" Micropocessing and Microprogramming, Vol. 37, pp. 105-108, 1993.

[Cat90] **R. Cates,** "*Gallium Arsenide Finds a new niche,*" Spectrum, pp. 25-28, April, 1995.

[Cha95] **V. Chandramouli, N. Michell, K. F. Smith,** "*A New, Precharged, Low-Power Logic Family for GaAs Circuits,*" IEEE, JSSC, Vol. 30, No. 2, Feb. 1995.

[Chi88] **H. Chiyokura,** "*Solid Modelling with Designbase,*" Addison-Wesley pub. 1988.

[Chu94] **E. Chu,** thesis title: "*Characterisation & Optimization of Computational Functional Blocks for ATM Switches in GaAs MESFET Technology*", Department of Electrical and Electronic Engineering, The University of Adelaide, 1994.

[Cle85] **A. Clements,** "*The Principles of Computer Hardware,*" Oxford Science Pub., 1985.

[Dey95] **I. Deyhimy,** "*Gallium Arsenide Joints The Giants,*" Spectrum, pp. 33-40, Feb. 1995.

[Esh91] **K. Eshraghian, R. Sarmiento, P. P. Carballo, A. Nunez,** "*Speed-Area-Power Optimization for DCFL and SDCFL Class of Logic Using Ring Notation,*"Microprocessing and Microprogramming 32, pp. 75-82, 1991.

[Esh95] **K. Eshraghian,** "*Fundamentals of Ultra-High Speed Systems: GaAs VLSI Technology,*" To be published (1995), Prentice Hall.

[Fox86] **E. R. Fox, K. J. Kiefer, R. F. Vangen, S. P. Whalen,** "*Reduced Instruction Set Architecture For A GaAs Microprocessor System,*" Computer, pp. 71-81, Oct. 1986.

[Gom93] **L. Gomez, A. Hernandez, A. Nunez,** "*Timing Analysis For DCFL/SDCFL VLSI Circuits,*" Microprocessing and Microprogramming, Vol. 38 n 1-5, pp. 511-518, Sept. 1993.

[Goo89] **A. J. Goor,** "*Computer Architecture And Design,*" Addison-Wesley Publishing Company, 1989.

[Hel89] **W. Helbig, V. Milutinovic,** "*A DCFL E/D-MESFET GaAs Experimental RISC Machine,*" IEEE Transactions on Computers, pp. 263-274, Feb. 1989.

[Hen90] **J. L. Hennseey,** "*Computer Architecture A Quantitative Approach,*" Morgan Kaufmann Publishers, Inc., 1990.

[Her93] **A. Hernandez,** "*An Empirical Model to Estimate Power Consumption in GaAs DCFL/SDCFL Circuits,*" Microprocessing & Microprogramming, Vol. 37, pp. 23-26, 1993.

[Hil73] **F. J. Hill,** "*Digital Systems: Hardware Organization and Design,*" Welly Publishing Company, 1973.

[Hil78] **F. J. Hill,** "*Digital Systems: Hardware Organization and Design,*" John Wiley & Sons, Inc., 1978.

[Hok90] **E. Hokenke, R. K. Montoye,** "*Leading-Zero Anticipator (LZA) in The IBM*

RISC System/6000 Floating-Point Execution Unit," IBM J. Res. Develop. Vol. 34, No. 1, pp. 71-77, Jan. 1990.

[Hwa79] **K. Hwang,** "*Computer Arithmetic: Principles, Architecture and Design,*" John Wiley & Sons, Inc., 1979.

[Ide93] **N. Ide, H. Fukuhisa, Y. Kondo, T. Yoshida, M. Nagamatu, J. Mori, I. Yamazaki, K. Ueno,** "*A 320-MFLOPS CMOS Floating-Point Processing Unit for Superscalar Processors,*" IEEE JSSC, Vol. 28, No. 3, pp. 352-360, March, 1993.

[IEEE754] **ANSI/IEEE Standard 754-1985** *for Binary Floating-point Arithmetic*, IEEE Computer Society Press, Los Alamitos, Calif., 1985.

[Jii92] **H. Fujii, C. Hori, T. Takada, N. Hatanaka, T. Demura, G. Ootomo,** "*A Floating-Point Cell Library and a 100-MFLOPS Image Signal Processor,*" IEEE JSSC, Vol. 27, No. 7, pp. 1080-1087, July 1992.

[Kan93$_1$] **V. Kantabutra,** "*Designing Optimum One-Level Carry-Skip Adders,*" IEEE Transactions on Computer, Vol. 42, No. 6, June, 1993.

[Kan93$_2$] **V. Kantabutra,** "*Accelerated Two-Level Carry-Skip Adders--A Type of Very Fast Adders,*" IEEE Transactions on Computer, Vol. 42, No. 11, Nov., 1993.

[Kat90] **A. Katsuno, H. Takahashi, H. Kubosawa, T. Sato, A. Suga, G. Goto,** "*A 64-bit Floating-Point Processing Unit with a Horizontal Instruction Code for Parallel Operations,*" 1990 IEEE Int. Conf. on Comp. Design: VLSI in Comp. & Proc., pp. 347-350, 1990.

[Lar86] **L. E. Larson, J. F. Jensen, P. T. Greiling,** "*GaAs High-Speed Digital IC Technology: An Overview,*" Computer, pp. 21-27, Oct. 1986.

[Lew83] **M. H. Lewin,** "*Logic Design and Computer Organization,*" Addison-Wesley Pub. Co., 1983.

[Lin81] **H. Ling,** "*High Speed Binary Adder,*" IBM J. Res. Develop., Vol. 25, No. 3, pp. 156-166, May, 1981.

[Lon90] **S. L. Long,** "*Gallium Arsenide Digital Integrated Circuit Design,*" McGraw-

Hill, Inc. 1990.

[Lon92] **K. R. Nary, S. L. Long,** "*GaAs Tow-Phase Dynamic FET Logic: A Low Power Logic Family for VLSI,*" IEEE JSSC, Vol. 27, No. 10, pp. 1364-1371, Oct. 1992.

[Lu92] **M. Lu, J. S. Chiang,** "*A Novel Division Algorithm for The Residue Number System,*" IEEE Transaction on Computers, Vol. 41, No. 8, pp. 1026-1032, Aug. 1992.

[Lyn92] **T. Lynch, E. E. Swartzlander,** "*A Spanning Tree Carry Look-ahead Adder,*" IEEE Transactions on Computers, Vol. 41, No. 8, August, 1992.

[Man79] **M. M. Mano,** "*Digital Logic And Computer Design,*" Prentice-Hall, Inc. Englewood Cliffs, 1979.

[Man88] **M. M. Mano**, "*Computer Engineering: Hardware Design,*" Prentice-Hall International, 1988.

[McG90] **K. J. McGee,** "*Comments on In-Place Updating of Path Metrics in Viterbi Decoders,*" IEEE JSSC, Vol. 25, No. 4, August 1990.

[Mil86$_1$] **V. Milutinlovic,** "*An Introduction to GaAs Microprocessor Architecture for VLSI,*" IEEE Computer, Vol. 19, No. 3, pp. 30-42, March 1986.

[Mil86$_2$] **V. Milutinovic,** "*GaAs Microprocessor Technology,*" Computer, pp. 75-78, October 1986.

[Mil88] **V. Milutinovic,** "*Adder Design Analysis for a 32-Bit GaAs Microprocessor,*" Tutorial: "Gallium Arsenide Computer Design", The Institute of Electrical and Electronics Engineering, Inc., 1988.

[Mit92] **M. Mittal, C. A. T. Salama,** "*DPTL 4-b Carry Look-ahead Adder,*" IEEE, JSSC, Vol. 27, No. 11, pp. 1644-1647, Nov. 1992.

[Moo93] **B. Moore,** "*Memory Architecture for a Solid Modelling Accelerator in Unified GaAs/BiCMOS/CMOS Technology,*" Report, Department of Electrical and Electronic Engineering, The University of Adelaide, 1993.

[Mur90] **W. D. Murray,** "*Computer and Digital System Architecture,*" Prentice-Hall Co., 1990.

[Nga86] **T. F. Ngai,** "*Regular, Area-Time Efficient Carry Look-ahead Adders,*" Journal of Parallel and Distributed Computing 3, pp. 92-105, 1986.

[Okl85] **V. G. Oklobdzija, E. R. Barnes,** "*Some Optimal Schemes for ALU Implementation in VLSI Technology,*" Proceeding of Seventh Symposium on Computer Arithmetic, pp. 2-8, 1985.

[Pat94] **D. A. Patterson and J. L. Hennessy,** "*Computer Organization and Design: the Hardware/Software Interface,*" Morgan Kaufmann, 1994.

[Pas91] **J. H. Pastemak, ,** "GaAs MESFET Differential Pass-Transistor Logic," IEEE JSSC, Vol. 26, pp. 1309-1316, Sept. 1991.

[Qua92] **N. T. Quach, M. J. Flynn,** "*High-Speed Addition in CMOS,*" IEEE Transactions on Computers, Vol. 41, No. 12, Dec., 1992.

[Ras86] **T. L. Rasset, R. A Niederland, J. H. Lane, W. A. Geideman,** "*A 32-bit RISC Implemented in Enhancement-Mode JFET GaAs,*" IEEE Computer, pp. 60-68, October 1986.

[Req92] **A. A. G. Requicha, J.R. Rossignac,** "*Solid Modelling and Beyond,*" IEEE CG &A, Vol. 12, No. 5, pp. 31-44, Sept., 1992.

[Roc90] **M. Rocchi,** "*High-Speed Digital IC Technologies,*" Artech House, 1990.

[Scl88] **N. Sclater,** "*Gallium Arsenide IC Technology,*" TAB Books, Inc., 1988.

[Skl60] **J. Sklansky,** "*Conditional-Sum Addition Logic,*" IRE Trans. Electron. Computer, EC-9:226-231, 1960.

[Sri92] **H. R. Srinivas, K.K Parhi,** "*A Fast VLSI Adder Architecture,*" IEEE JSSC, Vol. 27, No 5, May 1992.

[Sta93] **W. Stallings,** "*Computer Organization and Architecture: Principles of Structure and Function,*" Macmillan Publishing Company, 1993.

[Sto80] **H. S. Stone,** "*Introduction to Computer Architecture,*" Science Research Associates, 1980.

[Tan90] **A. S. Tanenbaum,** "*Structured Computer Organization,*" Prentice Hall, 1990.

[Tie94] **J. A. Tierno, A.J. Martin, D. Borkovic, Tak Kwan Lee,** "*A 100-MIPS GaAs Asynchronous Microprocessor,*" IEEE Design & Test of Computers, Summer, 1994.

[Tie94] **J. Tierno,** "*A 100-MIPS GaAs Asynchronous Microprocessor,*" IEEE Design & Test of Computers, pp. 43-49, summer, 1994.

[Tya90] **A. Tyagi,** "*A Reduce Area Scheme for Carry-Select Adders,*" Processings of 1990 IEEE International Conf. on Computer Design, pp. 255-258, 1990.

[Ued85] **K. Ueda, H. Kitazawa, I. Harada,** "*CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design,*" IEEE Transactions on Computer-Aided Design, Vol. CAD-4, No. 1, Jan. 1985.

[Unw93] **I. H. Unwala,** "*Superpipelined Adder Designs,*" 1993 IEEE International Symposium on Circuits and Systems, Part 3, pp. 1841-1844, 1993.

[War82] **F. A. Ware, W.H. McAllister, J.R. Carlson, D.K. Sun,** "*64 Bit Monolithic Floating-Point Processors,*" IEEE JSSC, Vol. SC-17, No. 5, pp. 898-907, October 1982.

[Was82] **S. Waser,** "*Introduction to Arithmetic for Digital Systems Designers,*" Holt, Rinehart & Winston, 1982.

[Wes93] **N. H. Weste,** "*Principles Of CMOS VLSI Design,*" Addison-Wesley Publishing Company, 1993.

[Win90] **O. Wing,** "*Gallium Arsenide Digital Circuits,*" Kluwer Academic Publishers, 1990.

[Yon91] **S. L. Yong,** " *IEEE Standard Floating-Point ALU With 60MHz Clock Frequency,*" Journal of Korean Institute of Telematics and Electronics, Vol. 28A, No. 11, pp. 61-68, Nov. 1991.

[ZEN95] **ZEN Newsletter,** *Electronic Engineering Times*, July 31, 1995.

[Zyn88] **G. B. Zyner,** "*Design of Arithmetic Systems in VLSI,*" PhD Thesis, Department of Electrical and Electronics Engineering, The University of Adelaide, 1988.