CHAPTER 1
INTRODUCTION

1.1 VLSI DOMAIN

Very-large-scale integration (VLSI) is the process of creating an integrated circuit (IC) by combining hundreds of thousands of transistors or devices into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI lets IC designers add all of these into one chip.

1.1.1 HISTORY

The history of the transistor dates to the 1920s when several inventors attempted devices that were intended to control current in solid-state diodes and convert them into triodes. Success came after World War II, when the use of silicon and germanium crystals as radar detectors led to improvements in fabrication and theory. Scientists who had worked on radar returned to solid-state device development. With the invention of transistors at Bell Labs in 1947, the field of electronics shifted from vacuum tubes to solid-state device.

With the small transistor at their hands, electrical engineers of the 1950s saw the possibilities of constructing far more advanced circuits. However, as the complexity of circuits grew, problems arose.

One problem was the size of the circuit. A complex circuit like a computer was dependent on speed. If the components were large, the wires interconnecting them must be long. The electric signals took time to go through the circuit, thus slowing the computer.

The invention of the integrated circuit by Jack Kilby and Robert Noyce solved this problem by making all the components and the chip out of the same block (monolith) of semiconductor material. The circuits could be made smaller, and the manufacturing process could be automated. This led to the idea of integrating all components on a single-crystal silicon wafer, which led to small-scale integration (SSI) in the early 1960s, medium-scale integration (MSI) in the late 1960s, and then large-scale integration (LSI) as well as VLSI in the 1970s and 1980s, with tens of thousands of transistors on a single chip (later hundreds of thousands, then millions, and now billions (109)).

The first semiconductor chips held two transistors each. Subsequent advances added more transistors, and as a consequence, more individual functions or systems were integrated over time. The first integrated circuits held only a few devices, perhaps as many as ten diodes, transistors, resistors and capacitors, making it possible to fabricate one or more logic gates on a single device. Now known retrospectively as small-scale integration (SSI), improvements in technique led to devices with hundreds of logic gates, known as medium-scale integration (MSI). Further improvements led to large-scale integration (LSI), i.e. systems with at least a thousand logic gates. Current technology has moved far past this mark and today's microprocessors have many millions of gates and billions of individual transistors.

At one time, there was an effort to name and calibrate various levels of large-scale integration above VLSI. Terms like ultra-large-scale integration (ULSI) were used. But the huge number of gates and transistors available on common devices has rendered such fine distinctions moot. Terms suggesting greater than VLSI levels of integration are no longer in widespread use.

In 2008, billion-transistor processors became commercially available. This became more commonplace as semiconductor fabrication advanced from the then-current generation of 65 nm processes. Current designs, unlike the earliest devices, use extensive design automation and automated logic synthesis to lay out the transistors, enabling higher levels of complexity in the resulting logic functionality. Certain high-performance logic blocks like the SRAM (static random-access memory) cell, are still designed by hand to ensure the highest efficiency.

1.1.2 STRUCTURED DESIGN

Structured VLSI design is a modular methodology originated by Carver Mead and Lynn Conway for saving microchip area by minimizing the interconnect fabrics area. This is obtained by repetitive arrangement of rectangular macro blocks which can be interconnected using wiring by abutment. An example is partitioning the layout of an adder into a row of equal bit slices cells. In complex designs this structuring may be achieved by hierarchical nesting.

Structured VLSI design had been popular in the early 1980s, but lost its popularity later because of the advent of placement and routing tools wasting a lot of area by routing, which is tolerated because of the progress of Moore's Law. When introducing the hardware description language KARL in the mid' 1970s, Reiner Hartenstein coined the term "structured VLSI design" (originally as "structured LSI design"), echoing Edsger Dijkstra's structured programming approach by procedure nesting to avoid chaotic spaghetti-structured program.

1.1.3 DIFFICULTIES

As microprocessors become more complex due to technology scaling, microprocessor designers have encountered several challenges which force them to think beyond the design plane, and look ahead to post-silicon:

- **Process variation**

  As photolithography techniques get closer to the fundamental laws of optics, achieving high accuracy in doping concentrations and etched wires is becoming more difficult and prone to errors due to variation. Designers now must simulate across multiple fabrication process corners before a chip is certified ready for production, or use system-level techniques for dealing with effects of variation.

- **Stricter design rules**

  Due to lithography and etch issues with scaling, design rules for layout have become increasingly stringent. Designers must keep in mind an ever increasing list of rules when laying out custom circuits. The overhead for custom design is now reaching a tipping point, with many design houses opting to switch to electronic design automation (EDA) tools to automate their design process.

- **Timing/design closure**

As clock frequencies tend to scale up, designers are finding it more difficult to distribute and maintain low clock skew between these high frequency clocks across the entire chip. This has led to a rising interest in multi core and multiprocessor architectures, since an overall speedup can be obtained even with lower clock frequency by using the computational power of all the cores.

- **First-pass success**

As die sizes shrink (due to scaling), and wafer sizes go up (due to lower manufacturing costs), the number of dies per wafer increases, and the complexity of making suitable photo masks goes up rapidly. A mask set for a modern technology can cost several million dollars. This non-recurring expense deters the old iterative philosophy involving several "spin-cycles" to find errors in silicon, and encourages first-pass silicon success. Several design philosophies have been developed to aid this new design flow, including design for manufacturing (DFM), design for test (DFT).

1.1.4 VLSI Design Flow

The VLSI IC circuits design flow is shown in the Fig(1) below. The various levels of design are numbered and the blocks show processes in the design flow.Specifications comes first, they describe abstractly, the functionality, interface, and the architecture of the digital IC circuit to be designed.

Behavioural description is then created to analyse the design in terms of functionality, performance, compliance to given standards, and other specifications.

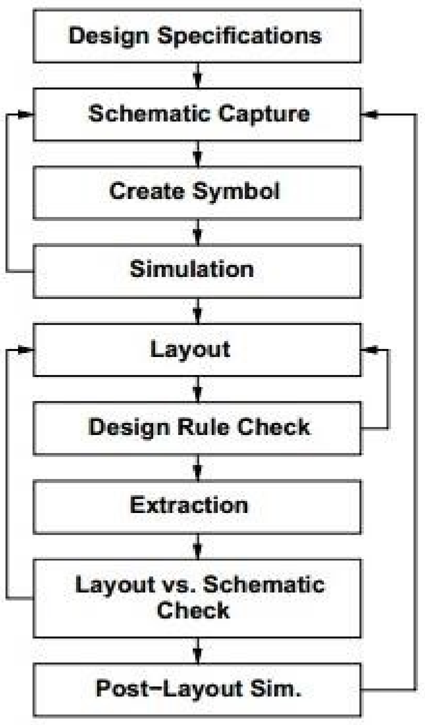


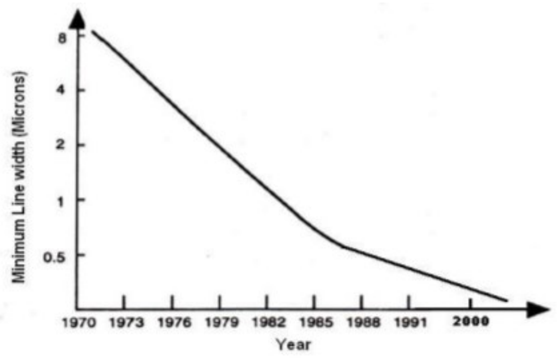
Fig 1 Design Flow

RTL description is done using HDLs. This RTL description is simulated to test functionality. From here onwards we need the help of EDA tools.RTL description is then converted to a gate-level net list using logic synthesis tools. A gatelevel netlist is a description of the circuit in terms of gates and connections between them, which are made in such a way that they meet the timing, power and area specifications.

Finally, a physical layout is made, which will be verified and then sent to fabrication.

1.2 METAL-OXIDE-SEMICONDUCTOR (MOS) AND RELATED VLSI TECHNOLOGY

The MOS technology is considered as one of the very important and promising technologies in the VLSI design process. The circuit designs are realized based on pMOS, nMOS, CMOS and BiCMOS devices. The pMOS devices are based on the p-channel MOS transistors. Specifically, the pMOS channel is part of a n-type substrate lying between two heavily doped p+ wells beneath the source and drain electrodes. Generally speaking, a pMOS transistor is only constructed in consort with an nMOS transistor. The nMOS technology and design processes provide an excellent background for other technologies. In particular, some familiarity with nMOS allows a relatively easy transition to CMOS technology and design. The techniques employed in nMOS technology for logic design are similar to GaAs technology. Therefore, understanding the basics of nMOS design will help in the layout of GaAs circuits. In addition to VLSI technology, the VLSI design processes also provides a new degree of freedom for designers which helps for the significant developments. With the rapid advances in technology the size of the ICs is shrinking and the integration density is increasing.



The graph shows a significant decrease in the size of the chip in recent years which implicitly indicates the advancements in the VLSI technology.

1.2.1 BASIC MOS TRANSISTORS

The MOS Transistor means, Metal-Oxide-Semiconductor Field Effect Transistor which is the most basic element in the design of a large scale integrated circuits(IC). These transistors are formed as a ``sandwich" consisting of a semiconductor layer, usually a slice, or wafer, from a single crystal of silicon; a layer of silicon dioxide (the oxide) and a layer of metal. These layers are patterned in a manner which permits transistors to be formed in the semiconductor material (the ``substrate"); a diagram showing a MOSFET is shown below.

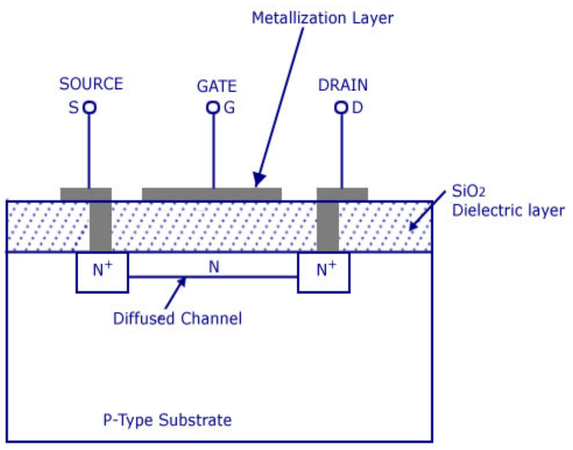


Fig 2 MOSFET

Silicon dioxide is a very good insulator, so a very thin layer, typically only a few hundred molecules thick, is used.In fact , the transistors which are used do not use metal for their gate regions, but instead use polycrystalline silicon (poly). Polysilicon gate FET's have replaced virtually all of the older devices using metal gates in large scale integrated circuits. (Both metal and polysilicon FET's are sometimes referred to as IGFET's (insulated gate field effect transistors), since the silicon dioxide under the gate is an insulator.

MOS Transistors are classified as n-MOS, p-MOS and c-MOS Transistors based on the fabrication. nMOS devices are formed in a p-type substrate of moderate doping level. The source and drain regions are formed by diffusing n- type impurities through suitable masks into these areas to give the desired n-impurity concentration and give rise to depletion regions which extend mainly in the more lightly doped p-region . Thus, source and drain are isolated from one another by two diodes. Connections to the source and drain are made by a deposited metal layer. In order to make a useful device, there must be the capability for establishing and controlling a current between source and drain, and .this is commonly achieved in one of two ways, giving rise to the enhancement mode and depletion mode transistors.

1.2.2 PMOS logic

P-type metal-oxide-semiconductor logic uses p-channel metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits. PMOS transistors operate by creating an inversion layer in an n-type transistor body. This inversion layer, called the p-channel, can conduct holes between p-type "source" and "drain" terminals.

The p-channel is created by applying voltage to the third terminal, called the gate. Like other MOSFETs, PMOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation.

While PMOS logic is easy to design and manufacture (a MOSFET can be made to operate as a resistor, so the whole circuit can be made with PMOS FETs), it has several shortcomings as well. The worst problem is that there is a direct current (DC) through a PMOS logic gate when the PUN is active, that is, whenever the output is high, which leads to static power dissipation even when the circuit sits idle.

Also, PMOS circuits are slow to transition from high to low. When transitioning from low to high, the transistors provide low resistance, and the capacitive charge at the output accumulates very quickly (similar to charging a capacitor through a very low resistance). But the resistance between the output and the negative supply rail is much greater, so the high-to-low transition takes longer (similar to discharge of a capacitor through a high resistance). Using a resistor of lower value will speed up the process but also increases static power dissipation.

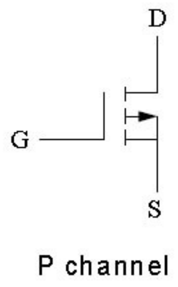


Fig 3 PMOS

1.2.3 NMOS logic

N-type metal-oxide-semiconductor uses n-type field-effect transistors (MOSFETs) to implement logic gates and other digital circuits. These nMOS transistors operate by creating an inversion layer in a p-type transistor body. This inversion layer, called the n-channel, can conduct electrons between n-type "source" and "drain" terminals. The n-channel is created by applying voltage to the third terminal, called the gate. Like other MOSFETs, nMOS transistors have four modes of operation: cut-off (or subthreshold), triode, saturation (sometimes called active), and velocity saturation.

MOS stands for metal-oxide-semiconductor, reflecting the way MOS-transistors were originally constructed, predominantly before the 1970s, with gates of metal, typically aluminium. Since around 1970, however, most MOS circuits have used self-aligned gates made of polycrystalline silicon. These silicon gates are still used in most types of MOSFET based integrated circuits, although metal gates (Al or Cu) started to reappear in the early 2000s for certain types of high speed circuits, such as high performance microprocessors.

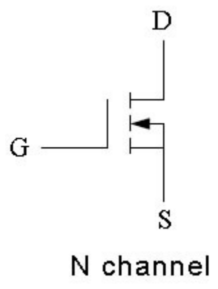


Fig 4 NMOS

1.2.4 CMOS

Complementary metal–oxide–semiconductor (CMOS) is a technology for constructing integrated circuits. CMOS technology is used in microprocessors, microcontrollers, static RAM, and other digital logic circuits. CMOS technology is also used for several analog circuits such as image sensors (CMOS sensor), data converters, and highly integrated transceivers for many types of communication. Frank Wanlass patented CMOS in 1963 (US patent 3,356,858) while working for Fairchild Semiconductor.

CMOS is also sometimes referred to as complementary-symmetry metal–oxide–semiconductor (COS-MOS). The words "complementary-symmetry" refer to the typical design style with CMOS using complementary and symmetrical pairs of p-type and n-type metal oxide semiconductor field effect transistors (MOSFETs) for logic functions.

Two important characteristics of CMOS devices are high noise immunity and low static power consumption. Since one transistor of the pair is always off, the series combination draws significant power only momentarily during switching between on and off states. Consequently, CMOS devices do not produce as much waste heat as other forms of logic, for example transistor–transistor logic (TTL) or N-type metal-oxide-semiconductor logic (NMOS) logic, which normally have some standing current even when not changing state. CMOS also allows a high density of logic functions on a chip. It was primarily for this reason that CMOS became the most used technology to be implemented in very-large-scale integration (VLSI) chips.

The phrase "metal–oxide–semiconductor" is a reference to the physical structure of certain field-effect transistors, having a metal gate electrode placed on top of an oxide insulator, which in turn is on top of a semiconductor material. Aluminium was once used but now the material is poly silicon. Other metal gates have made a comeback with the advent of high-κ dielectric materials in the CMOS process, as announced by IBM and Intel for the 45 nano meter node and smaller sizes.

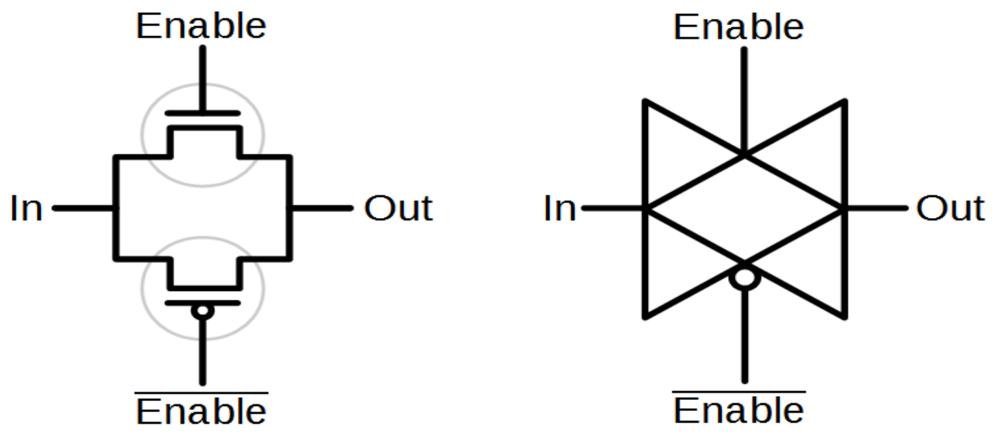


Fig 5 CMOS

1.2.5 BICMOS

BiCMOS is an evolved semiconductor technology that integrates two formerly separate semiconductor technologies, those of the bipolar junction transistor and the CMOS transistor, in a single integrated circuit device.

Bipolar junction transistors offer high speed, high gain, and low output resistance, which are excellent properties for high-frequency analog amplifiers, whereas CMOS technology offers high input resistance and is excellent for constructing simple, low-power logic gates. For as long as the two types of transistors have existed in production, designers of circuits utilizing discrete components have realized the advantages of integrating the two technologies; however, lacking implementation in integrated circuits, the application of this free-form design was restricted to fairly simple circuits. Discrete circuits of hundreds or thousands of transistors quickly expand to occupy hundreds or thousands of square centimetres of circuit board area, and for very high-speed circuits such as those used in modern digital computers, the distance between transistors (and the minimum capacitance of the connections between them) also makes the desired speeds grossly unattainable, so that if these designs cannot be built as integrated circuits, then they simply cannot be built.

In the 1990s,modern integrated circuit fabrication technologies began to make BiCMOS a reality. This technology rapidly found application in amplifiers and analog power management circuits, and has some advantages in digital logic. BiCMOS circuits use the characteristics of each type of transistor most appropriately. Generally this means that high current circuits use metal–oxide–semiconductor field-effect transistor (MOSFETs) for efficient control, and portions of specialized very high performance circuits use bipolar devices. Examples of this include radio frequency (RF) oscillators, bandgap-based references and low-noise circuits.

The Pentium, Pentium Pro, and Super SPARC microprocessors also used BiCMOS.

1.2.6 Comparison of BiCMOS and CMOS technologies

The BiCMOS gates perform in the same manner as the CMOS inverter in terms of power consumption, because both gates display almost no static power consumption. When comparing BiCMOS and CMOS in driving small capacitive loads, their performance are comparable, however, making BiCMOS consume more power than CMOS. On the other hand, driving larger capacitive loads makes BiCMOS in the advantage of consuming less power than CMOS, because the construction of CMOS inverter chains are needed to drive large capacitance loads, which is not needed in BiCMOS. The BiCMOS inverter exhibits a substantial speed advantage over CMOS inverters, especially when driving large capacitive loads. This is due to the bipolar transistor's capability of effectively multiplying its current. For very low capacitive loads, the CMOS gate is faster than its BiCMOS counterpart due to small values of Cint. This makes BiCMOS ineffective when it comes to the implementation of internal gates for logic structures such as ALUs, where associated load capacitances are small. BiCMOS devices have speed degradation in the low supply voltage region and also BiCMOS is having greater manufacturing complexity than CMOS.

1.2.7 Y Chart

The Gajski-Kuhn Y-chart is a model, which captures the considerations in designing semiconductor devices. The three domains of the Gajski-Kuhn Y-chart are on radial axes. Each of the domains can be divided into levels of abstraction, using concentric rings. At the top level (outer ring), we consider the architecture of the chip; at the lower levels (inner rings), we successively refine the design into finer detailed implementation−Creating a structural description from a behavioural one is achieved through the processes of high-level synthesis or logical synthesis. Creating a physical description from a structural one is achieved through layout synthesis.

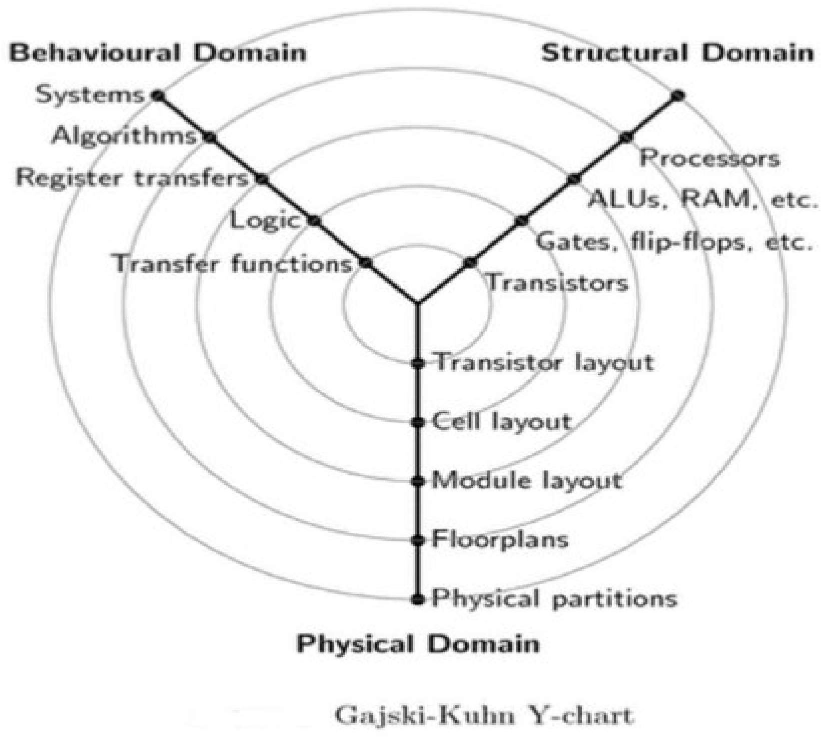


Fig 6 Y Chart

1.3 MOST OF TODAY'S VLSI DESIGN CATEGORIES

1.Analog:

Small transistor count precision circuits such as Amplifiers, Data converters, filters, Phase Locked Loops, Sensors etc.

2.ASICS or Application Specific Integrated Circuits:

Progress in the fabrication of IC's has enabled us to create fast and powerful circuits in smaller and smaller devices. This also means that we can pack a lot more of functionality into the same area. The biggest applicationof this ability is found in the design of ASIC's. Theseare IC's that are created for specific purposes - each device is created to do a particular job, and do it well. The most common application area for this is DSP - signal filters, image compression, etc. To go to extremes, consider the fact that the digital wristwatch normally consists of a single IC doing all the time-keeping jobs as well as extra features like games, calendar, etc.

3.SoC or Systems on a chip:

These are highly complex mixed signal circuits (digital and analog all on the same chip). A network processor chip or a wireless radio chip is an example of an SoC.

1.4 ARITHMETIC LOGIC UNIT

An arithmetic logic unit (ALU) is a digital circuit that performs arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit (CPU) of a computer, and even the simplest microprocessors contain one for purposes such as maintaining timers. Most ALUs can perform the following operations:

- Bitwise logic operations
    - AND
    - OR
    - NOT
    - XOR
    - XNOR
    - NAND
    - NOR
- Integer arithmetic operations
    - Addition
    - Subtraction
    - Multiplication
    - Division
- Bit-shifting operations
    - Right Shift
    - Left Shift
    - Rotate Right
    - Rotate Left

The processors found inside modern CPUs and graphics processing units (GPUs) accommodate every powerful and very complex ALUs; a single component may contain a number of ALUs. Mathematician John von Neumann proposed the ALU concept in 1945, when he wrote a report on the foundations for a new computer called the EDVAC. Research into ALUs remains an important part of computer science, falling under Arithmetic and logic structures in the ACM Computing Classification System.

CHAPTER 2
LITERATURE SERVEY

2.1 ADDER

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

There are several types of adders used in the present day technologies, some of them include : Carry Save Adder(CSA), Carry Look ahead Adder(CLA), Ripple Carry Adder(RCA), etc.,

Adders are designed using Half Adder or the Full Adder.

2.1.1 Half Adder

The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition. The value of the sum is 2C + S. The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. The Boolean logic for the sum (in this case S) will be $A'B + AB'$ whereas for the carry (C) will be AB. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder.[1] The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder. The input variables of a half adder are called the augend and addend bits. The output variables are the sum and carry.

2.1.2 Full Adder

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full-adder adds three one-bit numbers, often written as A, B, and Cin; A and B are the operands, and Cin is a bit carried in from the previous less-significant stage.[2] The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc.

bit binary numbers. The circuit produces a two-bit output. Output carry and sum typically represented by the signals Cout and S, where the sum equals 2Cout + S.

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with $S = A \oplus B \oplus Cin$ and $Cout = (A \cdot B) + (Cin \cdot (A \oplus B))$.

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple integrated circuit chips which contain only one gate type per chip.

A full adder can also be constructed from two half adders by connecting A and B to the input of one half adder, then taking its sum-output S as one of the inputs to the second half adder and Cin as its other input, and finally the carry outputs from the two half-adders are connected to an OR gate. The sum-output from the second half adder is the final sum output (S) of the full adder and the output from the OR gate is the final carry output (Cout).

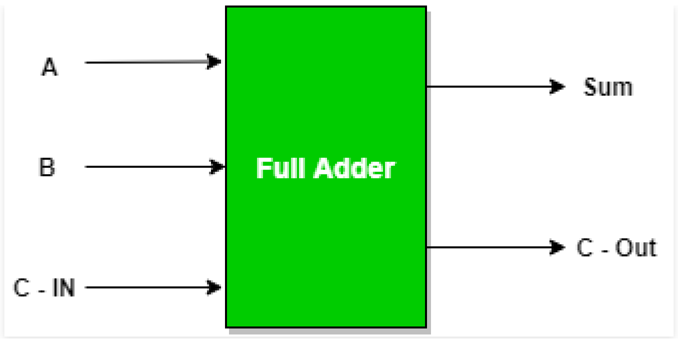


Fig 7 Basic Full Adder

INPUTS			OUTPUTS	
A	**B**	**CIN**	**SUM**	**COUT**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1 Truth Table of Full Adder

Logical Expression for SUM:

= A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN

= C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’)

= C-IN XOR (A XOR B)

Logical Expression for C-OUT:

= A’ B C-IN + A B’ C-IN + A B C-IN’ + A B C-IN

= A B + B C-IN + A C-IN

2.2 SUBTRACTOR

In electronics, a subtractor can be designed using the same approach as that of an adder. The binary subtraction process is summarized below. As with an adder, in the general case of calculations on multi-bit numbers, three bits are involved in performing the subtraction for each bit of the difference: the minuend (X_i), subtrahend (Y_i), and a borrow in from the previous (less significant) bit order position (B_i). The outputs are the difference bit (D_i) and borrow bit B_{i+1}

2.2.1 Half Subtractor

The half subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, the minuend X and subtrahend Y and two outputs the difference D and borrow out B_{out}. The borrow out signal is set when the subtractor needs to borrow from the next digit in a multi-digit subtraction. That is B_{out} =1 when X<Y. Since X and Y are bits,

B_{out} =1 if and only if X=0 and Y=1. An important point worth mentioning is that the half subtractor diagram aside implements X-Y and not Y-X since B_{out} is given by

B_{out}= ~X.Y

This is an important distinction to make since subtraction itself is not commutative, but the difference bit D is calculated using an XOR gate which is commutative.

2.2.2 Full Subtractor

The full subtractor is a combinational circuit which is used to perform subtraction of three input bits: the minuend X, subtrahend Y, and borrow in B_{in}. The full subtractor generates two output bits: the difference D and borrow out B_{out}. B_{in} is set when the previous digit is borrowed from X. Thus, B_{in} is also subtracted from X as well as the subtrahend Y. Or in symbols $X-Y-B_{in}$. Like the half subtractor, the full subtractor generates a borrow out when it needs to borrow from the next digit. Since we are subtracting X by Y and B_{in}a borrow out needs to be generated when $X<Y+B_{in}$. When a borrow out is generated, 2 is added in the current digit. (This is similar to the subtraction algorithm in decimal. Instead of adding 2, we add 10 when we borrow.) Therefore, $D=X-Y-B_{in}+2B_{out}$

2.3 MULTIPLIER

A binary multiplier is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers. It is built using binary adders.

A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together. This process is similar to the method taught to primary schoolchildren for conducting long multiplication on base-10 integers, but has been modified here for application to a base-2 (binary) numeral system.

Multipliers play an important role in today’s digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation.

AND gates are used to generate the Partial Products, PP, If the multiplicand is N-bits and the Multiplier is M-bits then there is N* M partial product. The way that the partial products are generated or summed up is the difference between the different architectures of various multipliers.

There are several multiplier algorithms used and they too are evolving day to day, some of them are: Array Multiplier, Booth Multipliers Wallace tree Multiplier, Vedic Multiplier, Dadda Multiplier, etc.,

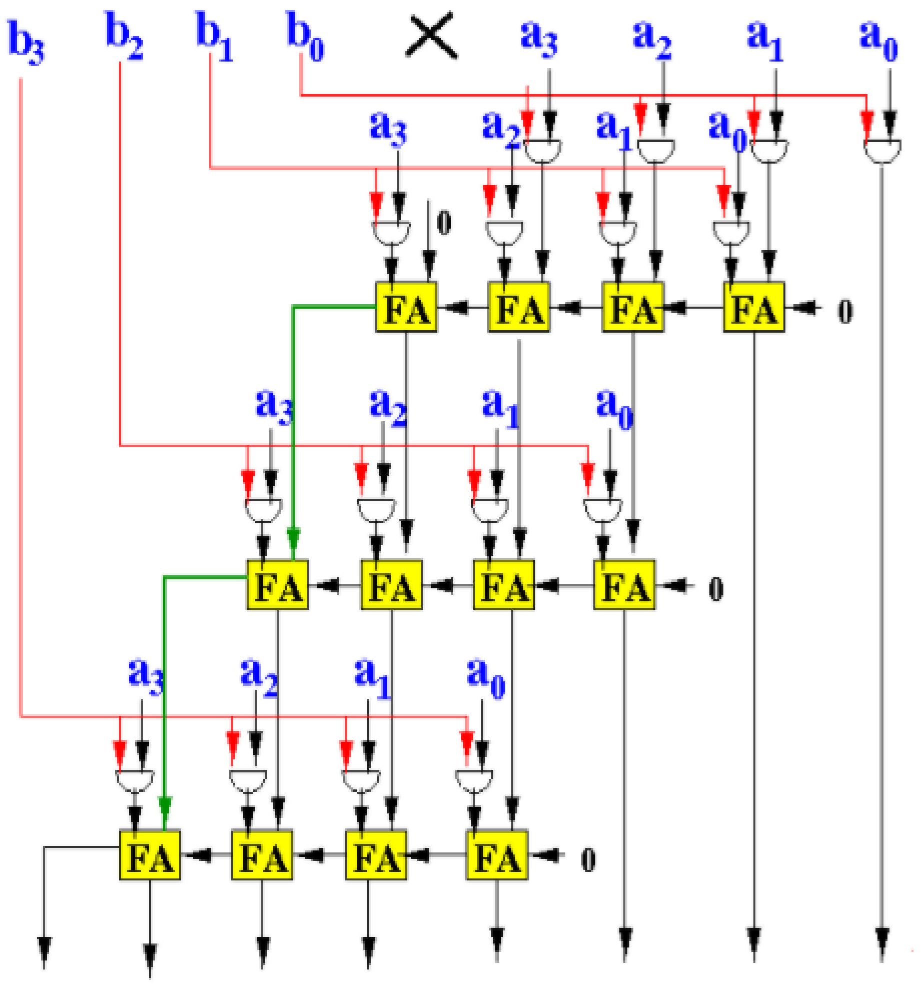


Fig 8.4-Bit Multiplier using Full Adders

2.4 DIVISION

A division algorithm is an algorithm which, given two integers N and D, computes their quotient and/or remainder, the result of division. Some are applied by hand, while others are employed by digital circuit designs and software.

Division algorithms fall into two main categories: slow division and fast division. Slow division algorithms produce one digit of the final quotient per iteration. Examples of slow division include restoring, non-performing restoring, non-restoring, and SRT division. Fast division methods start with a close approximation to the final quotient and produce twice as many digits of the final quotient on each iteration.

2.5 LOGICAL SHIFT

In computer science, a logical shift is a bitwise operation that shifts all the bits of its operand. The two base variants are the logical left shift and the logical right shift. This is further modulated by the number of bit positions a given value shall be shifted, such as shift left by 1 or shift right by n. Unlike an arithmetic shift, a logical shift does not preserve a number's sign bit or distinguish a number's exponent from its significand (mantissa); every bit in the operand is simply moved a given number of bit positions, and the vacant bit-positions are filled, usually with zeros, and possibly ones (contrast with a circular shift).

A logical shift is often used when its operand is being treated as a sequence of bits instead of as a number.

Logical shifts can be useful as efficient ways to perform multiplication or division of unsigned integers by powers of two. Shifting left by n bits on a signed or unsigned binary number has the effect of multiplying it by 2n. Shifting right by n bits on an unsigned binary number has the effect of dividing it by 2n (rounding towards 0).

Logical right shift differs from arithmetic right shift. Thus, many languages have different operators for them. The logical right shift operator is >>, and for left shift is <<.

2.6 LOGICAL ROTATE

In this operation, sometimes called rotate no carry, the bits are "rotated" as if the left and right ends of the register were joined. The value that is shifted into the right during a left-shift is whatever value was shifted out on the left, and vice versa for a right-shift operation. This is useful if it is necessary to retain all the existing bits, and is frequently used in digital cryptography.

CHAPTER 3
DESIGN METHODOLOGY

3.1 FUNCTIONS OF ALU

In this project we have defined 15 functions for the 32 bit ALU, based on the Select signal these functions are selected respectively and executed.

The tabular column for the designed operations for their respective select signals is as follows:

S_3	S_2	S_1	S_0	**Result**	**Operation**
0	0	0	0	a+b	Addition
0	0	0	1	a-b	Subtraction
0	0	1	0	a*b	Multiplication
0	0	1	1	a/b	Division
0	1	0	0	a>>n	Right shift by n bits
0	1	0	1	a<<n	Left shift by n bits
0	1	1	0	ROR a	Rotate right by n bits
0	1	1	1	ROL a	Rotate left by n bits
1	0	0	0	a&b	AND
1	0	0	1	a|b	OR
1	0	1	0	~(a&b)	NAND
1	0	1	1	~(a|b)	NOR
1	1	0	0	a^b	XOR
1	1	0	1	~(a^b)	XNOR
1	1	1	X	~a	NOT

Table 2 Truth Table of ALU

In the above table S_0,S_1,S_2,S_3 are the bits of Select line

3.2 DESIGNING OF 32-BIT ALU

The 32-bit ALU is designed using Verilog code in Xilinx ISE tool. All the functions mentioned in the above section are used to design the ALU. The code for the respected functions is written and is embedded in the single code using decoder, i.e., decoder is used to select the respective function and perform the operation. The code used for ALU is represented in the Verilog code section. The circuit obtained by code converted using Xilinx ISE is shown in the upcoming chapters.

CHAPTER 4
IMPLEMENTATION

4.1 32-Bit ALU

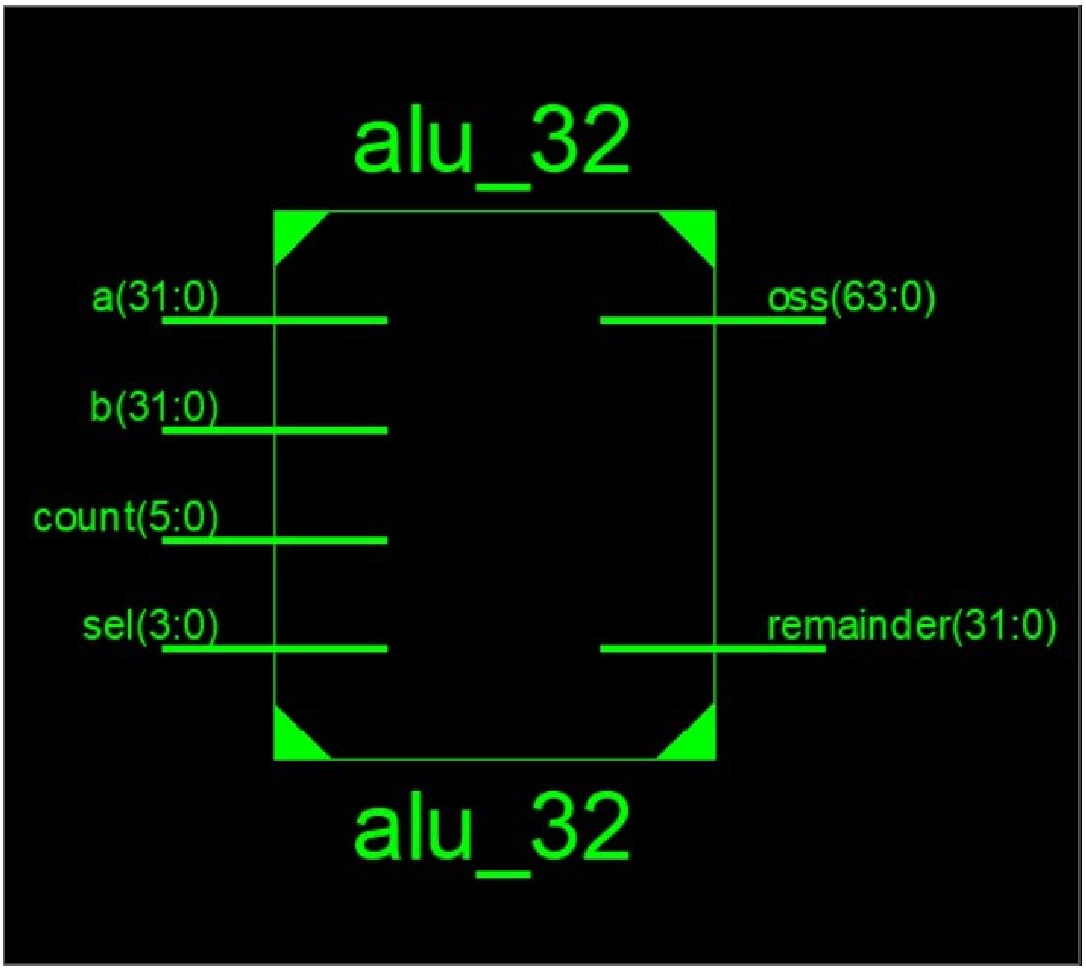


Fig 9 Schematic of 32-bit ALU

4.232-Bit Full Adder

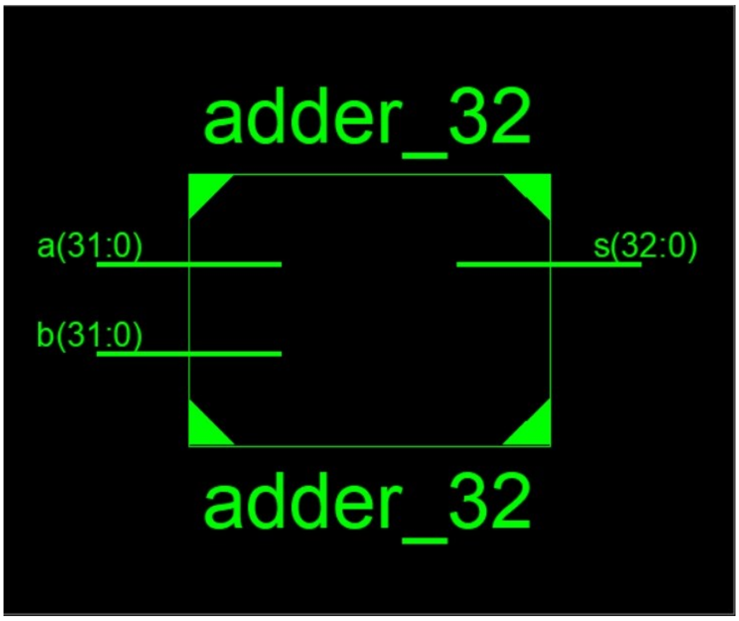


Fig 10 Schematic of 32-bit Adder

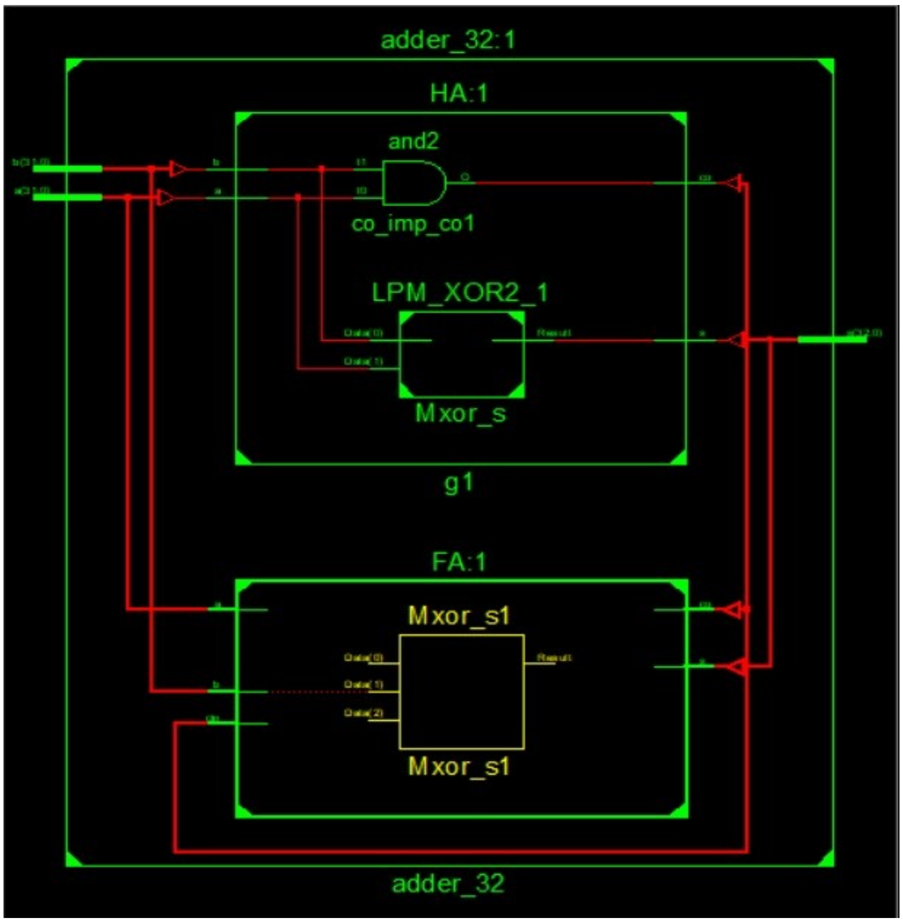


Fig 11 Internal circuit of Adder

4.3 32-Bit Full Subtractor

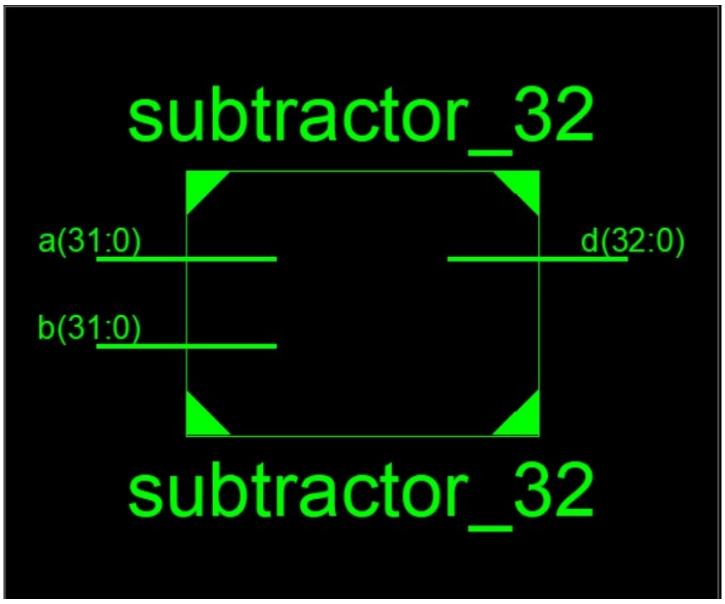


Fig 12 Schematic of 32-bit Subtractor

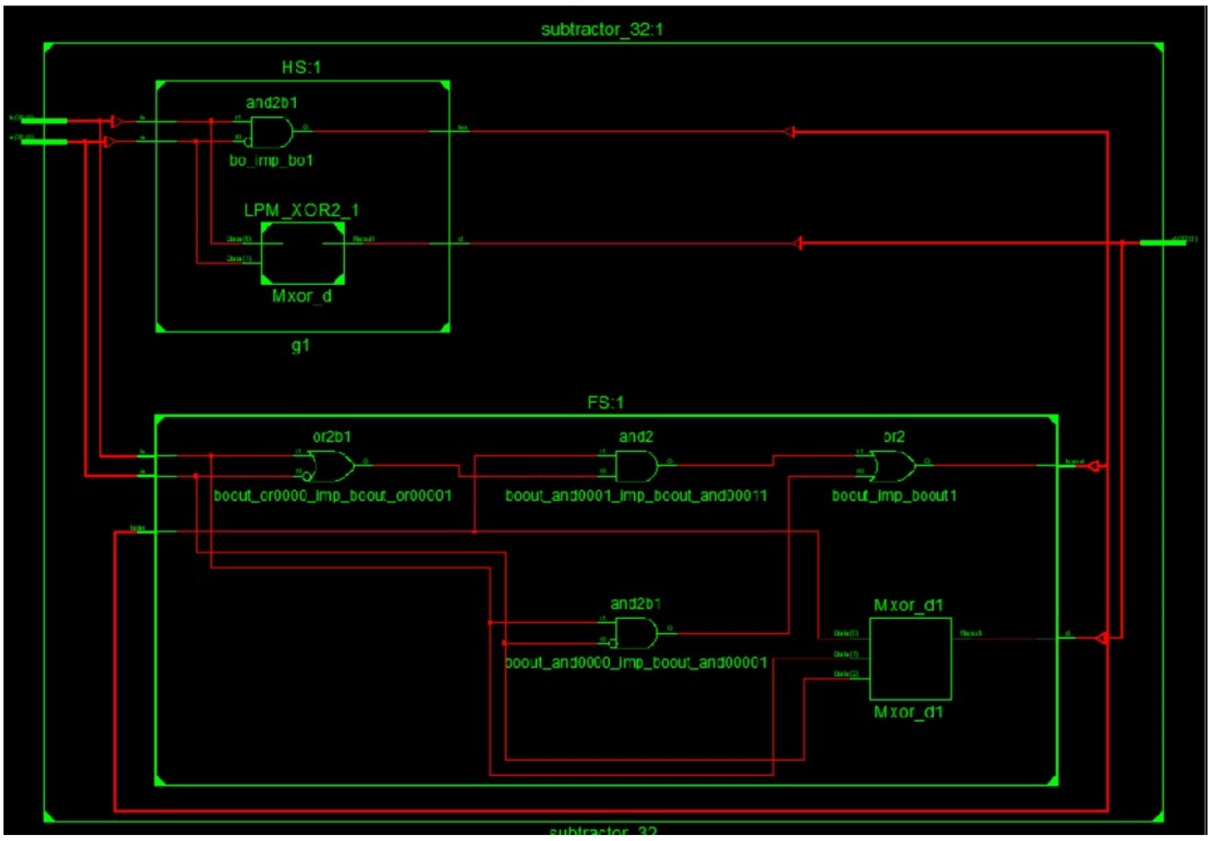


Fig 13 Internal circuit of Subtractor

4.4 32-Bit Multiplier

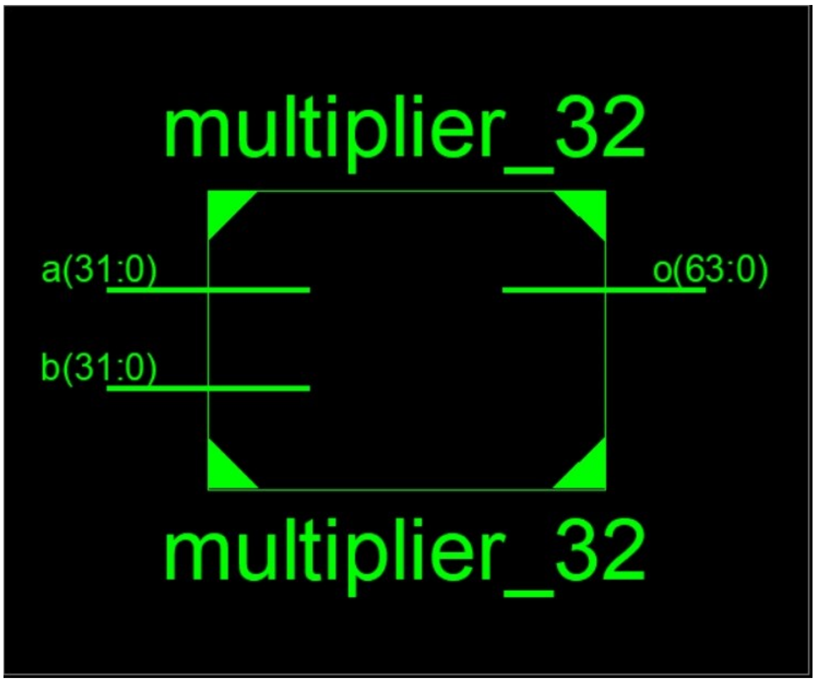


Fig 14 Schematic of 32-bit Multiplier

4.5 32-Bit Division

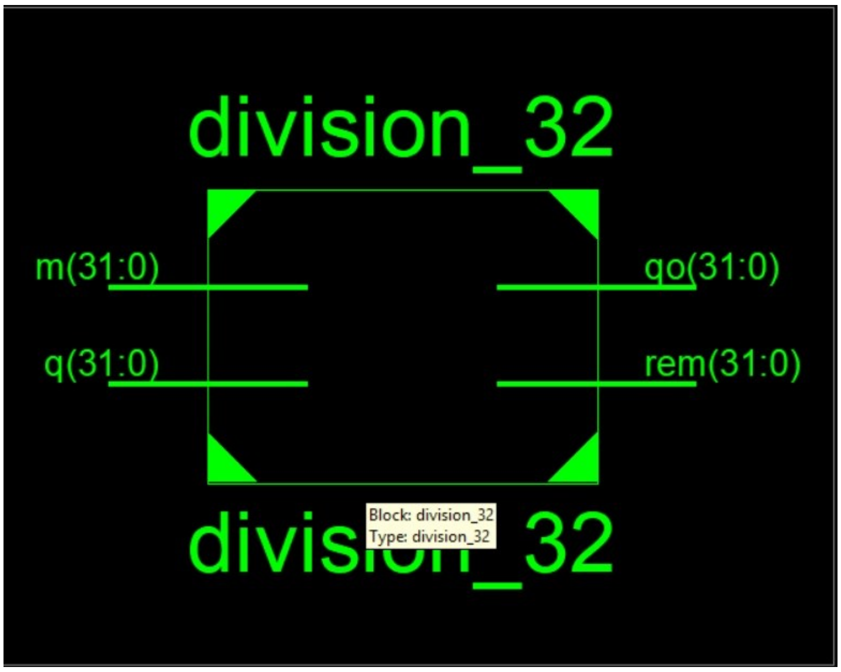


Fig 15 Schematic of 32-bit Division

4.6Left Shift

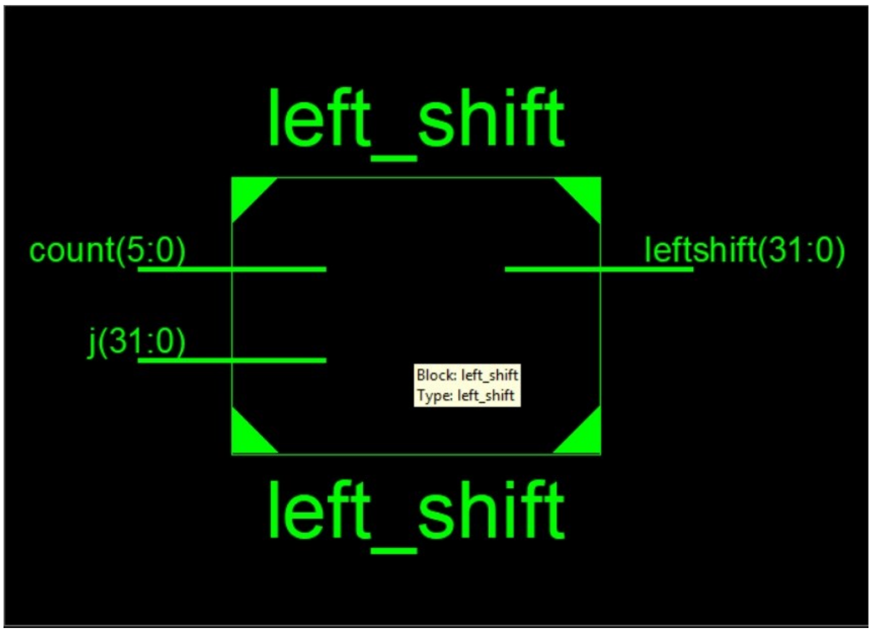


Fig 16 Schematic of N-bit Left Shifter

4.7 Right Shift

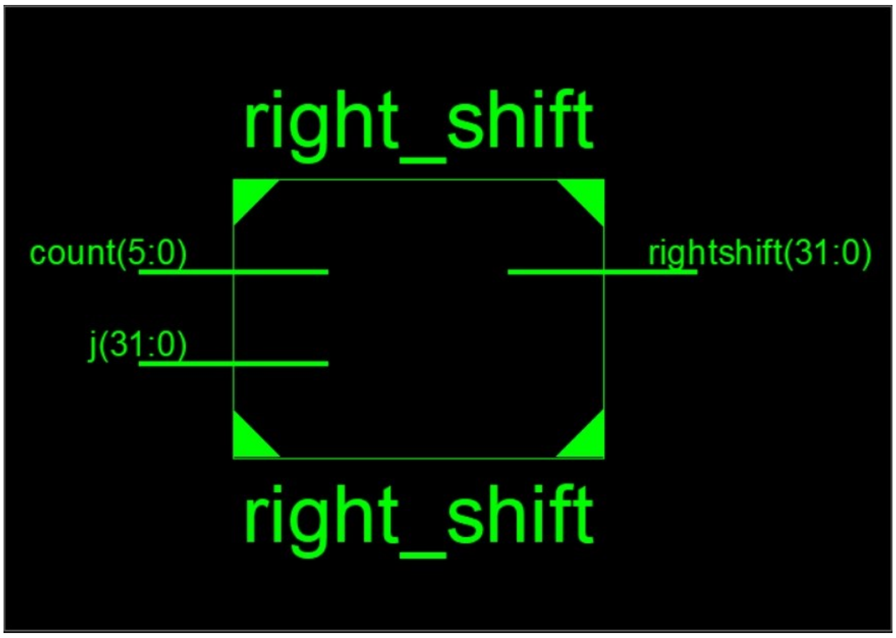


Fig 17 Schematic of N-bit Right Shifter

4.8Rotate Left

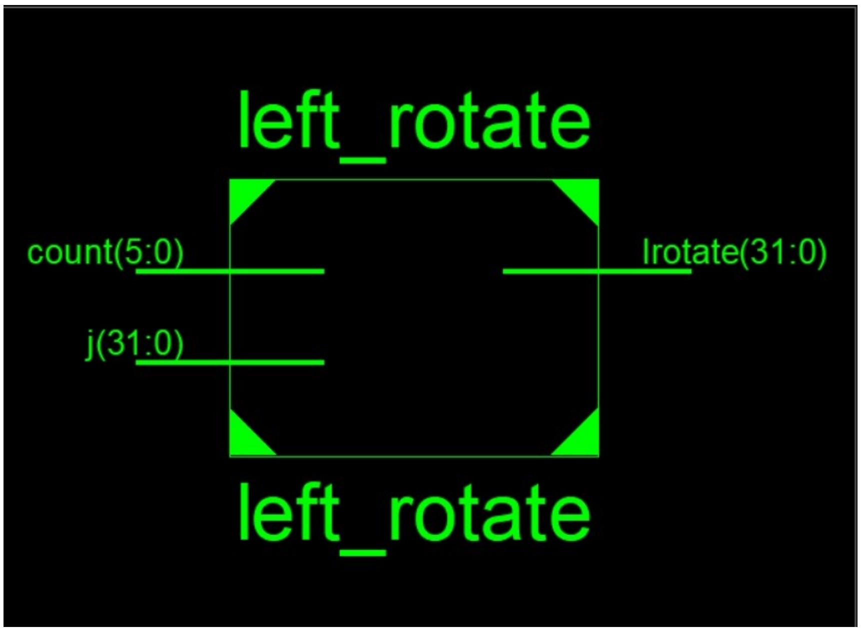


Fig 18 Schematic of N-bit Left Rotate

4.9Rotate Right

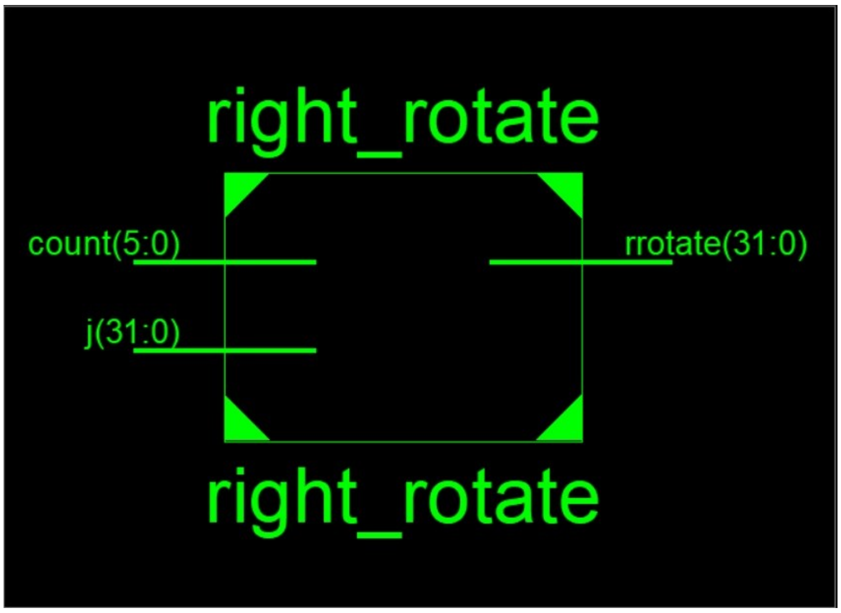


Fig 19 Schematic of N-bit Right Rotate

CHAPTER 5

SIMULATION RESULTS

5.1 ALU

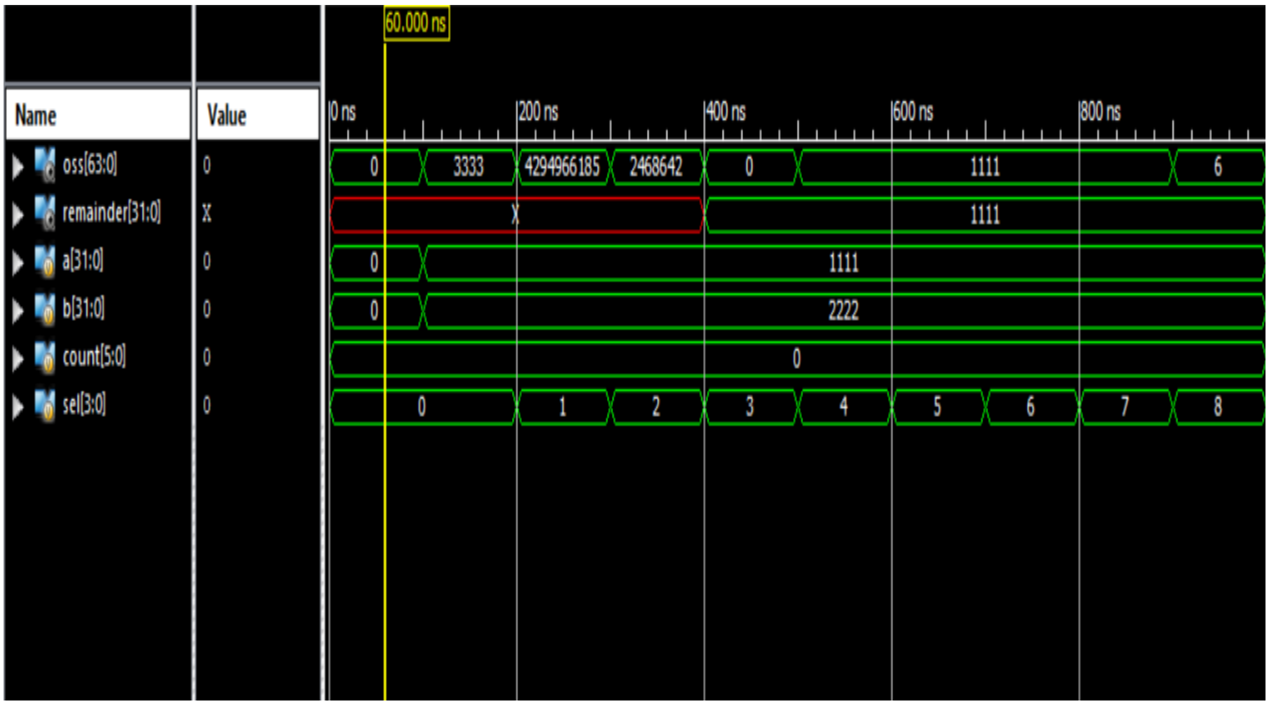


Fig 20 Output waveform of ALU

5.2 Full Adder

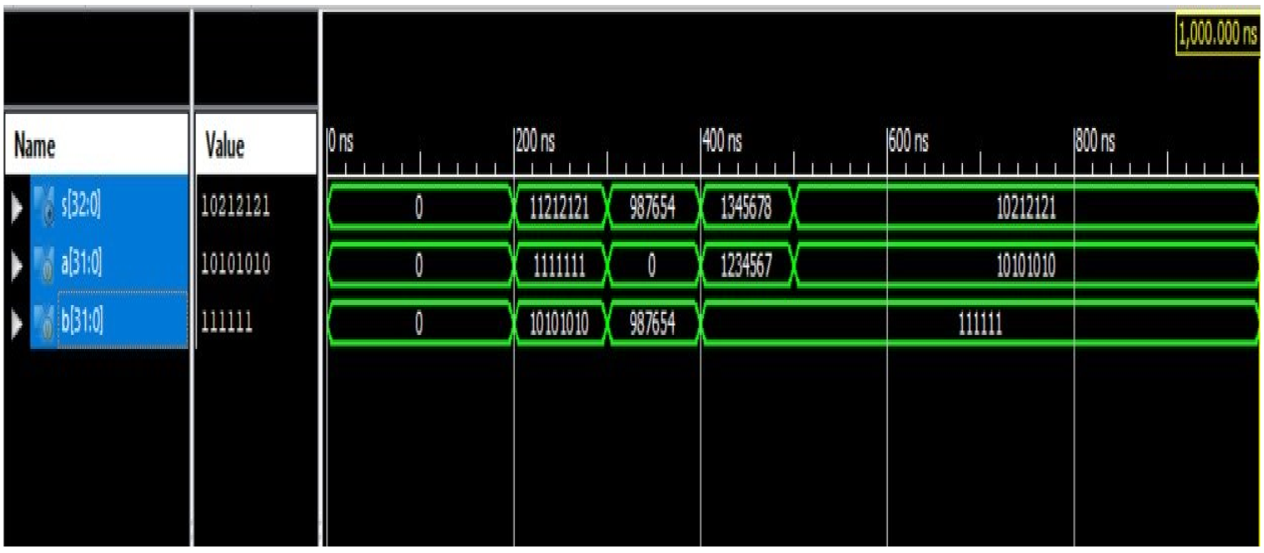
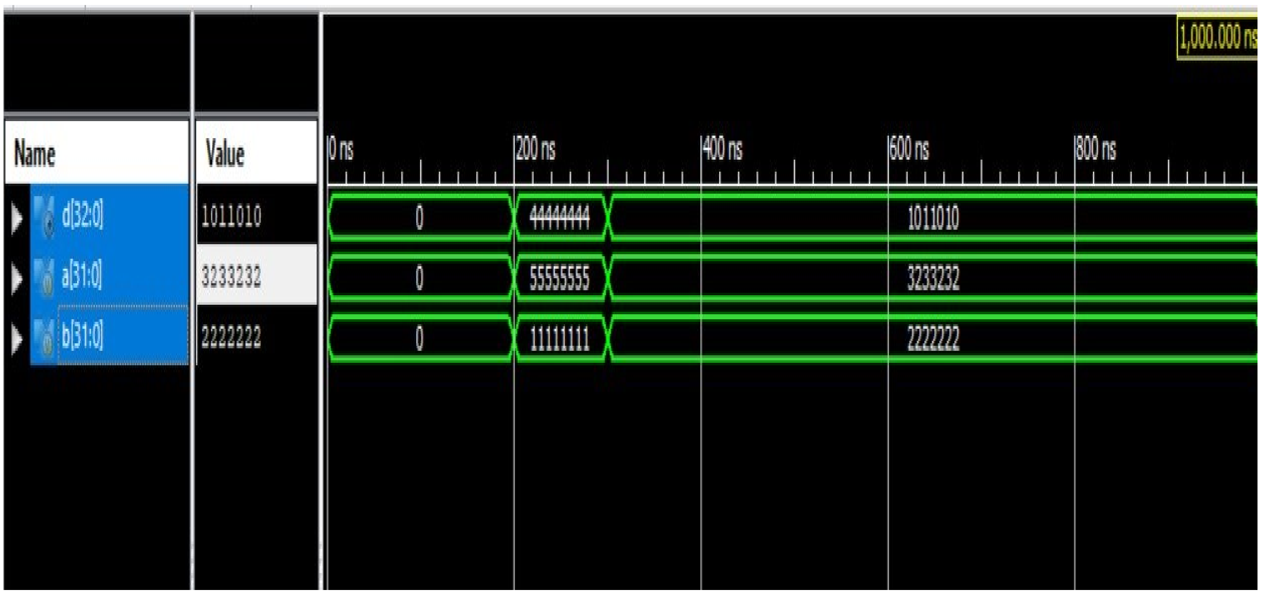


Fig 21 Output waveform of Adder

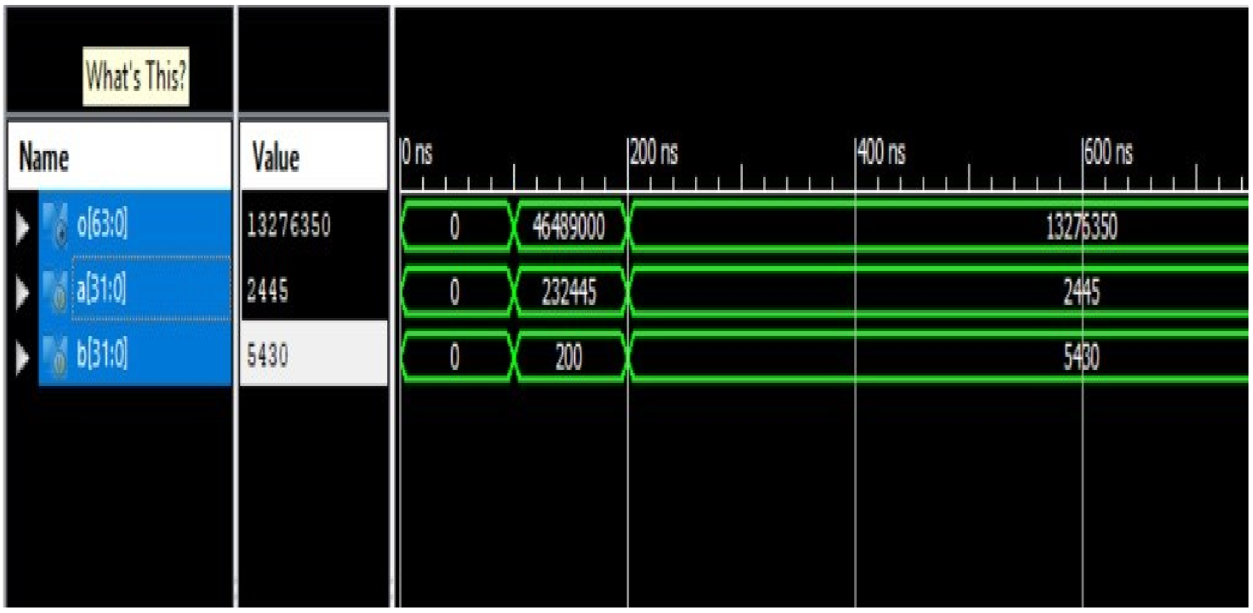
5.3 Subtractor



Name	Value			
d[32:0]	1011010	0	44444444	1011010
a[31:0]	3233232	0	55555555	3233232
b[31:0]	2222222	0	11111111	2222222

Fig 22 Output waveform ofSubtractor

5.4 Multiplier



Name	Value			
o[63:0]	13276350	0	46489000	13276350
a[31:0]	2445	0	232445	2445
b[31:0]	5430	0	200	5430

Fig 23 Output waveform of Multiplier

5.5Division

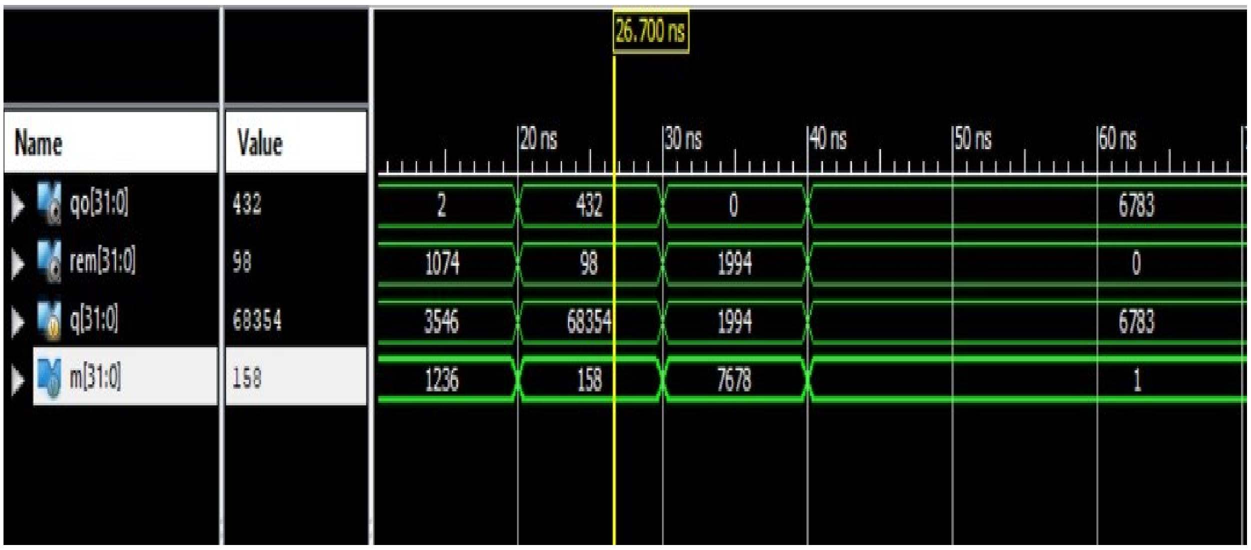


Fig24 Output waveform of Division

5.6Left Shift

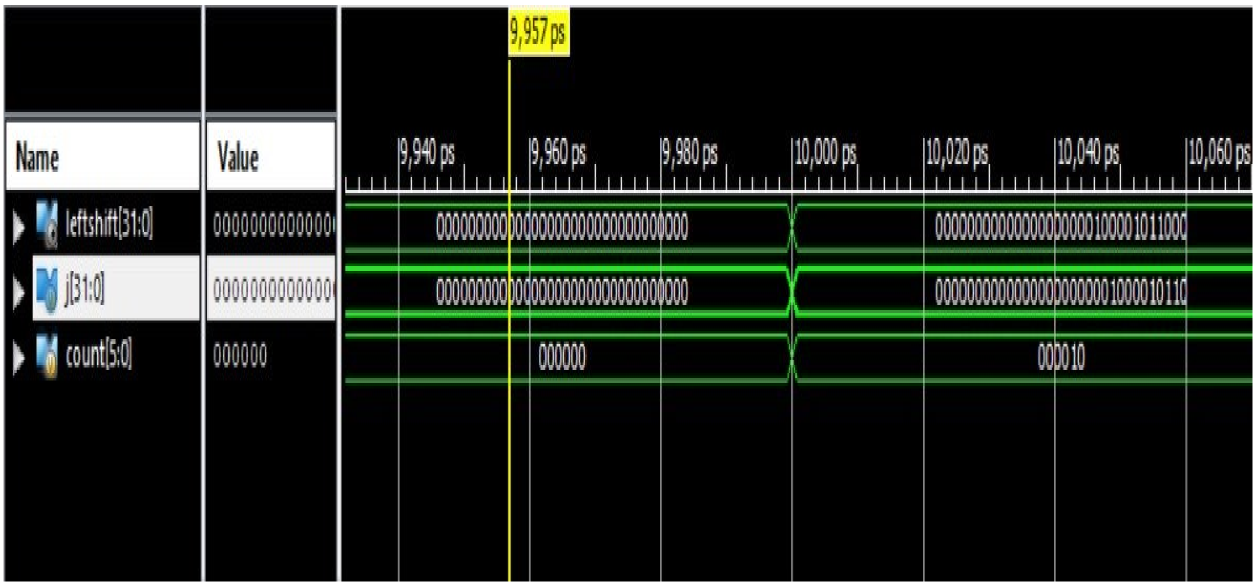


Fig 25 Output waveform of Left Shift

5.7Right Shift

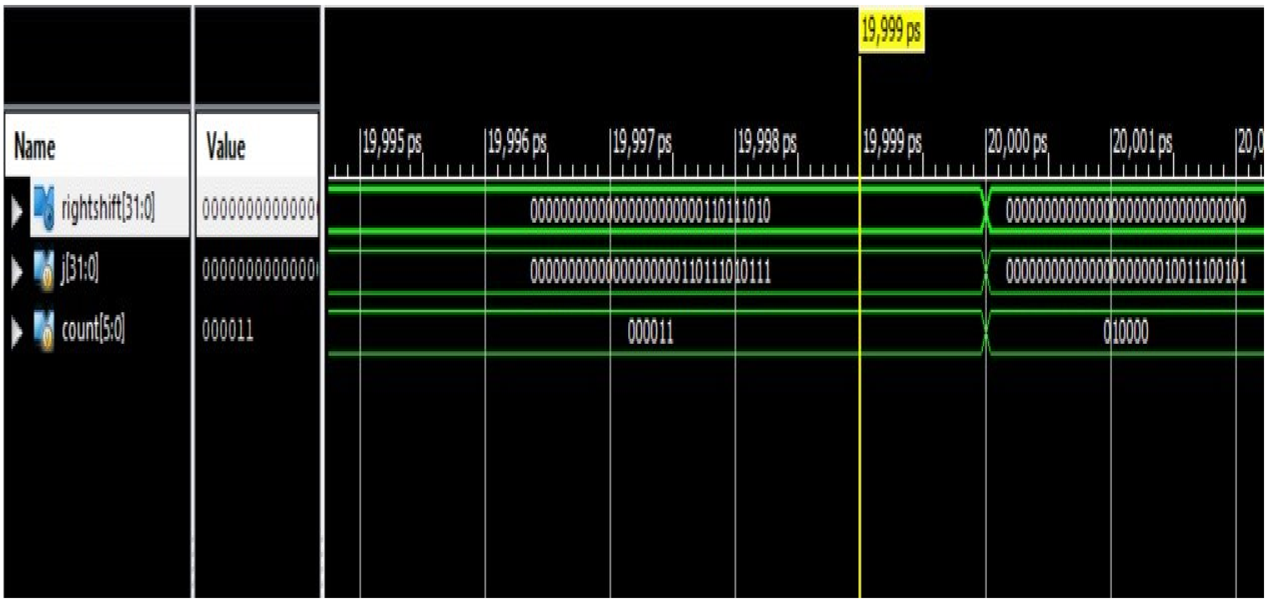


Fig26 Output waveform of Right Shift

5.8Rotate Left

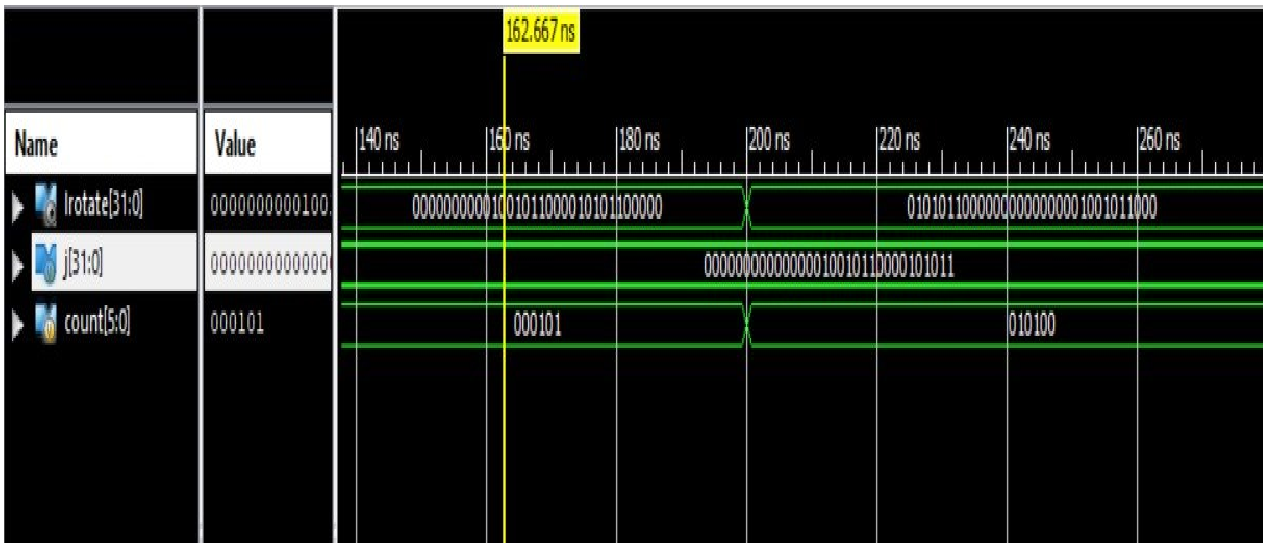


Fig 27 Output waveform of Rotate Left

5.9 Rotate Right

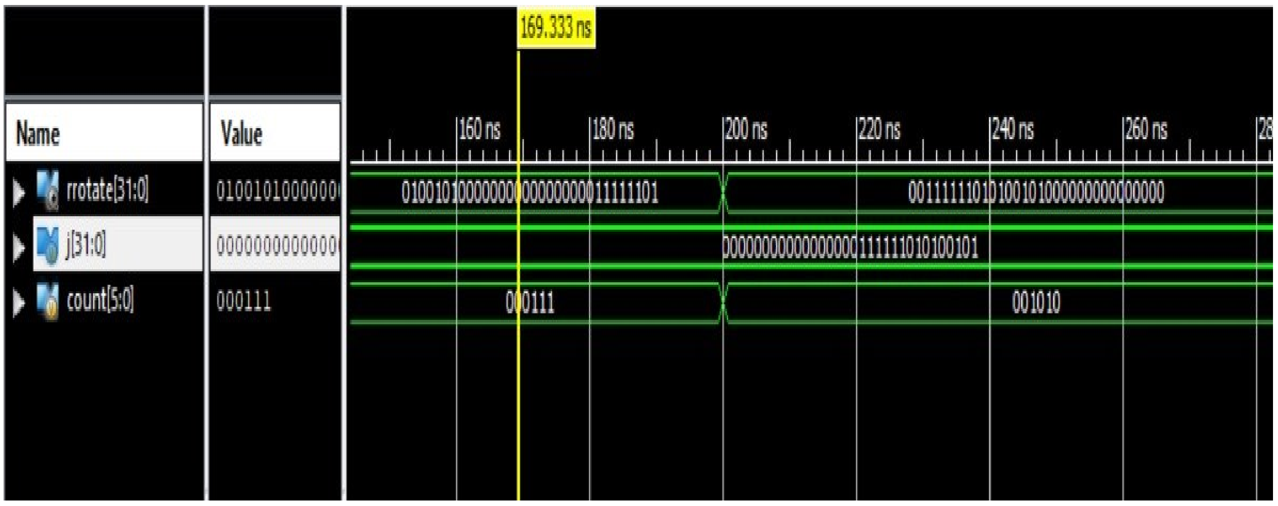


Fig28 Output waveform of Rotate Right

CHAPTER-6

APPLICATIONS

High speed system is very crucial in the critical applications, where the immediate human action is not possible, as in

- Space application.
- Defence surveillance.
- Medical supervisory system.

and other safety related services are the example of such critical applications.

Digital Signal Processing is an important unit in electronics devices. All the fundamental arithmetic operations are performed in many Very Large Scale Integration (VLSI) systems such as

- Digital Signal Processors (DSPs).
- Microprocessors.
- Multiple-precision arithmetic.

ALUs are used for variety of operations in a complex arithmetic circuit like multiplication, division and address calculation.

ADVANTAGES

- The proposed design is capable of producing precise calculations.
- The proposed circuit calculates any functionality in one operation.
- The proposed ALU is capable of complex operations in the system like pipelining, iterative operations, etc.,
- ALU plays a major role in the computing environment and the proposed design achieves it.

CONCLUSION

In our project “Design and Implementation of a 32-bit ALU ”we have designed and implemented a 32 bit ALU. Arithmetic Logic Unit is the part of a computer that performs all arithmetic computations, such as addition and subtraction, increment, decrement, shifting and all sorts of basic logical operations. The ALU is one component of the CPU (Central Processing Unit). Here, using Verilog we have designed a 32 bit ALU which can perform the various arithmetic operations of Addition, Subtraction, Increment, Decrement, Transfer, logical operations such as AND, OR, XOR, NOT and also the shift operation. All the above mentioned operations are then verified to see whether they match theoretically or not. The above given waveforms show that they match completely thereby verifying our results.

FUTURE SCOPE

The ALU can be implemented using different types of Adders, Subtractors, Multipliers and these can be implemented for more number of bits. Power consumption and delay can be decreased further more with the evolution of technologies.

APPENDIX I
XILINX

Xilinx is an American technology company, primarily a supplier of programmable logic devices. It is known for inventing the field-programmable gatearray (FPGA) and as the first semiconductor company with a fables manufacturing model.

Founded in Silicon Valley in 1984, the company is headquartered in San Jose, USA, with additional offices in Longmont -USA; Dublin - Ireland,Singapore,Hyderabad - India,Beijing, China,Shanghai, China,Brisbane, Australia and Tokyo, Japan.

Major FPGA product families include Virtex(high-performance), Kintex (mid-range) and Artix (low-cost), and the retired Spartan (low-cost) series. Major computer software includes Xilinx ISE and Vivado Design Suite.

Ross Freeman, Bernard Vonderschmitt, and James V Barnett II, who all had worked for integrated circuit and solid-state device manufacturer ZilogCorp, founded Xilinx in 1984.

While working for Zilog, Freeman wanted to create chips that acted like a blank tape, allowing users to program the technology themselves. At the time, the concept was paradigm changing. "The concept required lots of transistors and, at that time, transistors were considered extremely precious—people thought that Ross's idea was pretty far out", said Xilinx Fellow Bill Carter, who when hired in 1984, as the first IC designer, was Xilinx's eighth employee.

Big semiconductor manufacturers were enjoying strong profits by producing massive volumes of generic circuits. Designing and manufacturing dozens of different circuits for specific markets offered lower profit margins and required greater manufacturing complexity. What became known as the FPGA would allow circuits produced in quantity to be tailored by individual market segments.

Reeman failed to convince Zilog to invest in creating the FPGA to chase what was only a $100 million market at the time. Freeman and Barnett left Zilog and teamed up with their 60-year-old ex-colleague Bernard Vonderschmitt to raise $4.5 million in venture funding to design the first commercially viable FPGA. They incorporated the company in 1984 and began selling its first product by 1985.

By late 1987 the company had raised more than $18 million in venturecapital(equivalent to $37.95 million in 2016) and generated revenues at an annualized rate of nearly $14 million.

Growth

As demand for programmable logic continued to grow, so did Xilinx's revenues and profits. From 1988 to 1990, the company's revenue grew each year from $30 million to $50 million to $100 million. During this time period, the company which had been providing funding to Xilinx, Monolithic Memories Inc. (MMI), was purchased by Xilinx competitor AMD. As a result, Xilinx dissolved the deal with MMI and went public on the NASDAQ in 1989. The company also moved to a 144,000-square-foot (13,400 m^2) plant in San Jose, California in order to keep pace with demand from companies like HP, AppleInc., IBM and Sun Microsystems who were buying large quantities from Xilinx.

Xilinx competitors emerged in the FPGA market in the mid-1990s. Despite the competition, Xilinx's sales grew to $135 million in 1991, $178 million in 1992 and $250 million in 1993. The company reached $550 million in revenue in 1995, one decade after having sold its first product. According to market research firm iSuppli, Xilinx has held the lead in programmable logic device market share since the late 1990s. Over the years, Xilinx expanded operations to India, Asia and Europe.

Xilinx's sales rose from $560 million in 1996 to $2.2 billion by the end of its fiscal year 2013.[15][16] Moshe Gavrielov– an EDA and ASIC industry veteran who was appointed as president and CEO in early 2008 – introduced targeted design platforms that combine FPGAs with software, IP cores, boards and kits to address focused target applications. These targeted design platforms are an alternative to costly application-specific integrated circuits (ASICs) and application-specific standard products (ASSPs).

Today

The company has expanded its product portfolio since its founding. Xilinx sells a broad range of FPGAs, complex programmable logic devices (CPLDs), design tools, intellectual property and reference designs. Xilinx also has a global services and training program.

After using the introduction of 3D chips to deliver more powerful FPGAs, Xilinx then adapted the technology to combine formerly separate components in a single chip, first combining an FPGA with transceivers to boost bandwidth capacity while using less power.

According to Xilinx CEO Moshe Gavrielov, the addition of a heterogeneous communications device, combined with the introduction of new software tools and the Zynq-7000 line of 28 nm SoC devices that combine an ARM core with an FPGA, are part of shifting its position from a programmable logic device supplier to one delivering “all things programmable”.

The company's products have been recognized by EE Times, EDN and others for innovation and market impact. In addition to Zynq-7000, Xilinx product lines (see Current Family Lines) include the Virtex,Kintex and Artix series, each including configurations and models optimized for different applications. With the introduction of the Xilinx 7 series in June, 2010, the company has moved to three major FPGA product families, the high-end Virtex, the mid-range Kintex family and the low-cost Artix family, retiring the Spartan brand, which ends with the Xilinx Series 6 FPGAs. In April 2012, the company introduced the Vivado Design Suite - a next-generation SoC-strength design environment for advanced electronic system designs. In May, 2014, the company shipped the first of the next generation FPGAs: the 20 nm UltraScale.

Verilog HDL

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design andverification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of geneticcircuits. Hardware description languages such as Verilog differ from software programming languages because they include ways of describing the propagation time and signal strengths (sensitivity). There are two types of assignment operators; a blocking assignment (=), and a non-blocking (<=) assignment. The non-blocking assignment allows designers to describe a state-machine update without needing to declare and use temporary storagevariables. Since these concepts are part of Verilog's language semantics, designers could quickly write descriptions of large circuits in a relatively compact and concise form. At the time of Verilog's introduction (1984), Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits.

A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional ports. Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

Verilog's concept of 'wire' consists of both signal values (4-state: "1, 0, floating, undefined") and signal strengths (strong, weak, etc.). This system allows abstract modelling of shared signal lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire's (readable) value is resolved by a function of the source drivers and their strengths.

A subset of statements in the Verilog language is synthesizable. Verilog modules that conform to a synthesizable coding style, known as RTL (register-transfer level),can be physically realized by synthesis software. Synthesis software algorithmically transforms the (abstract) Verilog source into a net list, a logically equivalent description consisting only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are available in a specific FPGA or VLSI technology. Further manipulations to the net list ultimately lead to a circuit fabrication blueprint (such as a photo mask set for an ASIC or a bit stream file for an FPGA).

System tasks are available to handle simple I/O and various design measurement functions during simulation. All system tasks are prefixed with **$** to distinguish them from user tasks and functions. This section presents a short list of the most frequently used tasks. It is by no means a comprehensive list.

- $display —Print to screen a line followed by an automatic newline.
- $write — Write to screen a line without the newline.
- $swrite— Print to variable a line without the newline.
- $sscanf— Read from variable a format-specified string. (*Verilog-2001)
- $fopen— Open a handle to a file (read or write)
- $fdisplay— Write to file a line followed by an automatic newline.
- $fwrite— Write to file a line without the newline.
- $fscanf— Read from file a format-specified string. (*Verilog-2001)
- $fclose—Close and release an open file handle.
- $readmemh— Read hex file content into a memory array.
- $readmemb— Read binary file content into a memory array.
- $monitor — Print out all the listed variables when any change value.

BIBILOGRAPHY

1) T. K. Ghosh and A. J. Pal, Computer Organization and Architecture, Tata McGraw-Hill Publishing Company Limited, New Delhi, 2009.

2) Douglas L. Perry, VHDL Programming by Example, 4th ed., Tata McGraw-Hill Publishing Company Limited, New Delhi, 2002.

3) M. Morris Mano, “Register transfer and Microoperations,” in Computer System Architecture, 3rd ed. Pearson, India, pp. 106-118.

4) S. Salivahanan and S. Arvazhagan, “Digital Circuits and Design”, 3rd ed. Vikas publishing, India, pp. 168-169.

5) R.Uma and P.Dhavachelvan (2012, August). Logic optimization using technology independent MUX based adders in FPGA.International Journal of VLSI design & Communication Systems.[online].3 (4), pp. 135-147.

6) KanikaKaur and Arti Noor (2011,May). Strategies and Methodologies for low power VLSI designs: A review. International Journal of Advances in Engineering & Technology.[online]. ISSN:2231-1963.pp.159-164.

7) Rajeev Kumar (2012, June).Design & Implementation of 64 bit ALUfor Instruction Set Architecture & Comparison between Speed/PowerConsumption on FPGA.International Journal of Advanced Researchin Computer Engineering & Technology.[online]. 1(4), pp.186-192

8)BibhashSen, ManojitDutta, DebajyotyBanik, Dipak K Singh, Biplab K Sikdar, “Design of Fault Tolerant Reversible Arithmetic Logic Unit in QCA” 2012 International Symposium on Electronic System Design.

9H. Thapliyal and M. Srinivas, "Novel Reversible 'TSG' Gate and Its Application for Designing Components of Primitive Reversible/Quantum ALU," Tenth Asia-Pacific Computer Systems Architecture Conference, 2005.

10) H. Thapliyal and M. Srinivas, "Novel Reversible 'TSG' Gate and Its Application for Designing Components of Primitive Reversible/Quantum ALU," Tenth Asia-Pacific Computer Systems Architecture Conference, 2005.

11) M.H. Ghadiry, A.K. A'ain, M. Nadi, Design and analysis of a novel low PDP fulladder cell, J. Circuits Syst. Compute. (2011) 439–445.

12) G. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, "On the Use of Signed Digit Arithmetic for the New 6-Inputs LUT Based FPGAs," Proc. IEEE 15th Int'l Conf. Electronics, Circuits and Systems (ICECS), pp. 602-605, 2008.

13) M. Jhamb, H. Lohani, Design, implementation and performance comparison ofmultiplier topologies in power-delay space, Eng. Sci. Technol. Int. J. (2016) 355–363.

14) S. Ghosh, K. Roy, Novel low overhead post-silicon self-correction technique forparallel prefix adders using selective redundancy and adaptive clocking, IEEETrans. Very Large Scale Integr. VLSI Syst. (2011) 1504–1507.

15) I. Levi, O. Bass, A. Kaizerman, A. Belenky, and A. Fish, "High speed dual mode logic carry look ahead adder," in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 3037– 3040.

16) T.J.Todman, G.A.Constantinides, S.J.E.Wilton, O.Mencer, W.Lunk and P.Y.K.Cheung, "Reconfigurable computering: architectures and design methods", IEEE Pro.-Comput.Digit.Tech.,Vol.,152,No. 2,March 2005.

17) A.Srivasta and C.Srinivasan, "ALU DESIGN USING RECONFIGURABLE CMOS LOGIC", Department of Electrical and Computer Engineering Louisiana State University, LA 70803,USA.

18) Makoto Suzuki, Norio Ohkubo and Toshiaki Yamanaka, "A 1.5-ns 32-b CMOS ALU in Double Pass-Transistor Logic", IEEE JOURNAL OF SOLID-STATE CIRCIUTS.

19) W.W. Peterson, E.J. Weldon, Error-correcting Codes, MIT press, 1972.

20) M. Nicolaidis, Carry checking/parity prediction adders and ALUs, IEEE Trans.Very Large Scale Integr. VLSI Syst. (2003) 121–128.

21) J. M Rabey, Digital Integrated Circuits- A Design Perspective, Prentice Hall, 1996.

22) C. Khedhiri, M. Karmani, B. Hamdi, KaLok Man, Concurrent error detectionadder based on two paths output computation, in: IEEE InternationalSymposium on parallel and Distributed Processing with ApplicationsWorkshops (ISPAW), 2011.

23) J. Von Neumann, Probabilistic logics and the synthesis of reliable organismsfrom unreliable components, Automata Stud. (1956) 43–98.

24) D. Siewiorek, R. Swarz, Reliable Computer Systems: Design and Evaluation,Digital Press, 2014.

25) P. Reviriego, C.J. Bleakley, J.A. Maestro, Diverse double modular redundancy: anew direction for soft-error detection and correction, IEEE Des. Test. (2013) 87–95.

26) M. Valinataj, A novel self-checking carry look ahead adder with multiple errordetection/correction, Micro process. Microsystems.(2014) 1072–1081.

27) A. Majumdar, S. Nayyar, J.S Sengar, Fault tolerant ALU system, IEEEInternational Conference on Computing Sciences (ICCS), 2012.

28) E. Sicard, S. Ben Dhia "Basic CMOS cell design", Tata McGraw-Hill, ISBN 0-07-059933-5, 2005.

29)Neil Weste, David Harris, "CMOS VLSI Design: A Circuits and Systems Perspective" , 4th Ed.,, Pearson Ed. , ISBN 13: 978-0- 321-54774-3, 2011.

30) Microwind – CMOS Layout design and Simulation tool by Dr. Etienne Sicard, Microwind, France.

31) M. Fazeli, A. Namazi, S.G. Miremadi, A.Haghdoost, Operand width awarehardware reuse: a low cost fault-tolerant approach to ALU design in embeddedprocessors, Microelectron. Reliab.(2011) 2374–2387.